IBM

Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking

- Understand CS for z/OS TCP/IP security and policy capabilities
- See CS for z/OS security and policy implementation examples
- Protect your z/OS networking environment

Bill White
Gwen Dente
Octavio L Ferreira
Shizuka Katoh
Shuo Ni
Matt Nuttall
Yohko Ojima
Joel Porterie
Larry Templeton

Redbooks

ibm.com/redbooks

International Technical Support Organization

CS for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking

June 2008

SG24-7535-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (June 2008)

This edition applies to Version 1, Release 9, of Communications Server for z/OS.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at:
http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®
DB2®
eServer™
FICON®
HiperSockets™
IBM®
Language Environment®
MVS™
NetView®
OMEGAMON®
Parallel Sysplex®
RACF®
RDN™
Redbooks®
Redbooks (logo) ®
REXX™
RMF™
System z™
System z9®
System/360™
Tivoli®
VTAM®
z/OS®
z9™
zSeries®

The following terms are trademarks of other companies:

Java, SNM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ESP, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

For more than 40 years, IBM® mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z™, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360™ heritage. Likewise, its IBM z/OS® operating system is far superior to its predecessors in providing, among many other capabilities, world class and state-of-the-art support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for even more secure, scalable, and highly available mainframe TCP/IP implementations.

The *Communications Server for z/OS TCP/IP Implementation* series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of CS for z/OS TCP/IP.

This IBM Redbooks® publication, *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 4: Security and Policy-Based Networking*, SG24-7535, explains how to set up security for your z/OS networking environment. With the advent of TCP/IP and the Internet, network security requirements have become more stringent and complex. Because many transactions come from unknown users and from untrusted networks such as the Internet, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity.

In addition, there are certain applications shipped with TCP/IP such as File Transfer Protocol (FTP) that, without proper configuration and access controls in place, could allow unauthorized users access to system resources and data. The z/OS V1R9 Communications Server, along with other elements of z/OS, provides integrated IP security functions to address these TCP/IP security concerns, including:

- Protecting data privacy and integrity while in the network

  The Communications Server protects data in the network using secure protocols based on cryptography, such as IP Security (IPSec), Secure Sockets Layer (SSL), and Transport Layer Security (TLS). IPSec and TLS implementations with z/OS Communications Server are shown in Chapter 9, “IP Security” on page 373, and Chapter 11, “Application Transparent Transport Layer Security” on page 443, respectively.

Note: This book is focused on providing CS for z/OS TCP/IP implementation guidance. For readers who are unfamiliar with cryptographic technologies, technology overviews for cryptography are included in Appendix A, “Basic cryptography” on page 657.

- Protecting the system from the network

  The Communications Server is responsible for protecting the system against denial-of-service attacks from the network. It has built-in defenses and also provides several services that an installation can optionally deploy to protect against these attacks. Chapter 8, “IP filtering” on page 319, and Chapter 12, “Intrusion Detection Services” on page 471, provide implementation examples for those technologies.

- Protecting system resources and data from unauthorized access

  Communications Server applications and the TCP/IP protocol stack protect data and resources on the system using standard Security Access Facility (SAF)-based services. Part 1, “SAF-based security” on page 1, provides high-level discussion and implementation details regarding the IBM Resource Access Control Facility (RACF®) product.

Finally, because security technologies are complex and can be confusing, we have included helpful tutorial information in the appendixes of this book.

For more specific information about CS for z/OS base functions, standard applications, and high availability, refer to the other volumes in the series. These are:

- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance*, SG24-7534

In addition, *z/OS Communications Server: IP Configuration Guide*, SC31-8775, *z/OS Communications Server: IP Configuration Reference*, SC31-8776, and *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780, contain comprehensive descriptions of the individual parameters for setting up and using the functions described in this book. They also include step-by-step checklists and supporting examples.

It is not the intent of this book to duplicate the information in those publications, but to complement them with practical implementation scenarios that can be useful in your environment. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771. For complete details, we encourage you to review the documents referred to in the additional resources section at the end of each chapter.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Bill White is a Project Leader and Senior Networking Specialist at the International Technical Support Organization, Poughkeepsie Center.

Gwen Dente has spent 27 years with IBM, focusing on mainframe networking and security support for customers and the IBM technical community. She is a Consulting IT Specialist with IBM Advanced Technical Support at the Washington Systems Center in Gaithersburg, Maryland (USA). Gwen presents frequently at SHARE and other IBM technical conferences, and has co-authored other IBM Redbooks publications.

Octavio L Ferreira is a Senior IT Specialist in IBM Brazil. He has 28 years of experience in IBM software support. His areas of expertise include z/OS Communications Server, SNA and TCP/IP, and Communications Server in all platforms. For the last 10 years, Octavio has worked with the Area Program Support Group, providing guidance and support to clients and designing networking solutions such as SNA/TCP/IP integration, z/OS Connectivity, Enterprise Extender design and implementation, and SNA-to-APPN migration. He has also co-authored other IBM Redbooks publications.

Shizuka Katoh is an Advisory IT Specialist working for Advanced Technical Support in IBM Japan. She has more than 10 years of experience in the area of mainframe networking. Her areas of expertise include z/OS Communications Server, TCP/IP (IPv4 and IPv6), VTAM®, and OSA. Shizuka specializes in networking solutions for Parallel Sysplex® and High Availability Data Center environments.

Shuo Ni is an Advisory IT Specialist in IBM Global Services, China. Her areas of expertise include z/OS, TCP/IP, VTAM (Subarea and APPN), OSA-Express, HiperSockets™, WLM, and CICS®. For the last six years, Shuo has provided onsite service support to mainframe clients in southern China for z/OS Communication Server (including Enterprise Extender and CICS TCP/IP Sockets), as well as other mainframe subsystems.

Matt Nuttall has worked for IBM Canada for almost two decades. He has co-authored numerous IBM Redbooks publications, as well as a New Mainframe college textbook. Matt continues to specialize in networking and security.

Yohko Ojima is an IT Specialist in IBM Japan and has five years of experience in the area of enterprise networking. She specializes in TCP/IP and SNA networks with Communications Server products, routers, and switches. Yohko's responsibilities include providing technical support and reviewing customer networking design, including Communications Server for z/OS (TCP/IP and VTAM), Communications Server for distributed servers, OSA, and Communication Controller for Linux® on System z.

Joel Porterie is a Senior IT Specialist with 30 years of experience with IBM France. He works for Network and Channel Connectivity Services in the EMEA Product Support Group. His areas of expertise include z/OS, TCP/IP, VTAM, OSA-Express, and Parallel Sysplex. Joel has taught OSA-Express and FICON® problem determination classes and provided onsite assistance for these areas in numerous countries. He has also co-authored many IBM Redbooks publications.

Larry Templeton is a Network Architect with IBM Global Services, Network Outsourcing. He has 38 years of experience with IBM mainframe and networking systems, consulting with clients around the world. Larry's current responsibilities include architecting secure mainframe IP connectivity solutions, designing inter-company Enterprise Extender configurations, and assisting clients with high-availability data center implementations. He also co-authored other IBM Redbooks publications.

Thanks to the following people from the ITSO, Poughkeepsie Center, for their contributions to this project: David Bennin, Ella Buslovich, Rich Conway, Bob Haimowitz, and Lydia Parziale.

As is always the case with any complex technical effort, success would not have been possible without the advice, support, and review of many outstanding technical professionals. We are especially grateful for the significant expertise and contributions of content to this book from the Communications Server for z/OS Development team, especially from Jim Ash, Allen Bailey, Doris Bunn, Greg Callis, Alfred Christensen, Greg Curfman, Erin Farr, Jon Franks, Sara Haggar, Jay McKenna, Christopher Meyer, Scott Moonen, Alan Packett, Joyce Anne Porter, Marc Price, Dianne Shannon, Andy Tracy, Doug Trottman, Janet Wolf, and Mark Wright.

Finally, we would like to thank the authors of the previous *Communications Server (CS) for z/OS TCP/IP Implementation* series for creating the groundwork for this series: Rama Ayyar, Valirio Braga, Gilson Cesar de Oliveira, Octavio Ferreira, Adi Horowitz, Michael Jensen, Sherwin Lake, Bob Louden, Garth Madella, Yukihiko Miyamoto, Roland Peschke, Joel Porterie, Marc Price, Larry Templeton, Rudi van Niekerk, and Thomas Wienert.

Become a published author

Join us for a two- to six-week residency program! Help write an book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Part 1

SAF-based security

In this part, we explain how you can use the z/OS Security Access Facility (SAF) to protect your network and communications. SAF is the high-level infrastructure that allows you to plug in any commercially available security product. This book specifically focuses on the IBM Resource Access Control Facility (RACF) element of the z/OS Security Server.

Important: Many of the tasks, examples, and references in this book assume that you are using the z/OS Security Server element RACF. References to RACF apply to any other SAF-compliant security products that contain the required support. If you are using another security product, read the documentation for that product for instructions about task performance.

1

RACF demystified

In this chapter, we explain how you can use the IBM Resource Access Control Facility (RACF) to protect your network and communications. RACF uses the z/OS System Authorization Facility (SAF) to control access to resources such as data set and MVS™ commands. In a networking context, RACF resources can include networks and network services.

SAF is the high level MVS interface that allows you to plug in to any SAF-compatible security product. Although any product can use this interface, we only discuss RACF in this chapter. For the equivalent capabilities in other SAF security products, refer to that product's documentation.

The following topics are discussed in this chapter.

Section	Topic
1.1, "RACF - basic concepts" on page 4	Basic RACF concepts explained in simple terms
1.2, "Protecting your network resources" on page 6	How TCP/IP resources (stack, ports, commands, and so on) are protected by RACF
1.3, "Protecting your programs" on page 7	Concepts of program protection
1.4, "Associating a user ID with a started task" on page 8	How RACF correlates userids to STCs
1.6, "RACF multilevel security for network resources" on page 9	Basic concepts of MLS
1.7, "Digital certificates in RACF" on page 10	RACF support for keys and certificate management

1.1 RACF - basic concepts

RACF has evolved over more than 30 years to provide protection for a variety of resources, features, facilities, programs, and commands on the z/OS platform. Because of its vast array of commands and numerous methods of protection, it is easy to become confused by RACF. In this chapter, we try to demystify RACF by explaining the basic concepts.

The RACF concept is very simple: it keeps a record of all the resources that it protects in the RACF database. What is a resource? A *resource* can be a data set, a program, and even a subnetwork. RACF can, for example, set permissions for file patterns even for files that do not yet exist. Those permissions are then used if the file (or other object) is created at a later time. In other words, RACF establishes security policies rather than just permission records.

RACF initially identifies and authenticates users via a user ID and password when they log on to the system. When a user tries to access a resource, RACF checks its database. Then, based upon the information that it finds in the database, RACF either allows or denies the access request. It displays an ICH408I message if the access is denied.

To understand the basic concepts, we examine the ICH408I access denial message, shown in Example 1-1, to see what RACF is saying.

Example 1-1 ICH408I message

```
ICH408I USER(UTSM) GROUP(MTSM) NAME(TSOMON STC-USERID)
 EZB.PORTACCESS.SX00.TCP2.SAPSYS CL (SERVAUTH)
 INSUFFICIENT ACCESS AUTHORITY FROM EZB.PORTACCESS.*.*.SAPSYS (G)
```

This message means that the user is not authorized to access a resource defined as a TCP/IP port.

- The user ID is UTSM.
- The user ID belongs to RACF group MTSM.

  RACF keeps users with similar security access requirements in groups so that any access changes can be done simply to the group profile (record) rather than to each individual user's profile.
- The name recorded in the RACF database for the user ID is TSOMON STC-USERID.
- The TCP/IP port that failed access has a name SAPSYS, and it belongs to the TCP/IP stack named TCP2 on z/OS system SX00.
- The TCP/IP port belongs to the resource class SERVAUTH. The resource name that we use to query RACF is `EZB.PORTACCESS.SX00.TCP2.SAPSYS`.

  In other words, this TCP/IP port is defined to RACF as `EZB.PORTACCESS.SX00.TCP2.SAPSYS`. What we do not know yet is which port number, and which protocol, is really represented by this resource.

When the user ID UTSM tried to open the port named SAPSYS on the system SX00 and on TCP/IP stack TCP2, RACF checked its database for a discrete profile specific to `EZB.PORTACCESS.SX00.TCP2.SAPSYS`.

It could not find it, but instead found a generic profile `EZB.PORTACCESS.*.*.SAPSYS`, which covered the resource. (A generic profile protects multiple resources having similar characteristics.) The user ID UTSM was not in the access list and RACF failed the request.

Next, using the information in Example 1-2, we describe what should have been done to protect the TCP/IP port and to give proper access to the legitimate user.

Example 1-2 RACF commands to protect a TCP/IP port

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
RDEFINE EZB.PORTACCESS.*.*.SAPSYS UACC(NONE)
PERMIT EZB.PORTACCESS.*.*.SAPSYS CLASS(SERVAUTH) ID(UTSM) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

We examine each TSO command shown in Example 1-2, explaining why it is needed and what it does. (You need to have RACF authority to issue these commands.)

- SETROPTS CLASSACT(SERVAUTH) activates the SERVAUTH class of profiles that protect resources managed by the z/OS TCP/IP stack.

  When you activate a resource class, you are telling RACF to do authorization checking whenever an application (on behalf of a user ID) tries to access any resource protected under that class. RACF keeps resources with similar characteristics in one class. TCP/IP resources such as the stack, networks, and ports belong to the SERVAUTH class.

  Another example of a resource class is OPERCMDS, which protects the use of sensitive operator commands such as VARY TCPIP.

  Note: In most installations, the SERVAUTH class will be active. In that case, you can skip this step. You can issue a RACF command SETROPTS LIST in TSO to check if it is active. Look in the section starting with ACTIVE CLASSES =.

- SETROPTS RACLIST(SERVAUTH) tells RACF to read the profiles for the SERVAUTH class from the RACF database into the RACF data space, and to activate the sharing of these in-storage profiles.

  With these profiles in storage, RACF does not have to perform an input/output (I/O) to read the RACF database when making an access decision, and this improves performance.

- RDEFINE EZB.PORTACCESS.*.*.SAPSYS UACC(NONE) defines a generic profile to cover all TCP/IP ports that have the name SAPSYS.

  The profile that you have to define to protect the TCP/IP ports is of the format `EZB.PORTACCESS.systemname.stackname.portname`.

  - The first two qualifiers of the profile have to be `EZB.PORTACCESS`. This tells the system that this profile is protecting TCP/IP ports.
  - The third qualifier specifies the z/OS system name.
  - The fourth qualifier specifies the name of the TCP/IP stack.
  - The last qualifier is the name of the port. We have the wildcard character (*) for systemname and stackname. This will cover TCP/IP ports with name SAPSYS on all TCP/IP stacks and on all z/OS systems. Note that we have set `UACC(NONE)` in order to restrict its access.

- PERMIT EZB.PORTACCESS.*.*.SAPSYS CLASS(SERVAUTH) ID(UTSM) ACCESS(READ) gives READ access for the user ID UTSM to the TCP/IP port.

- SETROPTS RACLIST(SERVAUTH) REFRESH updates the in-storage SERVAUTH class profiles in the RACF data space.

Example 1-3 shows another example. The socket option IPV6_NEXTHOP is sensitive and you want to restrict its usage to authorized persons.

Example 1-3 RACF commands to restrict the use of socket option IPV6_NEXTHOP

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
RDEFINE SERVAUTH EZB.SOCKOPT.*.*.IPV6_NEXTHOP UACC(NONE)
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(UTSM) ACCESS(READ)
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Example 1-3 shows the RACF commands you will need.

- RDEFINE SERVAUTH EZB.SOCKOPT.*.*.IPV6_NEXTHOP UACC(NONE) defines the RACF profile to restrict the use of the IPV6_NEXTHOP socket option.
- PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(UTSM) ACCESS(READ) gives access for the user UTSM to use the IPV6_NEXTHOP socket option in the user's programs.

You can also protect the use of the socket option in another way. You can say that any user can use the socket option, if the user is doing it via a specific program. That way you are giving authority to a program rather than to a user to access the resource (socket option).

- PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ) allows anyone to access the socket option, provided the user is using program TSOMON.

In the following section, we show the various network resources protected by RACF.

1.2 Protecting your network resources

You have seen how you can define resource profiles to protect a TCP/IP port, and also to protect the use of an IPv6 socket option. All network resources are protected by RACF in the same way. Most TCP/IP resources are protected by profiles defined in the SERVAUTH resource class.

Tip: All z/OS Communications Server profiles in the SERVAUTH class have EZA, EZB, or IST as the high level qualifier (HLQ).

Use this method to protect a resource:

- If the SERVAUTH class is not active, activate it with the **SETROPTS** command. You need to perform this only once in your system (in most cases, it will already be active on your system).
- Identify the profile that protects the resource from Chapter 2, "Protecting network resources" on page 11. Define the profile with the RDEFINE command.
- Allow access to authorized users to this profile using the PERMIT command.
- Refresh the RACLIST in-storage profiles of the SERVAUTH class in the RACF data space using the SETROPTS command.

Refer to Chapter 2, "Protecting network resources" on page 11, for more detailed information about how RACF protects the various TCP/IP resources using this method.

1.3 Protecting your programs

One of the main strengths of the z/OS platform is the exceptionally robust protection of its programs from unauthorized alteration. This is one of RACF's most powerful features, and it makes the z/OS platform effectively immune to computer viruses. Very strict controls and protection mechanisms in RACF make it almost impossible for any unauthorized person to modify programs on the z/OS platform.

z/OS security has evolved and matured over a period of more than quarter of a century. Many other operating system platforms cannot match the inherent security of the z/OS platform, because they were originally designed either with a single user in mind or for academic collaboration.

RACF uses the following mechanisms to secure programs from unauthorized access:

- Authorized Program Facility (APF)
- Program Protection by RACF resource class PROGRAM
- Program Access Control
- Controlling program access by SYSID
- The sticky bit in the UNIX® environment

In the following sections, we discuss each mechanism in more detail.

Authorized Program Facility

z/OS protects the use of sensitive system functions and supervisor calls (SVC) using the Authorized Program Facility (APF). Programs have to be APF-authorized in order to use these system functions. To obtain APF authorization, a program should meet two conditions:

- It must reside in a library that is in the APF list or in the Link Pack Area (LPA).
- The program must be link-edited with authorization code AC=1.

In addition, the program libraries are protected by RACF. These protections make virus attacks very unlikely on z/OS.

Program protection by RACF resource class PROGRAM

RACF treats program load modules as protected resources. PROGRAM is the RACF resource class that protects programs. Example 1-4 shows the RACF commands to protect a program. You use the ADDMEM parameter in RDEFINE to specify the library where the program resides.

Example 1-4 RACF command to protect a program

```
RDEFINE PROGRAM MYPROGRAM ADDMEM('SYS1.LINKLIB') UACC(NONE)
PERMIT MYPROGRAM CLASS(PROGRAM) ID(SOMEUSER) ACCESS(READ)
```

Program Access Control

You can use the Program Access Control facility to specify that access to a resource is allowed only if you are accessing it using a specific program. The program itself has to be in a controlled library and restricted to only authorized users.

In Example 1-5, we show how to restrict the use of advanced IPV6 socket options by program access control.

Example 1-5 PADS to protect use of sockect option

```
PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(*) WHEN(PROGRAM(TSOMON)) ACCESS(READ)
```

Controlling program access by SYSID

Access to programs (load modules) can be controlled based on the SMF system ID of the z/OS system, as shown in Example 1-6.

Example 1-6 Controlling program access by system ID

```
PERMIT MYPROGRAM CLASS(PROGRAM) ID(SOMEUSER) WHEN(SYSID(PROD_SYSTEM))
```

The sticky bit in the z/OS UNIX environment

Because z/OS UNIX files are not as secure as MVS data sets, sensitive programs running under z/OS UNIX do not load from the z/OS UNIX file system. z/OS will instead turn to the standard MVS search order to look for a copy of the executable file in an MVS load library. z/OS UNIX System Services uses the sticky bit on the program library to bypass loading of a program from the UNIX Systems Services file system. Often the program needs to reside in APF authorized libraries protected by program control.

The “sticky bit” is one of the bits in the Access Control List (ACL) of the z/OS UNIX file. To determine whether the sticky bit is set on a file (program), issue the UNIX command **ls -l**, as shown in Example 1-7. The T as the last character in the access list for the file IMWCGIBN indicates that its sticky bit is on. This means the system will not look for the program IMWCGIBN in the UNIX files; instead, it will search for it in more secure authorized z/OS libraries.

Example 1-7 UNIX command to show the sticky bit

```
/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: >
```

1.4 Associating a user ID with a started task

RACF makes sure that everyone who accesses the system resources is accountable. This applies to the system tasks as well. For this, RACF associates every started task (STC) with a specific user ID. RACF keeps this information in a resource class called STARTED. Example 1-8 shows you how to define this to RACF.

Example 1-8 RACF commands to associate a user ID with a started task

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

Before you can start an STC in the system, you must tell RACF to give the STC user ID access to all the resources used by the STC, by using the PERMIT commands.

1.5 Setting up security for daemons in z/OS UNIX

TCP/IP and other, related daemons work in the z/OS UNIX environment and use many of its services. Therefore, it is important to understand how to set up security for daemons working in the z/OS UNIX security environment.

To set up a daemon under z/OS UNIX, follow these steps:

1. Define a user ID for the daemon.
2. Define an OMVS segment for the user ID.
3. Give superuser authority for the user ID.
4. Give user ID access to various RACF profiles protecting the resources for which the daemon will need access.
5. Associate the user ID with the daemon.
6. For some daemons, you have to turn the sticky bit on to indicate that the program module resides in a protected z/OS library, rather than in the z/OS UNIX file pointed to by the module.

For more detail, refer to *z/OS UNIX System Services User's Guide*, SA22-7801.

1.6 RACF multilevel security for network resources

Multilevel security (MLS) addresses government requirements for highly secure data. This supports sharing of classified information among multiple agencies on demand. As security controls become more critical in emerging on demand virtual environments, this new technology has applications in the general business sectors, as well. This secondary layer is on the top of existing RACF resource protection.

1.6.1 Basic MLS concepts

In MLS, the resources are divided into a number of categories based on where they belong. For example, you can classify the resources of your organization based on departments such as PAYROLL, PERSONNEL, RESEARCH, MARKETING, SALES, PRODUCTION, and so on.

Resources in each category are further classified based on their importance and sensitivity. For example, you can classify them into GENERAL, CONFIDENTIAL, SENSITIVE, and TOP-SECRET in the ascending order of their importance and sensitivity. This classification is hierarchical, which means GENERAL would be the lowest that everyone can access. The level goes up with CONFIDENTIAL, then SENSITIVE, and the highest level is TOP-SECRET.

After this classification is done, assign a similar category and security level for each user by default. After you switch on MLS, when a user tries to access a resource, RACF will check whether the user's security level is equal to or above that of the resource. RACF will also verify that the user and the resource belong to the same category; thus, a user in the PERSONNEL department will be able to access a resource only in the PERSONNEL department.

Also, the user should have the right security level. For example, a user from PERSONNEL with a security level of GENERAL will not be able to access a PERSONNEL resource with a security level of CONFIDENTIAL. A user from MARKETING will not be able to access a resource from RESEARCH, though the user may have TOP-SECRET security level in the MARKETING department.

RACF uses security labels (SECLABELs) to enforce multilevel security.

SECLABELS

A SECLABEL, or security label, consists of two entities:

- A security *category* such as PAYROLL, PERSONNEL, or RESEARCH
- A security *level* such as CONFIDENTIAL, SENSITIVE, or TOP-SECRET

The security administrator sets security labels for each user and each resource. When a user tries to access a resource, RACF allows access only if the security level in the user's SECLABEL is higher or equal to the security level specified in the resource's SECLABEL for the security category being accessed.

A user might be permitted to access several security labels, but can only be logged onto one of them at a time.

You can provide additional layers of protection for your network resources by implementing MLS. For more details, refer to section 4.1 of *z/OS 1.6 Security Services Update*, SG24-6448.

1.7 Digital certificates in RACF

RACF allows you to create and maintain security keys, key rings, and digital certificates in the RACF database. In a client/server environment, RACF has the ability to map a client's digital certificate to a RACF user ID by either storing the digital certificate in the RACF database, or by mapping using a certificate name filter rule. A digital certificate or digital ID, issued by a Certificate Authority, contains information that uniquely identifies the client.

See Chapter 9, "IP Security" on page 373, for information about how to set up digital certificate keys and key rings.

1.8 Further information

You can find samples of jobs with the RACF commands required for z/OS Communications Server and applications in your installation library TCPIP.SEZAINST(EZARACF). (Note that the high level qualifier of this library could be different in your installation.)

2

Protecting network resources

This chapter discusses the RACF security profiles that can be used to protect access to various network resources.

The following topics are discussed in this chapter.

Section	Topic
2.1, “The SERVAUTH resource class” on page 12	Basic setup using the RACF SERVAUTH class to protect your resources
2.2, “Protecting your TCP/IP stack” on page 12	How the TCP/IP resource is protected by RACF
2.3, “Protecting your network access” on page 13	How to protect your network
2.4, “Protecting your network ports” on page 16	How RACF protects your ports
2.5, “Protecting the use of socket options” on page 18	How to restrict the use of sensitive socket options
2.6, “Protecting sensitive network commands” on page 19	How to set up security for your sensitive network commands to prevent unauthorized use
2.7, “Protecting FTP-related resources” on page 26	The setup to protect FTP resources
2.8, “Protecting network management resources” on page 27	How to use RACF to protect network management resources (such as data collection agents)
2.9, “Protecting miscellaneous resources” on page 27	RACF support to protect miscellaneous network resources

2.1 The SERVAUTH resource class

Most network resources are protected by the SERVAUTH resource class profiles. To protect a resource, perform the following tasks:

- If the SERVAUTH class is not active, you need to activate it using the SETROPTS command. You need to do this only once in your system and in most cases this would already be active on your system.
- Identify the profile that protects the resource. This chapter lists security profiles that can be used to protect your resources. Define the profile using the RACDEF command.
- Allow access to authorized users to this profile using the PERMIT command.
- Refresh the RACLIST in-storage profiles of the SERVAUTH class in the RACF data space using the SETROPTS command.

In the following sections, we describe how to set up the RACF profiles to protect various network resources.

2.2 Protecting your TCP/IP stack

This section discusses the Security Access Facility (SAF) security profiles that can be used to protect access to a TCP/IP stack's resources.

2.2.1 Stack access overview

Stack access control provides a way to permit or deny users or groups of users access to a TCP/IP stack. The function controls the ability of a user to open an IP socket with the socket() API function. Stack access control is implemented by defining a SERVAUTH class RACF profile. The profile name is in the format:

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

Where:

- `EZB.STACKACCESS` is constant.
- *`sysname`* is the name of the z/OS image (&SYSNAME symbolic).
- *`tcpipname`* is the job name of the TCP/IP stack.

2.2.2 Example setup

Here we explain how to control access to the TCP/IP stack running with a job name of TCPIPD on the z/OS system SC30. The first thing to do is to add the SERVAUTH STACKACCESS profile to RACF.

As mentioned previously, the format of the profile name is EZB.STACKACCESS.*sysname.tcpname*. Therefore, in our example, profile name will be EZB.STACKACCESS.SC30.TCPIPD, as shown in Example 2-1.

Example 2-1 RACF SERVAUTH profile to protect TCP/IP stack access

```
RDEFINE EZB.STACKACCESS.SC30.TCPIPD UACC(NONE)
PERMIT EZB.STACKACCESS.SC30.TCPIPD CLASS(SERVAUTH) ID(UTSM) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

We specified that Universal Access (UACC) is none, meaning that the default for any user is to be denied access. Then we gave access to user UTSM using the RACF PERMIT

command. After the profile is added, we refresh the in-storage RACF profiles with a **setropts raclist(servauth) refresh** command.

This explains how to enable a user to access the TCP/IP stack. This concept of a “user” applies equally to the owner of any server running on the stack (for example, the FTP started task user ID has to be given access to the stack’s RACF profile).

2.3 Protecting your network access

Network access control enables system administrators to represent access to an IP network, subnetwork, or host as a RACF resource. The ability to send IP packets to those networks, subnetworks, or hosts can then be permitted or denied at a RACF user or group level.

This feature provides an additional layer of security to any authentication or authorization that is used at the target system. It might be used, for example, to prevent access to the Internet by anyone except the SMTP server, or it could be used to stop general users attempting to telnet to a server that contained payroll information.

2.3.1 Network access control overview

In the TCP/IP profile, there is a parameter block, NETACCESS/ENDNETACCESS. This is where you specify the mapping of an IP network, subnetwork, or host to an SAF profile. Example 2-2 shows a sample NETACCESS block.

Example 2-2 Sample NETACCESS block

```
NETACCESS
  10.40.2.0/24       MYSUBNET    ;my workstation subnet
  10.40.2.119/32     MYPC        ;my workstation
  DEFAULT 0          WORLD       ;everything else
ENDNETACCESS
```

In this example, access to hosts on subnet 10.40.2 is mapped to RACF profile MYSUBNET, access to host 10.40.2.119 is mapped to SAF profile MYPC, and access to any other host is mapped to SAF profile WORLD. These RACF profiles are defined to RACF in the SERVAUTH class. The profile name to be defined is in the following format:

`EZB.NETACCESS.`*`sysname`*`.`*`tcpipname`*

Where:

- `EZB.NETACCESS` is constant.
- *`sysname`* is the name of the z/OS image (&SYSNAME symbol).
- *`tcpipname`* is the job name of the TCP/IP stack.
- *`resourcename`* is the name specified in the NETACCESS block.

Note: On the NETACCESS statement, there are two ways to specify the subnet mask.

- The first way is to use the traditional decimal notation, such as 255.255.255.0.
- The second way is to use a number, up to 32, that specifies the number of bits, left to right, that should be used as a subnet mask if the mask is expressed in binary.

  For example, the subnet mask 255.255.255.0 expressed in binary is 11111111.11111111.11111111.00000000. Note that there are 24 bits set on.

  To specify this particular mask on a NETACCESS statement, you could use either of the following two ways, using the IP address 192.168.100.0 as an example:

  NETACCESS 192.168.100.0 255.255.255.0

  Or:

  NETACCESS 192.168.100.0/24

The system that we used in Example 2-2 on page 13 was on a z/OS image named SC30 with a TCP/IP stack name of TCPIPD. Therefore, the three profiles that need to be defined are:

- EZB.NETACCESS.SC30.TCPIPD.MYSUBNET
- EZB.NETACCESS.SC30.TCPIPD.MYPC
- EZB.NETACCESS.SC30.TCPIPD.WORLD

If you define these profiles to RACF with UACC(NONE), then users must be specifically permitted access to these profiles in order to send IP packets to the addresses represented by the profiles.

If a user is attempting to send an IP packet to a host that is not covered by any network or mask entry in the NETACCESS block, then access is automatically allowed. However, if a DEFAULT statement is present, then access is granted or denied based on the user's access to the SAF profile mapped by the DEFAULT statement.

2.3.2 Server considerations

End users are not the only users of TCP/IP to be affected by NETACCESS control. Any IP applications (servers) that run under their own userids would need access to the RACF profiles of the desired networks, if these networks are protected by a NETACCESS statement. For example, a server such as the FTP daemon would need to be permitted access to any hosts or subnets that an FTP transfer will be performed with.

There is another consideration: If you have a user in the network who wants to FTP to or from your FTP server, then the user ID that this user logs on with must have been permitted to NETACCESS if the user's own IP address or network is protected.

2.3.3 Using NETSTAT for network access control

The console command **`D TCPIP,stackname,Netstat,ACCess,NETWork`** shows how IP addresses and masks are mapped to SAF profiles. See Figure 2-1 on page 15 for the NETSTAT console command to display the network access control for the test TCPIPD system.

```
D TCPIP,TCPIPD,NETSTAT,ACCESS,NETWORK

NETWORK ACCESS INFORMATION
INBOUND: NO   OUTBOUND: YES
SAF NAME  NETWORK PREFIX AND PREFIX LENGTH
--------  ------------------------------
WORLD     DEFAULT
  PRFNM: <NONE>                                  SECLABEL: <NONE>
MYSUBNET  10.40.2.0/24
  PRFNM: <NONE>                                  SECLABEL: <NONE>
MYPC      10.40.2.119/32
  PRFNM: <NONE>                                  SECLABEL: <NONE>
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

Figure 2-1 NETSTAT command to display network access control in our test system

Note that the TSO NETSTAT command does not display this information.

2.3.4 Working example of network access control

We implement network access control on the TCP/IP stack discussed in 2.3.1, “Network access control overview” on page 13. In that section, we show the three SAF profile names that need to be defined for the given NETACCESS block. Example 2-2 on page 13 shows the configuration.

When the RACF profile names shown in 2.3.1, “Network access control overview” on page 13, have been defined to RACF, we issue the TSO command PING 10.40.2.119 to send a packet to workstation 10.40.2.119. Because we have not been permitted access to the 10.40.2.119 host (profile MYPC), we would expect an error. Figure 2-2 shows the error message from RACF, indicating that access to the MYPC profile was not permitted.

```
 ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TESTTEST          )
   EZB.NETACCESS.SC30.TCPIPD.MYPC CL(SERVAUTH)
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
 CS V1R9: Pinging host 10.40.2.119
 sendto(): EDC5111I Permission denied.
```

Figure 2-2 RACF error while attempting PING to host covered by host profile

Note: Even though both the host 10.40.2.119 and its subnet 10.40.2.0 are protected by RACF profiles, the RACF check is only performed on the *most specific* network or host entry. That is, the entries in the NETACCESS block are checked starting with those with the most bits specified in the subnet mask first, until a match is found.

If we now attempt to ping a new host 10.40.2.120, we would expect a RACF error when the TCP/IP address space does a RACF check for access to the MYSUBNET profile, as this is the most specific entry for that host address. We attempt to ping the IP address 10.40.2.120 and we receive the error as expected; see Figure 2-3 on page 16.

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETACCESS.SC30.TCPIPD.MYSUBNET CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS V1R9: Pinging host 10.40.2.120
sendto(): EDC5111I Permission denied.
```

Figure 2-3 RACF error while attempting to ping a host covered by subnet profile

Finally, we show an example of the error you would receive when attempting network access to a host not specified on any NETACCESS statements. This assumes that you have coded a DEFAULT statement in the NETACCESS block. If you do not code this statement, access permission will not be checked for any host not covered by any other NETACCESS statement.

We tried to ping 10.40.5.10, an IP address that is not specifically stated in a host or subnet entry. The DEFAULT NETACCESS statement is used for the SAF check that is mapped to profile WORLD. As shown in Figure 2-4, we received a RACF error message indicating that we did not have access to EZB.NETACCESS.SC63.TCPIPC.WORLD.

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETACCESS.SC30.TCPIPD.WORLD CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS V1R9: Pinging host 10.40.5.10
sendto(): EDC5111I Permission denied.
```

Figure 2-4 RACF error while attempting to ping a host covered by the DEFAULT statement

2.4 Protecting your network ports

The ability of a server to bind to a specific port can be controlled in a number of ways using the UDPCONFIG, TCPCONFIG, and PORT (or PORTRANGE) TCP/IP profile statements.

The use of TCPCONFIG RESTRICTLOWPORTS and UDPCONFIG RESTRICTLOWPORTS is encouraged to enhance security. If these statements are present, low ports (< 1024) can only be bound when at least one of the following is true:

- The bind is issued from a process with a UNIX superuser (UID 0).
- The bind is issued from an APF-authorized application.
- The port is reserved for the application by job name, which may include *, OMVS, or a TSO user ID.
- If an SAF resource name is used, the user ID of the binding process must be permitted to the resource by the security product; refer to 2.4.1, “The PORT/PORTRANGE SAF keyword” on page 17.
- 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.
- Specifying a job name on a PORT or PORTRANGE statement restricts the use of that port (and protocol) to the specified job name. Multiple PORT statements can be specified for a TCP port but not for UDP. Note that a job name of * (the wildcard character) is normally used with the SAF keyword, which is described next.

- Specifying the SAF keyword and profile name provides a mapping from a port and protocol to an SAF SERVAUTH class profile. A server attempting to bind to this port and protocol is checked for SAF access to the profile named. The use of the SAF keyword is covered in this section.

2.4.1 The PORT/PORTRANGE SAF keyword

The SAF keyword can be specified along with all other valid options on the PORT and PORTRANGE statements. The special job name wildcard * is normally used with the SAF keyword so that access to the port is completely handled by the server's SAF profile rather than job name. Of course, a specific job name and the keyword SAF can still be coded together if you would like to secure not only the job name that can bind a socket, but also the SAF user associated with the job.

To illustrate the concept, we show how to protect the FTP ports here. Sample PORT statements for the system SC30 are shown in Figure 2-5.

```
PORT
    20 TCP *  NOAUTOLOG SAF FTP20 ; FTP Server
    21 TCP OMVS  BIND 10.40.1.230 SAF FTP21 ;control port
    23 TCP INTCLIEN           ; MVS Telnet Server
    512 UDP RESERVED          ; Shut down port 512
    23 TCP OMVS BIND 10.40.1.230; OE Telnet Server
   500 UDP IKED               ; IKE Daemon
  4500 UDP IKED               ; IKE Daemon
```

Figure 2-5 PORT statements for stack TCPIPC

Given the PORT statements in Figure 2-5, UDP port 512 is reserved, and therefore completely unavailable for use. Any process attempting to use TCP ports 20 or 21 have to be SAF-authorized. The following SERVAUTH profiles have to be defined (assuming the system name is SC30 and the TCP/IP stack name is TCPIPD):

- EZB.PORTACCESS.SC30.TCPIPD.FTP20
- EZB.PORTACCESS.SC30.TCPIPD.FTP21

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

With FTP ports 20 or 21 (or any well-known port usually used for a system-type server), there is a possible security exposure when permitting port use by the SAF keyword. When a user is permitted to bind to, for example, port 20, the user can bind to that port using any program, not just through the FTP server. To prevent this, we recommend that you do *not* code an SAF keyword for port 20, but instead use the RESTRICTLOWPORTS parameter of the TCPCONFIG statement in conjunction with specifying “OMVS” as the job name for port 20. This restricts the use of port 20 to APF-authorized programs, UNIX superusers, or the FTP server.

The FTP daemon is the only user that needs to access the FTP control port 21, and hence should have access to the RACF SERVAUTH profile EZB.PORTACCESS.SC30.TCPIPD.FTP21. If the FTP server does not have access to the profile, you will get a RACF error similar to that shown in Figure 2-6 on page 18.

```
S FTPDD
$HASP100 FTPDD    ON STCINRDR
IEF695I START FTPDD    WITH JOBNAME FTPDD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 FTPDD    STARTED
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(####################) 400
  EZB.PORTACCESS.SC30.TCPIPD.FTP21 CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
+EZY2714I FTP server shutdown in progress
+EZYFT59I FTP shutdown complete.
$HASP395 FTPDD    ENDED
```

Figure 2-6 FTP server unauthorized to use port 21

The FTP data connection port (20) is bound under the identity of the user, not the FTP daemon. Therefore, if the PORT statement for port 20 is configured as shown in Figure 2-5 on page 17, then all end users (including the default user, if defined) who can potentially perform FTP need to be permitted to the port 20 RACF SERVAUTH profile EZB.PORTACCESS.SC30.TCPIPD.FTP20.

2.4.2 Using NETSTAT to display Port Access control

The TCPIPC stack contains the PORT statement shown in Figure 2-5 on page 17. The console command **D TCPIP,stackname,Netstat,PORTlist** in Figure 2-7 shows the configuration of the ports, and whether an SAF profile is associated with a port (the **F** in the FLAGS column).

```
D TCPIP,TCPIPD,NETSTAT,PORTLIST

PORT# PROT USER     FLAGS    RANGE       SAF NAME
00020 TCP  *        DF                   FTP20
00021 TCP  OMVS     DABFU                FTP21
      BINDSPECIFIC: 10.40.1.230
00023 TCP  OMVS     DABU
      BINDSPECIFIC: 10.40.1.230
00023 TCP  TCPIPD   DAU
00500 UDP  IKED     DA
00512 UDP  RESERVED DA
04500 UDP  IKED     DA
7 OF 7 RECORDS DISPLAYED
END OF THE REPORT
```

Figure 2-7 NETSTAT PORTLIST console command display for TCPIPD

2.5 Protecting the use of socket options

You can use RACF profiles to prevent the misuse of the sensitive SO_BROADCAST and IPv6 API socket options.

2.5.1 SO_BROADCAST socket option access control

The SO_BROADCAST option provides control over the broadcast function, which can be prone to misuse if not restricted.

RACF profile EZB.SOCKOPT.*sysname.tcpname*.SO_BROADCAST controls this option.

Where:

- *sysname* is the z/OS system name. Our system name was SC30.
- *tcpname* is the jobname of the TCP/IP stack. Our stack was TCPIPD.

We show sample RACF commands to define this resource and then give access to OMPROUTE, which needs to use the SO_BRODCAST socket option; see Example 2-3.

Example 2-3 Sample RACF commands to protect SO_BROADCAST socket option

```
RDEFINE SERVAUTH EZB.SOCKOPT.SC30.TCPIPD.SO_BROADCAST UACC(NONE)
PERMIT EZB.SOCKOPT.SC30.TCPIPD.SO_BROADCAST CLASS(SERVAUTH) ACCESS(READ) ID(OMPROUT)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

If desired, you can protect this option on all your systems and on all the stacks by using a single generic profile. The profile that you can use is:

```
EZB.SOCKOPT.*.*.SO_BROADCAST
```

2.5.2 IPv6 advanced socket API options

The z/OS Communications Server has a number of advanced socket API options that need to be restricted to only authorized users. There is one SERVAUTH class profile to protect each of these socket options. These profiles are shown in the following list.

- IPV6_NEXTHOP - EZB.SOCKOPT.*sysname.tcpname*.IPV6_NEXTHOP
- IPV6_TCLASS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_TCLASS
- IPV6_RTHDR - EZB.SOCKOPT.*sysname.tcpname*.IPV6_RTHDR
- IPV6_HOPOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_HOPOPTS
- IPV6_DSTOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_DSTOPTS
- IPV6_RTHDRDSTOPTS - EZB.SOCKOPT.*sysname.tcpname*.IPV6_RTHDRDSTOPTS
- IPV6_PKTINFO - EZB.SOCKOPT.*sysname.tcpname*.IPV6_PKTINFO
- IPV6_HOPLIMIT - EZB.SOCKOPT.*sysname.tcpname*.IPV6_HOPLIMIT

Note that the last qualifier of the profile is the same as the socket option itself.

2.6 Protecting sensitive network commands

This section discusses the ways to control the use of the TCP/IP system administration commands. These commands are categorized by where they originate.

From the z/OS console, you can use the DISPLAY and VARY commands for the TCP/IP address spaces. The VARY TCPIP command can be used to stop and start TCP/IP interfaces, reload configuration parameters, start and stop traces, drop TCP connections, and quiesce the TN3270 server. This command is very powerful and should only be authorized to operators or system administrators.

From TSO, there is the NETSTAT command, and from the UNIX shell there is the **onetstat** command. Both of these commands are used primarily to display information about the local TCP/IP environment. You may want to restrict these commands so that people are unable to obtain information about your TCP/IP configuration, perhaps in preparation for an attack of some kind.

2.6.1 z/OS VARY TCPIP command security

This section describes the mechanisms by which you can limit users to the VARY TCPIP commands.

RACF profile details

The z/OS console VARY TCPIP commands are protected with RACF profiles defined in the resource class OPERCMDS. You can define a single profile to represent all VARY TCPIP commands, or you can specify individual profiles for each VARY TCPIP command option.

The format of the profile name is:

`MVS.VARY.TCPIP.`*`command`*

Where:

- MVS.VARY.TCPIP is a constant.
- *command* is either a double asterisk (**), meaning all command options, or a specific VARY TCPIP option name, such as OBEYFILE.

Important: The profile name does not specify a z/OS image name or TCP/IP stack name. Keep in mind, therefore, that if there is more than one stack on your z/OS image or your SAF database is shared between multiple z/OS systems, granting access to a command enables that command to be performed by a user against any TCP/IP stack in any z/OS system that shares the database.

Protecting VARY TCPIP at the command level

To specify protection at the command level (any option), specify the generic OPERCMDS profile with a profile name of MVS.VARY.TCPIP.**. Figure 2-8 shows how this is done using RACF commands.

```
SETROPTS GENERIC(OPERCMDS)
RDEFINE OPERCMDS (MVS.VARY.TCPIP.**) UACC(NONE)
SETROPTS GENERIC(OPERCMDS) REFRESH
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure 2-8 Defining generic VARY TCPIP profile to protect command with all options

Note: If the MVS.VARY.TCPIP.** profile is defined, any user who needs to use any VARY TCPIP command must have CONTROL access to this profile, *regardless* of whether they have access to other MVS.VARY.TCPIP.command profiles.

Protecting VARY TCPIP at the command option level

You may decide to protect the VARY TCPIP command at a more granular level so that you can control who has authority to use the options of the VARY TCPIP command. To protect the command options, define the particular VARY TCPIP option that you want to protect as the last qualifier in the profile name.

Exceptions to this rule are the VARY TCPIP START and VARY TCPIP STOP commands, which are protected together with the profile named MVS.VARY.TCPIP.STRTSTOP.

Refer to Table 2-1 on page 21 for a list of RACF profile names to protect the VARY TCPIP command options.

Table 2-1 List of RACF profiles to protect various VARY TCPIP console commands

RACF profile name	Command protected
MVS.VARY.TCPIP.**	All VARY TCPIP options
MVS.VARY.TCPIP.DROP	VARY TCPIP,,DROP
MVS.VARY.TCPIP.OBEYFILE	VARY TCPIP,,OBEYFILE
MVS.VARY.TCPIP.PKTTRACE	VARY TCPIP,,PKTTRACE
MVS.VARY.TCPIP.STRTSTOP	VARY TCPIP,,START or VARY TCPIP,,STOP

Figure 2-9 shows the VARY TCPIP,,STOP and VARY TCPIP,,START commands being protected.

```
RDEFINE OPERCMDS MVS.VARY.TCPIP.STRTSTOP UACC(NONE)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure 2-9 Defining specific VARY TCPIP profile to protect VARY TCPIP,,STOP/START commands

VARY TCPIP command security scenario

The command V TCPIP,TCPIPC,STOP,OSA22E0 was chosen to test the console RACF security profiles. The command output is shown in Figure 2-10 before any RACF security profiles were defined to the system. The SAF profile that controls the V TCPIP,,STOP command (in addition to the generic MVS.VARY.TCPIP.**, if defined) is MVS.VARY.TCPIP.STRTSTOP.

```
V TCPIP,TCPIPC,STOP,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STOP,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2080
```

Figure 2-10 VARY TCPIP,TCPIPC,STOP command output: No RACF profiles defined yet

For the first test, we only defined the generic MVS.VARY.TCPIP.** profile to protect all options of the V TCPIP command.

For the second test, we additionally defined the MVS.VARY.TCPIP.STRTSTOP profile to protect the V TCPIP,,STOP command. Both times, unauthorized use of the command caused the expected RACF error.

Defining the generic profile to protect all V TCPIP commands

The generic profile MVS.VARY.TCPIP.** was added to the OPERCMDS class with UACC(NONE), as shown in Figure 2-8 on page 20. At this stage, no user had any access to this profile. A user then attempted a V TCPIP,TCPIPC,START,OSA2080 command from the console, which resulted in the RACF error shown in Figure 2-11 on page 22. Note that the profile being SAF checked was MVS.VARY.TCPIP.**

```
V TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 689
  MVS.VARY.TCPIP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  FROM MVS.VARY.TCPIP.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

Figure 2-11 RACF error for VARY TCPIP STOP command showing generic profile violation

Defining the specific profile to protect the V TCPIP,,START command

The specific profile MVS.VARY.TCPIP.STRTSTOP was added to the OPERCMDS class with UACC(NONE), as shown in Figure 2-9 on page 21. At this stage, both the MVS.VARY.TCPIP.STRTSTOP and the generic MVS.VARY.TCPIP.** profiles were defined, and the user did not have access to either of them.

After the V TCPIP,TCPIPC,STOP,OSA2080 command was issued, Figure 2-12 shows that the profile name that is causing the access problems is MVS.VARY.TCPIP.**, even though the MVS.VARY.TCPIP.STRTSTOP profile is defined. This confirms the statement in “Protecting VARY TCPIP at the command level” on page 20 that the generic profile is always checked first, if defined.

```
V TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 699
  MVS.VARY.TCPIP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  FROM MVS.VARY.TCPIP.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

Figure 2-12 RACF error for VARY TCPIP,,STOP command shows generic profile name

If the generic profile MVS.VARY.TCPIP.** is defined, any user who wants to enter a VARY TCPIP command must have CONTROL access to it. Example 2-4 shows the RACF commands to give such access to user CS09.

*Example 2-4 Giving CONTROL access to MVS.VARY.TCPIP.***

```
PE MVS.VARY.TCPIP.** CLASS(OPERCMDS) ID(CS09) ACCESS(CONTROL)
SETROPTS RACLIST(OPERCMDS) REFRESH
SETROPTS GENERIC(OPERCMDS) REFRESH
```

Now that we had CONTROL access to the generic profile MVS.VARY.TCPIP.**, the SAF check is for the profile that controls the VARY TCPIP,,STOP command, which is V TCPIP,TCPIPC,START,OSA2080. Figure 2-13 on page 23 shows the RACF error resulting when the user does not have CONTROL access to the specific profile.

```
V TCPIP,TCPIPC,START,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,START,OSA2080
IEE345I VARY     AUTHORITY INVALID, FAILED BY SECURITY PRODUCT
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          ) 712
  MVS.VARY.TCPIP.STRTSTOP CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(CONTROL)  ACCESS ALLOWED(NONE   )
EZZ0059I MVS.VARY.TCPIP.STRTSTOP COMMAND FAILED: NOT AUTHORIZED
```

Figure 2-13 RACF error from unauthorized use of TSO NETSTAT command - Specific profile

The user CS09 was given CONTROL access to the specific profile MVS.VARY.TCPIP.STRTSTOP, as shown in Example 2-5.

Example 2-5 Give CONTROL access to MVS.VARY.TCPIP.STRTSTOP

```
PE MVS.VARY.TCPIP.STRTSTOP CLASS(OPERCMDS) ID(CS09) ACCESS(CONTROL)
SETROPTS RACLIST(OPERCMDS) REFRESH
SETROPTS GENERIC(OPERCMDS) REFRESH
```

The V TCPIP,TCPIPC,STOP,OSA22E0 command completed successfully, as shown in Example 2-6.

Example 2-6 Successful START command

```
V TCPIP,TCPIPC,START,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,START,OSA2080
EZZ0053I COMMAND VARY START COMPLETED SUCCESSFULLY
```

2.6.2 TSO NETSTAT and UNIX onetstat command security

The TSO NETSTAT command and the UNIX shell **onetstat** command can be protected from unauthorized use at both the command level (NETSTAT with any option) and the command option level. By defining an SAF profile to represent the NETSTAT command and option, you can grant permission by user or group to the NETSTAT command and its options.

RACF profile details

The SERVAUTH class is used to define a profile to protect the NETSTAT command. The format of the profile name is:

EZB.NETSTAT.*sysname*.*tcpprocname*.*option*

Where:

- EZB.NETSTAT is constant.
- *sysname* is the z/OS image name.
- *tcpprocname* is the TCP/IP stack name.
- *option* is either an asterisk (*), meaning all options, or a specific NETSTAT option name.

As mentioned, you can protect NETSTAT/**onetstat** at the command level and the command option level.

Protecting NETSTAT/onetstat at the command level

To specify protection at the command level, specify the SERVAUTH profile with a command option of an asterisk (*) and define the SERVAUTH profile to be generic. Example 2-7 shows how this is done using RACF commands, assuming that the system name is SC30 and the TCP/IP stack name is TCPIPD.

Example 2-7 Protecting NETSTAT/onetstat at the command level

```
SETROPTS GENERIC(SERVAUTH)
RDEFINE SERVAUTH (EZB.NETSTAT.SC30.TCPIPD.*) UACC(NONE)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Note that the SETROPTS GENERIC(SERVAUTH) needs to be done only once.

Protecting NETSTAT/onetstat at the command option level

You may decide to protect the NETSTAT/**onetstat** command at a more granular level so that you can control who has authority to use the options of the NETSTAT/**onetstat** command. To protect the command options, define the particular NETSTAT option that you want to protect as the last qualifier in the profile name. Example 2-8 shows the NETSTAT HOME or **onetstat -h** command being protected for TCP/IP system TCPIPC on z/OS system SC30.

Example 2-8 Defining NETSTAT profile to protect NETSTAT/onetstat command with home option

```
RDEFINE SERVAUTH (EZB.NETSTAT.SC30.TCPIPD.HOME) UACC(NONE)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Restriction: The NETSTAT DROP command is internally implemented as a z/OS console command VARY TCPIP,,DROP, and as such is protected by the SAF profile MVS.VARY.TCPIP.DROP, not by a NETSTAT SAF profile.

SAF checking

The NETSTAT SAF check is performed whenever a TSO NETSTAT or UNIX **onetstat** command is attempted. An SAF check is performed against the most specific profile name first. If a profile for the specific command option does not exist, then a check is made against the generic profile (the profile with a command option specified as an asterisk (*) is the generic profile).

NETSTAT security scenario

Our system name at the ITSO is SC30 and the TCP/IP stack name is TCPIPD. Our tests were to use the TSO NETSTAT HOME command. For the first test, we defined the generic NETSTAT profile to protect all options of the NETSTAT command. For the second test, we defined the profile to protect the NETSTAT HOME command. Both times, unauthorized use of the command caused the expected RACF error.

Note: In the discussions that follow, wherever the TSO NETSTAT command is mentioned, the OMVS **onetstat** command is also implied.

Defining the NETSTAT generic profile to protect all NETSTAT commands

The generic profile EZB.NETSTAT.SC30.TCPIPD.* was added to the SERVAUTH class with UACC(NONE), as shown in Example 2-7 on page 24. At this stage, no user had any access to this profile. A user then attempted a TSO NETSTAT HOME command, which resulted in the RACF error shown in Example 2-9. Note that the profile being RACF checked was EZB.NETSTAT.SC30.TCPIPD.HOME, but because that did not exist, the profile EZB.NETSTAT.SC30.TCPIPD.* was checked.

Example 2-9 Access denied by generic profile

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
   EZB.NETSTAT.SC30.TCPIPD.HOME CL(SERVAUTH)
   INSUFFICIENT ACCESS AUTHORITY
   FROM EZB.NETSTAT.SC30.TCPIPD.* (G)
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
 Access to Netstat HOME denied - SAF RC is 00000008
```

Defining the NETSTAT profile to protect the NETSTAT HOME command

The specific profile EZB.NETSTAT.SC30.TCPIPD.HOME was added to the SERVAUTH class with UACC(NONE), as shown in Example 2-8 on page 24. At this stage, no user had any access to this profile. User CS09 then attempted a TSO NETSTAT HOME command, which resulted in the RACF error shown in Example 2-10.

Note that the profile that has been SAF checked is now EZB.NETSTAT.SC30.TCPIPD.HOME rather than EZB.NETSTAT.SC30.TCPIPD.*.

Example 2-10 RACF error from unauthorized use of TSO NETSTAT command - Generic profile

```
ICH408I USER(CS09    ) GROUP(SYS1    ) NAME(TEST          )
  EZB.NETSTAT.SC30.TCPIPD.HOME CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
Access to Netstat HOME denied - SAF RC is 00000008
```

The OMVS command equivalent of TSO NETSTAT HOME is **onetstat -h**. The same user who attempted the TSO NETSTAT command in Example 2-10 used the command **onetstat -h -p tcpipc** (the **-p** parameter targets the TCP/IP stack named TCPIPD) and got the expected error, as shown in Example 2-11.

Example 2-11 OMVS error from unauthorized use of the onetstat command

```
CS09 @ SC30:/u/cs09>onetstat -h -p tcpipd
EZZ2385I Access to Netstat -h denied - SAF RC is 00000008
CS09 @ SC30:/u/cs09>
```

2.6.3 Policy Agent command security

The z/OS UNIX **pasearch** command queries information from the Policy Agent (PAGENT). The policy Agent contains sensitive information about the policies that control the security of your network such as IP Security (IPSec), Application Transparent Transport Layer Security (AT-TLS), Intrusion Detection, VPN tunnels, and so on. This command should be restricted to those network and security administrators who need to know policy settings.

You can define the following RACF profile EZB.PAGENT.*sysname.tcpname.policy_type* in SERVAUTH class to individually protect each policy type, where:

- *sysname* is the z/OS SMFID.

- *tcpname* is the TCP/IP stack name, which is the name of the TCP/IP started task.
- *policy_type* is the policy type, which can be one of the following:
  - QOS for Policy QoS
  - IDS for Policy IDS
  - IPSec for Policy IPSec
  - TTLS for Policy AT-TLS
  - PBR for Policy-based routing

The policy_type can also be a wildcard character (*) to protect all the policy types with a single profile.

2.6.4 IPSec command access control

The **ipsec** commands are sensitive and are used to display and monitor IP Security management activities. The RACF profiles in the SERVAUTH class required to protect these commands are:

- EZB.IPSECCMD.*sysname.tcpprocname*.DISPLAY
- EZB.IPSECCMD.*sysname.tcpprocname*.CONTROL

You can also define a single generic profile EZB.IPSECCMD.*sysname.tcpprocname*.* to control both of these commands.

2.6.5 For more information

For more information about the profile names used to protect the various commands and options, see *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780.

2.7 Protecting FTP-related resources

In this section, we discuss protecting FTP-related resources.

2.7.1 FTP SITE command control

FTP commands SITE DUMP and DEBUG generate a large amount of output and their use should be restricted. The SERVAUTH class profiles that protect these resources are:

- EZB.FTP.*sysname.ftpdname*.SITE.DUMP
- EZB.FTP.*sysname.ftpdname*.SITE.DEBUG, where:
  - *sysname* is the z/OS SMFID.
  - *ftpdname* is the name of the FTP started task.

2.7.2 FTP server access control

To control the ability to access the FTP server based on SAF user ID associated with TLS-authenticated X.509 client certificate, you need to define the profile EZB.FTP.*sysname.ftpdname*.PORTxxxxx in the SERVAUTH class.

2.7.3 FTP z/OS UNIX access control

This provides the ability to protect z/OS UNIX access by FTP users. The profile name is of the following form: EZB.FTP.*sysname.ftpdaemonname*.ACCESS.HFS. For example, the profile name for FTP daemon FTPD running on system MVSA would be EZB.FTP.MVSA.FTPD1.ACCESS.HFS.

2.7.4 RACF-delegation of cryptographic resources

If you are securing connections using TLS/SSL for your z/OS FTP server, then you need to permit each FTP client to the sensitive cryptographic resources such as CFSERV and CFSKEYS. You can avoid having to permit each FTP client individually by identifying these resources as 'RACF DELEGATED'. The RACF command to do this is:

```
RALTER CSFSERV CSFENV APPLDATA('RACF DELEGATED')
```

The FTP daemon will now access these resources on behalf of the user, though the user does not have explicit access to these sensitive resources.

2.8 Protecting network management resources

In this section, we discuss protecting network management resources.

2.8.1 SNMP agent control

You can control which of the SNMP subagents are permitted to connect to the SNMP agent. You need to define a SERVAUTH class profile EZB.SNMPAGENT.*sysname.tcpname* for this. After creating the profile, use the RACF PERMIT command to define the userids of those subagents that should be permitted to connect via TCP to the SNMP Agent.

2.8.2 TCP connection information service access control

The TCP connection information service allows network management applications to obtain information about TCP connection activity. Access to this information can be controlled by RACF SERVAUTH class profile EZB.NETMGMT.*sysname.tcpname*.SYSTCPCN. You also need to permit the userids of the applications authorized to access this resource.

2.8.3 CIM provider access control

The Common Information Model (CIM) provides a model for describing and accessing data across an enterprise. CIM providers gather the CIM data. You can control this function and restrict the collection of CIM data to authorized providers by defining the SERVAUTH class profile EZB.CIMPROV.*sysname.tcpname*. Access is granted if the user ID associated with the client of the z/OS CIM server is permitted (has read access) to this resource profile.

2.9 Protecting miscellaneous resources

In this section, we discuss protecting miscellaneous resources.

2.9.1 Digital Certificate Access Server access control

This controls the ability to access the Digital Certificate Access Server (DCAS) based on the SAF user ID associated with the TLS-authenticated X.509 client certificate. The profile that protects this resource is:

```
EZB.DCAS.cvtsysname
```

2.9.2 MODDVIPA utility program control

This restricts the usage of the MODDVIPA utility program (creates new DVIPA on system). The profile that protects this resource is:

`EZB.MODDVIPA.`*`sysname`*`.`*`tcpname`*

2.9.3 Fast Response Cache Accelerator access control

This controls the ability to create a Fast Response Cache Accelerator (FRCA) cache. (FRCA is used by Web servers for caching static Web pages in the stack.) The profile that protects this resource is:

`EZB.FRCAACCESS.`*`sysname`*`.`*`tcpname`*

2.9.4 Real-time SMF information service access control

This restricts access to select real-time SMF records accessible using the SMF information service. It is intended for network management applications. The profile that protects this resource is:

`EZB.NETMGMT.`*`sysname`*`.`*`tcpname`*`.SYSTCPSM`

2.9.5 TCP/IP packet trace service access control

This restricts access to select real-time packet trace records accessible using the TCP/IP packet trace service. It is intended for network management applications. The profile that protects this resource is:

`EZB.NETMGMT.`*`sysname`*`.`*`tcpname`*`.SYSTCPDA`

2.9.6 TCP/IP stack initialization access control

This controls the ability of applications to open a socket before the AT-TLS policy is loaded into the TCP/IP stack. Normally you cannot start any application before the POLICY AGENT address space starts. The profile that protects this resource is:

`EZB.INITSTACK.`*`sysname`*`.`*`tcpname`*

Part 2

Working with digital certificates

In this part, we describe how digital certificates can be implemented and used with a z/OS system. We explain how to set up the various servers and clients to use certificates. Further, we provide information and samples using digital certificates in a policy-based z/OS environment.

We also introduce the terms and technologies used when dealing with and managing digital certificates.

3

Certificate management in z/OS

Digital certificates are usually created and maintained within a central repository. In this chapter, we discuss the use of RACF and the **gskkyman** utility to provide these functions.

The first part of the chapter focuses on an overview of industry standard terms and usages. Next, using RACF and the **gskkyman** utilities, we explain how digital certificates and key rings are created and maintained. Additionally, we show how these utilities can be used to obtain and manage the set of certificates required for the scenarios discussed.

For complete details about certificate management, refer to the following publications:

- *z/OS Security Server RACF Command Language Reference*, SA22-7687
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

The following topics are discussed in this chapter.

Section	Topic
3.1, “Digital certificates overview” on page 32	Basic concepts of digital certificates
3.2, “Digital certificate types” on page 34	Definition and use of types of digital certificates
3.3, “Configuring the utilities to generate certificates in z/OS” on page 39	Identification of utilities and their command structures, which generate digital certificates in z/OS
3.4, “Using certificates in sample IBM environments” on page 46	Examples of using RACF and **gskkyman** to manage digital certificates

3.1 Digital certificates overview

A summary of certificate requirements for SSL/TLS is provided in Table 3-1. If you are not implementing client authentication, you do not need the first two rows of this table. The table assumes that the server will be running on the z/OS host and the client will be running remotely. The second column of the table adds some of the relevant RACF commands. Since **gskkyman** is menu-driven, only RACF commands listed are in this table.

Table 3-1 also assumes that you are using some type of signer or issuer root Certificate Authority (CA), and a separate, personal certificate that is signed by this root CA. There are two variations on this scenario that you might need to be aware of:

1. If you are using a self-signed certificate, Table 3-1 still applies. The self-signed certificate with the private key included is exactly the same as a personal certificate. The self-signed certificate without the private key is exactly the same as a root CA.
2. You, or your CA vendor, will be using intermediate CA certificates. Any intermediate certificates should be placed at the same location as their corresponding personal certificates. They do not need to be loaded into a database with the root CA at all. This is because all intermediate certificates are transmitted, along with the personal certificate, during the SSL/TTLS handshake.

Table 3-1 Summary of certificate requirements for SSL/TLS

Certificate	Does this certificate reside in the z/OS RACF or gskkyman database?	Client's key repository
Client's personal certificate	Does not need to be in the z/OS server's database. The client certificate will be sent out during the handshake and should not exist in any other database.	In the personal certificates section, designated as the application's default certificate.
Root CA of client's personal certificate	ADDed or ALTERed as CERTAUTH and TRUSTed. Connected to server's key ring. Needed to authenticate client certificate when it is received during handshake.	In the Signer Certificates section marked as trusted root CA.
Server's personal certificate	ADDed and CONNECTed to server's key ring as USAGE(PERSONAL) and DEFAULT (default can be overridden with AT-TLS).	Does not need to be in client's database. The server's personal certificate will be sent out during the handshake.
Root CA of server's personal certificate	ADDed as CERTAUTH and TRUSTed. CONNECTed to server's key ring.	In the Signer Certificates section as a trusted root CA.

In the z/OS environment, digital certificates are used by Secure Sockets Layer (SSL) and Transport Layer Security (TLS) to authenticate between the client and the server. The certificates contain encryption keys that are used to encrypt the messages of the session setup protocol.

In practice, an SSL/TLS server is required to send its certificate to the client. Optionally, a server may be configured to request a certificate from the client. This authentication process can be provided natively by the application itself, or the process can be performed transparently to the application by implementing Application Transparent TLS (AT-TLS). Native TLS is sometimes referred to as "internal TLS", and AT-TLS is sometimes referred to as "external TLS".

For discussions in this chapter, SSL and TLS are equivalent unless stated otherwise. And, from a certificate standpoint, AT-TLS and native SSL/TLS are also effectively equivalent.

3.1.1 What is a digital certificate

The process of setting up an SSL/TLS connection is referred to as the *handshake*. The handshake uses X.509 certificates (see RFC 2459 for a definition) to verify identity (authenticate) during this handshake. The handshake uses public and private keys (contained within the X.509 certificate) that enable one partner of the connection to confirm that the other partner is who it says it is.

Public and private keys exist in pairs. A private key can encrypt a message, but only its associated public key can decrypt. The reverse is also true: anything encrypted with a public key can only be decrypted by the associated private key.

Private keys are never sent over a network. Only the “owner” of a private key can sign a certificate to prove its identity. Generally, only one copy of a private key is ever in existence, and it resides within the key repository of that server (or client) that needs to prove its identity. Public keys are freely distributed with the X.509 certificate.

3.1.2 How digital certificates work

This section first discusses terminology used with digital certificates, and then examines details of the contents of a certificate.

Subjects and issuers

Within a certificate there are two distinct identification areas which are referred to as the Distinguished Names (or DNs): the issuer’s DN and the subject’s DN.

Issuer’s DN The issuer’s DN field identifies the certificate that was used to sign (and create) this certificate. A certificate’s issuer might also be referred to as the “signer”. The terms are synonymous.

Subject’s DN The subject’s DN of a certificate is the field that identifies the owner of the certificate itself. The subject’s DN might also be referred to at the “owner’s” DN. An owner of a certificate might be an application (such as a server), or even a person, a workstation, or a whole department.

To create a digital signature within the certificate, the certificate issuer first generates a character string from the subject’s Distinguished Name (DN), the subject’s public key, and the issuer’s DN. The character string is the result of hashing the information down to a few bits, usually between 128 to 160 bits. The string of bits is referred to as a *message digest*, which is unique for the given information. The message digest is then encrypted using the issuer’s private key, creating the sender’s signature. The hashing algorithm is included in the certificate.

Returning to an SSL/TLS handshake, when the remote endpoint receives a certificate and message digest, the remote endpoint will use the issuer’s public key to decrypt the message digest. It next uses the included algorithm to create a second string based on the same two DNs and the public key as originally used by the sender. If the second string exactly matches the decrypted string, the remote endpoint is assured that the sending endpoint is genuinely in possession of the private key. Therefore, the sender can be trusted.

But, how did this hypothetical remote endpoint come to already posses the public key that was able to decrypt the digest? The public key must, at some earlier time, have been placed into the certificate repository of the remote endpoint. As a general rule, public keys are freely distributed. Some are so freely available as to be populated, by default, in certificate repositories from the application or operating system vendor. For example, Windows®, Linux and RACF environments already contain certificates with public keys from external vendors such as VeriSign and thawte. Such vendors are referred to as well-known Certificate

Authorities, or well-known CAs. We discuss CAs in more detail in 3.2, “Digital certificate types” on page 34.

3.2 Digital certificate types

The following digital certificate types are described in this section:

- Certificate Authority certificates
- User (personal) certificates
- Site certificates

3.2.1 Certificate Authority certificates

A Certificate Authority (CA) certificate is one which signs or issues other certificates. There are two types of CA certificates:

root CA A root CA is the very first certificate created; it forms the top of any certificate chain. The root CA is used to sign other certificates that are lower down in the chain (hierarchy). A unique characteristic of a root CA is that the issuer and subject names are the same. This makes sense: the root CA is first in line: how could there possibly be an issuer? Thus, a root CA is signed by its own private key.

intermediate CA An intermediate CA is a certificate that has been signed by a root CA or signed by another intermediate CA. In other words, an intermediate certificate is a certificate that is not a root certificate or a personal certificate.

Certificates exist in hierarchical chains of authority, with the uppermost authority always being the root CA. A root CA from VeriSign, for example, would be referred to as a well-known root CA. Intermediate certificates are becoming more popular. For example, as of April 2006, VeriSign only distributes certificates that have been signed by a VeriSign intermediate certificate.

The certificate hierarchy might exist for organizational purposes. For example, ITSO Electronics Company might issue a root CA that represents the highest level of authority or control in the organization. This root CA might be used to sign several intermediate CA certificates that are used by individual departments.

As mentioned, a CA certificate can be provided to you by a well-known CA that is in the business of providing certificate services (see 3.2.4, “How a digital certificate can be obtained” on page 37). In addition to being commercially provided, a CA certificate can be created internally by your own company. Such a local root CA would be used in signing and validating other internal certificates.

Well-known Certificate Authority certificates

In 3.1.2, “How digital certificates work” on page 33 we mentioned that well-known root CAs are distributed by a few external vendors. This is a useful technique. By ordering and using a certificate from a well-known CA, an organization will be assured that the public key is already in their key repositories. No distribution of the public key is required.

Locally signed CA certificates

A locally provided CA certificate is a certificate that has been generated to indicate its use as a CA certificate. As the name suggests, it is locally generated. Although it is a root CA certificate, it is not from a well-known CA vendor. Entities that want to authenticate a

certificate signed by this local CA must have the public root CA certificate installed into the entity's certificate repository manually.

Locally signed certificates are not as straightforward to use as those signed by a well-known CA. The reason is because, to use a locally signed certificate, the local root CA certificate must be manually distributed to all participating clients. Many clients allow this to be done dynamically, the first time a handshake is attempted. However, this represents a security exposure: it is up to the user to determine whether this "new" CA is really something to be trusted.

3.2.2 User (personal) certificates

CA certificates are used in two contexts. First, as part of the environment setup process, a CA certificate (intermediate or root) can be used to sign another certificate. Second, as part of the SSL/TLS handshake itself, the root CA certificates are used to authenticate (using a public key) a certificate or certificate chain that is received during the handshake. A certificate that is sent out by an endpoint during the handshake to identify itself is referred to as a *user certificate* or *personal certificate*.

Intermediate certificates

It is a requirement of the SSL/TLS protocols that all known intermediate certificates be sent out during the handshake. That means if an FTP server, for example, wants to prove its identity to an FTP client, the FTP server must send out its user certificate along with all intermediate certificates, all the way up to the root certificate. The only certificate that is required to be already present at the client end is the root CA certificate of this chain. Because an intermediate certificate is a signer of either another intermediate certificate or of a user certificate, an intermediate certificate is really a CA certificate.

Public and private keys

The private key of the user certificate must be locally present and accessible in order for an endpoint to create its message digest. This message digest is used to prove an endpoint's identity. However, all intermediate certificates, and root CA certificates used to authenticate a certificate, should only have their public keys included. Public key authentication can only work in this world if the private key is tightly controlled and only accessible to the endpoint that wants to prove its identity.

Can those public keys be used to decrypt information encrypted by other public keys? No. Only the private key can decrypt information which has been encrypted by a matching public key. As long as this server is the only server that has access to the private key, information encrypted by the public key is very secure.

This public and private key pair is never used for encrypting and decrypting data. Instead, the public and private keys are used to communicate a new key that will be used for data encryption. The cryptography used for public and private key communication is referred to as *asymmetric encryption*. The single new key used to encrypt and decrypt data is referred to a *symmetric encryption*.

For more information about cryptography, including public and private key pairs, see Appendix A, "Basic cryptography" on page 657.

Certificates at the server endpoint

The SSL/TLS protocol normally requires a server to supply a digital certificate (and intermediates) to a client. Next, the client must validate the server certificate by checking the trusted root CA in its own key database. The client can then use the server's public key from the certificate to communicate the rest of the SSL handshake.

Certificates at the client endpoint

Often, SSL/TLS is implemented such that the server is the only entity that is required to prove its identity. The client's role in the handshake is to authenticate the certificate (or certificate chain) sent out by the server. As mentioned, all that is required to authenticate the server is the root CA of the chain. However, the handshake can have an additional step required, where the server requests that the client prove its identity after the server has done so.

Client authentication

An SSL/TLS-enabled server may be configured to request that the client provide a digital certificate (or certificate chain) to verify the client's identity. The server must then validate the client certificate by checking that it has the trusted root CA in its key database. The server does not make use of the client's public key contained in the certificate for any handshake communications; this request is for identification purposes only.

A good way to view client authentication is to consider it a mirror image of server authentication. In a RACF context, when client authentication is requested by the server, the server will be configured to authenticate the client to a particular level. These levels are:

Level 1 The server ensures that the signer of the client's certificate is trusted by checking the trusted root CA certificate that is in the server's key ring.

Level 2 The authentication requires that the client certificate also be registered with RACF (or another SAF-compliant security product) and that it be in TRUSTED status. The RACF user ID that the certificate is associated with is that given in the `ID()` parameter of the RACDCERT ID() ADD command when the client certificate was added to RACF. The CA that issued the client certificate must have a CA certificate connected to the server's key ring. Note that this level cannot be used if the z/OS server is using a key database created by using **`gskkyman`**.

Level 3 The authentication provides, in addition to level 1 and level 2 support, the capability to restrict access to the server based on the user ID returned from RACF. This level is implemented entirely in RACF; that is, a server only selects level 2 authentication, and if the appropriate profiles for the server are defined in RACF, the authentication level is upgraded to level 3. (Note that this level cannot be used if the z/OS server is using a key database that is created by using **`gskkyman`**.)

Self-signed certificates

A root CA certificate has an issuer DN the same as its subject DN. A root CA is the first in the chain, and it consequently must use its own private key to sign itself—which is the meaning of self-signed. Consequently, all root CA certificates are self-signed. This is true for both local and well-known CA certificates.

However, in some contexts, rather than using the root CA to sign a user certificate, the root certificate itself is used in the handshake. This can only be a locally signed root certificate, because vendor-supplied root CAs will never include the private key (and a private key is needed to create that message digest). Usually the use of self-signed certificates is limited to initial testing efforts. In practice, a locally signed user certificate is preferred.

Whether locally signed or signed by a well-known CA, a user certificate is owned by a specific user ID, or it can be a site certificate owned by the user ID, SITE (irrsitec). But it has not been

signed nor validated by *any* CA-certificate (commercially provided or internally provided). This means that the digital signature on the certificate can only be verified by the public key given on the *same* certificate. Because the certificate is not authenticated by any CA, it must be taken at face value by the client or server receiving it.

The validation procedure for a certificate is still performed for a self-signed certificate. This means that receivers check their key database for the CA certificate that signed the certificate, but the CA is represented by the certificate itself. Therefore, the self-signed certificate must have been previously received by some other means and preloaded in the receiver's key database as a trusted certificate.

3.2.3 Site certificates

Site certificates are unique to the RACF environment. They represent an extra level of control that is not present in many other certificate repositories. In other repositories (including the gskkyman key database), if a private key is present in the repository, then any user or process that has access to that repository will also have access to the certificate's private key. In RACF, only the user ID that is associated with a user certificate can have access to the private key. A site certificate is a special user ID in the RACF environment that allows the sharing of the private key among more than one user ID.

A site certificate is associated with a single location entity consisting of multiple servers, multiple clients, and other multiple instance applications, such as a sysplex or a data center. As long as the appropriate RACF permissions are in place, any user ID in the LPAR or sysplex can take advantage of a site certificate. These multiple users can share a site certificate, which avoids having to create and manage a unique certificate for each one.

Although this arrangement eases the burden of certificate management, it also reduces the granularity of control that an organization has: imagine a scenario where a single certificate is used for all TN3270 and FTP servers across all images in all sysplexes. If this certificate's private key were to be compromised, then all servers would be impacted. If separate certificates were used for each server instance, the impact of a compromised private key would be far less. However, the likelihood of a compromised private key is incredibly small, so most organizations use a shared site certificate.

When a server within the organization has a copy of this site certificate and has marked it as trusted, even the client can use the site certificate to send to a server during server-required client authentication. Because the client is sending a certificate to the server that the server already has marked as TRUSTED, client authentication succeeds.

3.2.4 How a digital certificate can be obtained

There are three ways for you to prepare a certificate for use in SSL/TLS connections. In this section, we describe the ways in which you can obtain a digital certificate, and explain which way is most appropriate for which usage.

- Request that a well-known CA sign your certificate

  If you are requesting a certificate for a server, and you plan to make your server available to the public or your business partners, you should get your certificate from a well-known CA. As mentioned, well-known CA certificates already have their CA root certificates contained in the default key repository of SSL/TLS applications.

- Generate a certificate yourself

  This type of certificate is called a *self-signed* certificate, because the issuer of the certificate is the same as the subject of the certificate. This type of certificate might be useful for testing purposes, or for securing TLS connections within your intranet.

- Create a self-signed CA certificate and function as a local CA

  To validate a certificate, the receiver checks its key database for a trusted root CA certificate that has the same subject DN as that of the received certificate's issuer. The root CA certificate must be located in the receiver's local database and marked as trusted; most systems refer to this database as a *key ring*. This method is illustrated in Figure 3-1.

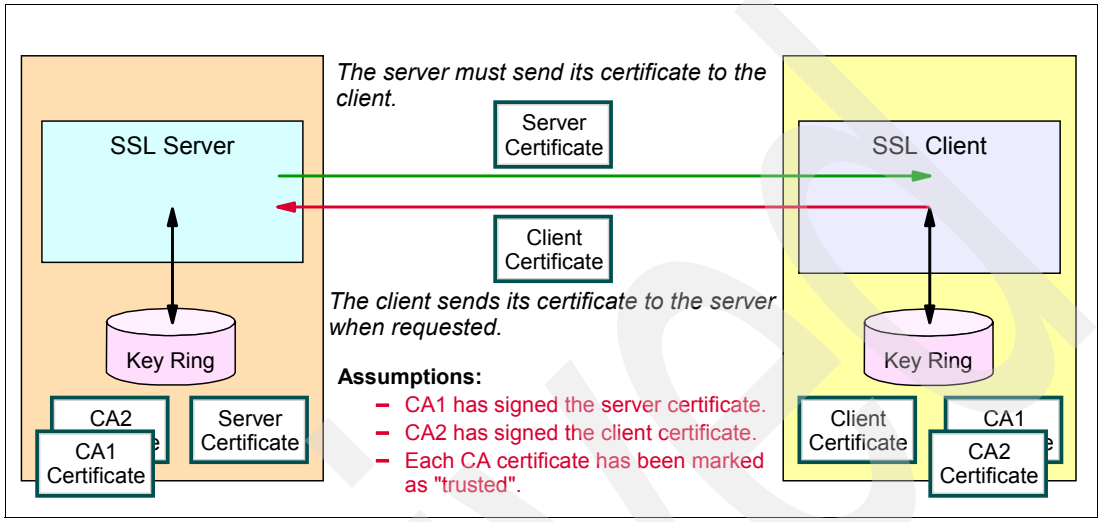


Figure 3-1 SSL certificate management: CA-signed certificates

The example in Figure 3-1 has the client sending a certificate back to the server. This implies that the server is configured to require client authentication. In this illustration, the private key for the server certificate must exist on the SSL server endpoint. Only the public key, within the root CA certificate CA1, should exist in the client's database.

Likewise, the private key for the client's personal certificate should exist only in the key ring on the SSL client host. The public root CA certificate (CA2) that matches the client's certificate would need to be in the SSL server's key ring. The personal server certificate and the personal client certificate should not be the same certificates. However, it is acceptable to have both certificates signed by the same root CA. In other words, CA1 could be used to sign both the server certificate and the client certificate.

If you choose to use a self-signed certificate (shown in Figure 3-2 on page 39) instead of a CA-signed certificate, the CA certificate that should be trusted is similar to the server/client certificate itself. If the server itself has signed its own certificate and client authentication is not required, the server certificate (public key only) 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.

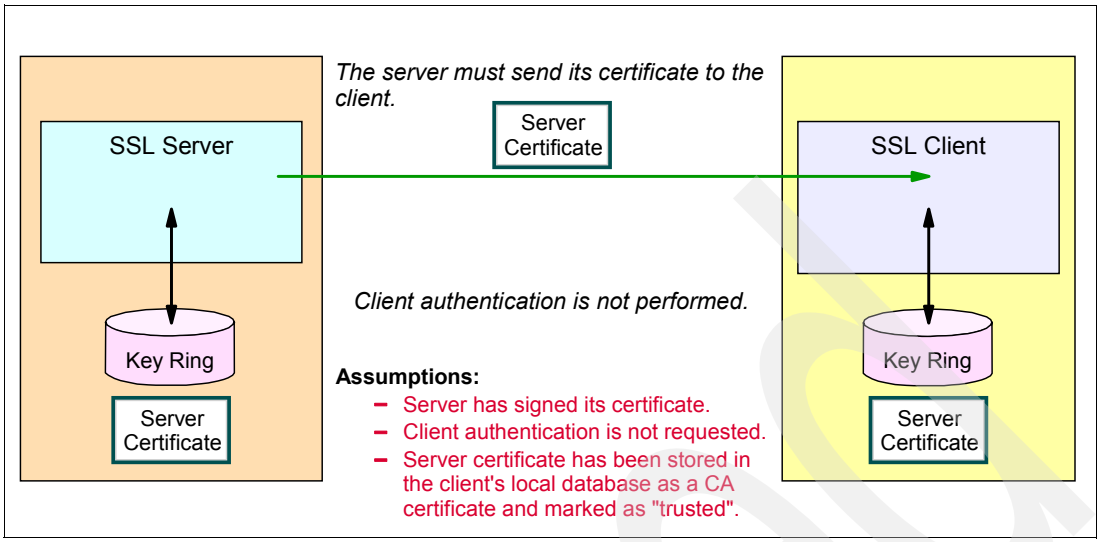


Figure 3-2 SSL certificate management: Self-signed certificate without client authentication

In Figure 3-2, the private key of the certificate need only be available to the SSL server. On the client side, only the public key is required.

3.3 Configuring the utilities to generate certificates in z/OS

For detailed information about the creation and maintenance of digital certificates in z/OS, see the “Certificate/Key Management,” topic in *z/OS Cryptographic Services System SSL Programming*, SC24-5901. For a reference on the RACDCERT command, see *z/OS Security Server RACF Command Language Reference*, SA22-7687.

Table 3-2 lists the certificate management utility topics in this section.

Table 3-2 Certificate management utilities and functions

Section	Description
3.3.1, “Utilities in z/OS for managing certificates” on page 39	Positions RACF and gskkyman
3.3.2, “Digital certificate field formats” on page 40	Field format and structure of a certificate
3.3.3, “Using the RACF RACDCERT command” on page 41	How to use the RACDCERT command
3.3.4, “Using the gskkyman command” on page 44	How to use the **gskkyman** command

3.3.1 Utilities in z/OS for managing certificates

Most SSL-enabled applications in z/OS make use of the System SSL toolkit. For certificate storage and management, two command utilities exist:

- The RACF command RACDCERT creates and maintains certificates and key rings that are stored in the RACF database. This command can also be used to create self-signed certificates, local CA certificates as well as certificate requests for other CAs.

- The **gskkyman** utility creates and maintains a key database as a file in the z/OS UNIX file system. It can also create self-signed certificates, local CA certificates as well as certificate requests for other CAs.

Using RACF key rings is, in many organizations, the preferred method because it provides better security for the certificates and their private keys. With RACF key rings, stash files containing key database passwords are not used and access to key rings, certificates and private keys is controlled by RACF. In this section, we show both methods of creating and managing certificates.

3.3.2 Digital certificate field formats

When you create a digital certificate, whether using **gskkyman** or RACDCERT, certain fields are required and others are optional.

- Distinguished name (DN): The issuer of a certificate and the subject of a certificate are both represented by a Distinguished Name. For a self-signed or root CA certificate, the issuer's Distinguished Name will be copied from the subject's Distinguished Name. A Distinguished Name contains the following subfields (with RACDCERT parameter names in parentheses):
  - Common Name: (CN). For a server certificate, this field often contains the server's DNS name. For a client certificate, this will identify the individual or computer (again, a DNS name might be used).
  - Organization-name: (O). Company name or similar.
  - Title: (T). Salutation for an individual.
  - Organizational-unit: (OU). Used for classification within the Organization-name, listed above.
  - Locality: (L). City or town.
  - State-or-province: (SP).
  - Country: (C). Two-character ISO code for country.
- Period of validity: The **gskkyman** utility asks for the number of days from today that the certificate is valid for. RACDCERT sets the lower and upper dates with the NOTBEFORE(DATE(yyyy-mm-dd) and the NOTAFTER(DATE(yyyy-mm-dd) parameters.

> **Note:** Choose an expiry date carefully, particularly for any root or intermediate CA certificates. After SSL/TLS is in place, it is transparent to users. Consequently, it is readily forgotten that it is being used at all. Be sure to put some kind of a formal reminder in place so that an expired certificate does not come as a surprise.

Some SSL/TLS applications will automatically compare the CN of the received certificate to the DNS name of the sender. If they match, the handshake is allowed to proceed. This is not part of the SSL/TLS standard and most applications now allow this option to be turned off, if desired.

The label field is also needed. This field is not part of the X.509 specification, but it is used to organize certificates in the key database. You can use the label to list, alter, and delete individual certificates. A label must be unique except for storage within RACF, where labels can be duplicated as long as they are associated with different RACF user IDs (with the `ID(userID)` parameter).

3.3.3 Using the RACF RACDCERT command

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 key rings 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.

The RACF environment always contains a certificate database. In order to access any certificates, a logical key ring must be created. Certificates are connected to the logical key ring as they are needed. An application, in turn, will make reference to this key ring.

The topics discussed in this section are:

- RADCERT command format
- RADCERT command supported certificates
- RADCERT command supported key rings
- RADCERT command security

RACDCERT command format

```
RACDCERT [ID(user) | SITE | CERTAUTH] command-options
```

The RACDCERT command can be directed to a RACF user ID's digital certificates or key rings by the `ID(user)` parameter, to a CA's resources by the `CERTAUTH` parameter, and to a site's resources by the `SITE` parameter. If no `ID`, `SITE`, or `CERTAUTH` parameter is included, the command issuer's user ID is used.

For instance, the **`racdcert certauth list`** command lists all CA certificates in the RACF database. The **`racdcert list`** command shows all of your (that is, the command issuer's) certificates.

There is also a `MULTIID` parameter for mapping functions. This and other parameters are explained fully in *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

RACDCERT command supported certificates

RACF distinguishes three types of digital certificates:

- Certificate Authority certificates: These certificates are associated with CAs and are used to verify signatures in other certificates.
- Site certificates: These certificates are associated with a location comprised of multiple servers or systems, such as a data center sysplex, all of which can be identified by the same certificate. Each server or system does not need a unique certificate, they can share a common site certificate.
- User certificates: These certificates are associated with a RACF user ID and are used to authenticate a user's identity. The user can be an individual person or a server started task.

A user (personal) certificate or certificate that has been connected to a key ring with USAGE(PERSONAL) is the only type of certificate whose private key can be used to create the message digest during the SSL/TLS handshake. Therefore, all server certificates for local servers need to be user certificates, or they need to be connected to an appropriate key ring with USAGE(PERSONAL). As mentioned earlier, servers can share a site certificate to avoid the burden of maintaining multiple personal certificates.

Although CA certificates and user (or personal) certificates are industry standard designations, the term “site” is unique to the RACF environment (it does not even have an equivalent in **gskkyman**).

The RACF ISPF panels can be used to maintain the digital certificates if you do not choose to use the TSO RACDCERT command. We used the TSO commands in our examples because they can be submitted in a batch job.

RACDCERT command supported key rings

Using a RACF key ring is a way to logically group together a number of certificates. A certificate can be connected to one or more key rings.

Each key ring is associated with only one user ID, but the certificates that are connected to that key ring may or may not be the key ring owner’s certificates.

Figure 3-3 shows the logical relationship between RACF key rings and digital certificates stored in the RACF database. There can be more than one key ring in the database with the same name, but each must be assigned to a different user ID.

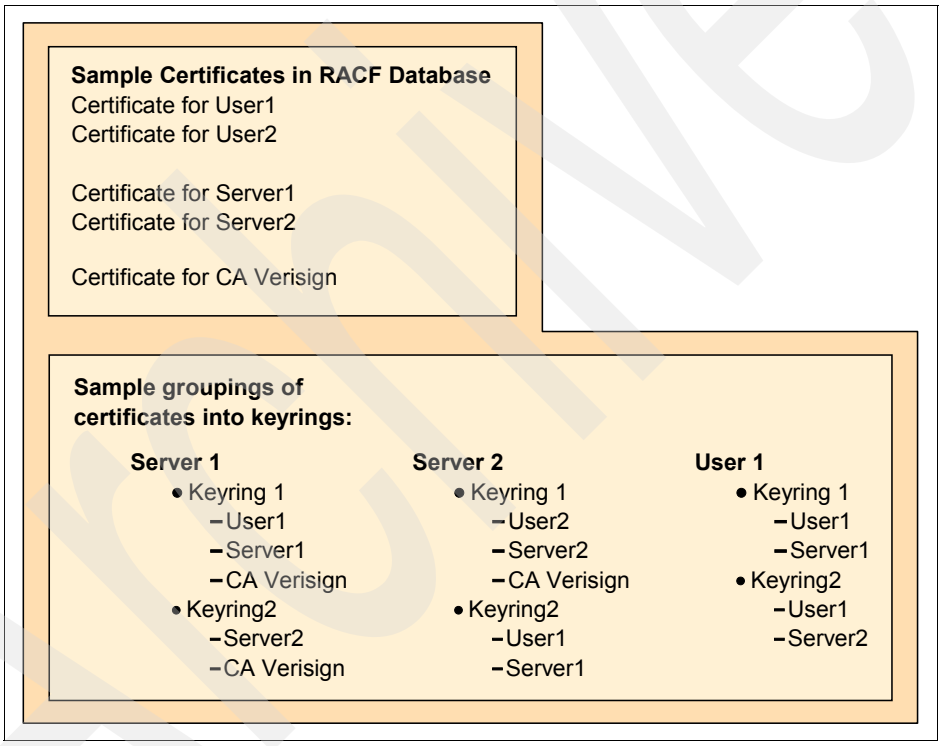


Figure 3-3 Example showing how key rings contain pointers to certificates

Typically, a z/OS server (or AT-TLS, on the server’s behalf) that uses digital certificates will have a configuration parameter where the RACF key ring name is specified. TN3270 and FTP are examples of servers that use key rings. During SSL/TLS session setup, the server sends its certificate to the client. The server will get its certificate, and any intermediate certificates, from the RACF key ring specified in the server’s configuration file and that is associated with the server’s RACF user ID. A server may also look at the certificates in its key ring for one or more root CA certificates with which to validate client certificates, if client authentication is configured.

Note: If the TN3270 server is configured for AT-TLS support (TTLSPORT nnn), the key ring name is not specified within the server's configuration profile. Instead, the key ring is specified within the AT-TLS configuration statements section for the Policy Agent.

Likewise, if the FTP server is configured for AT-TLS support (TLSMECHANISM ATTLS), the key ring name is not specified within the server's `FTP.DATA` configuration file. Instead, the key ring is specified within the AT-TLS configuration statements section for the Policy Agent.

A z/OS client that uses digital certificates, such as FTP, will use a key ring to access its certificates. During SSL/TLS handshaking, the server sends its certificate to the client, and the client looks at the certificates in its key ring for a root CA certificate with which to validate the server certificate. If the server requests that the client send a certificate, the client will get its certificate from the specified key ring. Note that a client certificate must be in trusted status in the RACF database.

Note: If the FTP client is configured for AT-TLS support (TLSMECHANISM ATTLS), the key ring name is *not* specified within the client's `FTP.DATA` configuration file. Instead, the key ring is specified within the AT-TLS configuration statements section for the Policy Agent.

Sharing a key ring

While many key rings can be created, there are times when using one key ring for many applications or user IDs (FTP clients, for example) is advantageous. If user ID MATT wants access to a key ring owned by user ID BILL, there are two things that need to be done:

1. Prefix the RACF key ring name with the ring owner's user ID, in the format user/keyring. (For example, BILL/TESTRING1.)
2. Make certain that user ID MATT (the user ID wanting access to the other user ID's key ring), has UPDATE access to IRR.DIGTCERT.KEYRING in the FACILITY class.

A shared key ring allows access to public keys only, unless a certificate is designated as a SITE certificate. For accessing a private key in a shared key ring, see 3.2.3, "Site certificates" on page 37.

RACDCERT command security

Authority to the IRR.DIGTCERT.function resource in the facility class allows a user to issue the RACDCERT command. To issue the RACDCERT command, users must have one of the following RACF authorities:

- The SPECIAL attribute.
- Sufficient authority to 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).

Important: Any z/OS-based client or server that uses a RACF key ring issues internal RACDCERT LIST and RACDCERT LISTRING commands. The RACF user ID associated with the server must therefore be granted READ access to the RACF profiles controlling these commands, which are IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING. For a list of servers that use RACF key rings, see Table 3-2 on page 39.

Figure 3-4 shows the RACF commands needed to permit a user (TCP/IP, in this case) to issue the RACDCERT LIST and RACDCERT LISTRING commands.

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
SETR RACLIST(FACILITY) REFRESH
```

Figure 3-4 RACF commands to the TCP/IP user ID

For detailed RACF security requirements, see *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

3.3.4 Using the gskkyman command

The **gskkyman** UNIX command is used to create and maintain digital certificate key databases in a z/OS UNIX file system. This is an alternative to storing digital certificates in the RACF database. Note that if you are using SSL/TLS client authentication to map a digital certificate to a RACF user ID, then you must use the RACF RACDCERT command to store the client certificate, not **gskkyman**.

In RACF, a key database is always present. In a **gskkyman** environment, a key database must be explicitly created as a certificate repository. In turn, this key database can be shared by any user IDs that have access via the HFS.

In the examples shown later in this chapter, we assume that a **gskkyman** key database has been set up. The procedure to set up a new key database (and stash file) is as follows:

1. Set up access to the **gskkyman** command from your UNIX shell. This process is covered in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.
2. From the UNIX shell, enter the **gskkyman** command. Figure 3-5 on page 45 shows the initial panel. This example shows how to create a new key database in the z/OS UNIX file system. The database will be created in the subdirectory from which you entered the **gskkyman** command. The password you enter here will be used to open the database in the future.

```
CS08 @ SC63:/>gskkyman

Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

  11 - Create new token
  12 - Delete token
  13 - Manage token
  14 - Manage token from list of tokens

   0 - Exit program

Enter option number: 1
Enter key database name (press ENTER to return to menu): matt.kdb
Enter database password (press ENTER to return to menu):
Re-enter database password:
Enter password expiration in days (press ENTER for no expiration):
Enter database record length (press ENTER to use 2500):

Key database /u/cs08/matt.kdb created.
```

Figure 3-5 Setting up a new key database in a z/OS UNIX using gskkyman

Because the key database has a password, there must be a mechanism for a server to supply it to read the contents. This mechanism is implemented by using a stash file, which is a file using the same name as the key database, but with a suffix of `.sth` rather than `.kdb`. This file contains the key database password in encrypted form, and is created from the **gskkyman** panel.

3. After successfully creating a new key data base (or opening an existing key database) you should see an option 10 that allows the storing of the password in a stash file. This is shown in Figure 3-6.

```
CS08 @ SC63:/>gskkyman

Key Management Menu

       Database: /u/cs08/matt.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Database password stored in /u/cs08/matt.sth.

Press ENTER to continue.

CS08 @ SC33:/u/cs08>ls -la matt.*
-rw-------   1 CS08     SYS1       45080 Sep 24 20:16 matt.kdb
-rw-------   1 CS08     SYS1          80 Sep 24 19:24 matt.rdb
-rw-------   1 CS08     SYS1         129 Sep 24 23:32 matt.sth
```

Figure 3-6 Creation of the stash file using gskkyman

Note that after the stash file was created, the UNIX file attributes were displayed with the UNIX **ls** command. As you can see in Figure 3-6, the file attributes of the key database and the stash file are both `-rw-------`, which means only the creator of the database (the user of the **gskkyman** command) can read and write to this file. You should use the UNIX **chmod** command to set the permission bits so that the server's UNIX UID is able to read both the key database and the stash file. An example command to allow the owner read/write access and the owners group to have read access is **chmod 640 matt.***.

Note that we see an extra file in the list output: `matt.rdb`. This rdb file is a repository for holding certificate request information.

3.4 Using certificates in sample IBM environments

This section shows the practical steps necessary to obtain digital certificates in a z/OS environment using **gskkyman** and RACF. In addition, Personal Communications and IBM HTTP server examples are covered to a limited extent.

In all the examples that follow, the server runs on z/OS under the RACF user ID TCPIP, and the end-user's RACF user ID is CS08. The end-user's user ID is only needed when you are storing client certificates in RACF using RACDCERT.

Table 3-3 lists the sections that show how to create, verify, and manage the indicated certificate types.

Note: Using a shared site certificate connected to a shared key ring is the recommended certificate management scheme. It provides the most efficient and least complex method of certificate management. As mentioned earlier, this does mean that the same private key may be used for all your servers, leaving you with a single point of failure in the unlikely event that the user certificate or private key is compromised.

The other listed procedures show the details of each certificate type.

Table 3-3 Certificate types

Section	Description
3.4.2, “Shared site certificate and shared key ring” on page 47	Preferred scheme for managing certificates within z/OS sysplexes and data centers
3.4.3, “Self-signed certificates” on page 55	A method for testing new implementation
3.4.4, “Internal (local) Certificate Authority” on page 70	Often used for intranet LAN access and internal users
3.4.5, “External (well-known) Certificate Authority” on page 74	Used for formal external user access

3.4.1 Host On-Demand and certificates

An excellent Host On-Demand (HoD) certificate presentation is available at the following URL:

http://www-1.ibm.com/support/docview.wss?rs=132&context=SS5RCF&q1=ssl&uid=swg27008673&loc=en_US&cs=utf-8&lang=en

For further information, refer to the Host On-Demand product documentation. Host On-Demand, and many other SSL/TLS-capable clients, is capable of dynamically accepting a server certificate during the handshake. Although this can certainly decrease the work involved with distribution and maintenance of certificates, it does present a security exposure. A user could inadvertently direct the Host On-Demand client to the wrong server, and subsequently accept the server certificate that is presented. This would result in a secure connection to a potentially non-secure server.

3.4.2 Shared site certificate and shared key ring

The preferred method in setting up your certificate management scheme is to create a single common key ring to be shared by multiple servers, and generate a single site certificate to be shared by these multiple servers.

The benefit of sharing a key ring

The concept of using a shared key ring for all the servers within a sysplex greatly reduces the security administration work load. Traditionally, each server has been assigned its own key ring which has been defined as being owned by the server’s user ID. The server’s default certificate and any root CAs using in client authentication have had to be connected to the server’s key ring as well. As the number of servers increase, each one with perhaps a unique user ID, the burden of key ring administration has grown. In a data center environment, where the number of system LPARs has increased over the past few years, the increase in key rings

has created a complex work load for the security administration group. The implementation of a shared key ring can greatly reduce the complexity of security administration.

Note that a shared key ring is very useful for clients (like FTP), too. Most often, SSL/TLS clients only need access to public keys of root CA certificates for the purpose of authenticating a server. Because no private key access is required, there is no need to use any site certificates with SSL/TLS clients (unless client authentication is required by the remote server).

The benefit of sharing a common site certificate

The use of individual server certificates also drives up the cost of obtaining multiple certificates, especially if they are obtained from an outside vendor. If a number of servers (for example, those within a sysplex, or perhaps even all the servers within a data center) could share a common site certificate, then the cost of obtaining the certificates, as well as the burden of managing the certificates and the complexity of security administration, could be significantly reduced.

Sharing a key ring and a site certificate

A single site certificate can be imported to multiple systems, even across sysplexes, to all systems within the data center. However, a key ring is an entity only within a single RACF database, which is usually self-contained within a sysplex. So you could have only one site certificate, but have to deploy a few shared key rings, one per sysplex.

The shared site certificate is connected to the shared key ring. The CA certificate that signs the site certificate, whether an internal CA or external CA certificate, must also be connected to the shared key ring because it is the signer of the site certificate. If using a locally signed CA certificate, you will need to import it to all TLS clients that access these servers. At each of the clients, the CA certificate will have to be marked as a trusted root CA certificate.

Setting up an internal (local) CA and creating and signing a site certificate with that CA certificate is very similar to the Internal CA signed server certificate described in “Internal-CA signed server certificate RACDCERT procedure” on page 71. The certificate being distributed to the clients in this scenario is the same internal CA certificate that was created for the individual server certificate.

Sharing a private key requires a high degree of authority for each server involved. The key ring containing the shared certificate must be protected, and each server must be configured to access the shared key ring and have sufficient access authority to read it. In addition, each server must have CONTROL authority for the `IRR.DIGTCERT.GENCERT` resource. This resource controls the server’s ability to retrieve private keys, and is checked when you issue the RACDCERT GENCERT SIGNWITH command.

Configuring the internal CA signed site certificate with RACDCERT

This procedure is similar for any z/OS server. In this example, we are producing a site certificate for use by a TN3270 server and an FTP server. This certificate is needed by TN3270 clients and FTP clients using SSL/TLS.

Whether the servers are using their own native SSL/TLS support, or they are configured to use the external AT-TLS support, the certificate generation process here is the same. In order for the client to validate the SITE certificate, the internal CA certificate will be needed at the client.

The steps are:

1. Create a self-signed certificate for the local (internal) CA, as shown in Figure 3-7 on page 49.

```
//CERTAUTH  JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Step 1:                                                       *
//*     Create Certificate Authority Certificate for ITSO             *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT certauth gencert                      -    1
SUBJECTSDN( O('IBM CORPORATION')               -
                CN( MATT.ITSO.COM )               -
                OU('ITSO CERTIFICATE AUTHORITY')   -
                C('US'))                           -
    NOTBEFORE(DATE(2007-09-11))                    -
    NOTAFTER(DATE(2008-09-11))                     -
    KEYUSAGE (CERTSIGN)                            -
    WITHLABEL('CS19 ITSO CA1')
  SETROPTS RACLIST(FACILITY) REFRESH
  RACDCERT CERTAUTH LIST /*
```

Figure 3-7 Batch job to create internal CA certificate in RACF database

1 Instead of a user ID, the CERTAUTH keyword is used to indicate that this certificate is to be used as a CA certificate, and is not associated with a specific user.

2. Generate a site certificate for the servers to share, as shown in Figure 3-8.

```
//CERTSITE  JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTSITE EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Step 2:                                                       *
//*     CREATE SITE AUTHORITY CERTIFICATE FOR ALL SERVERS (SHARED)    *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT SITE GENCERT SUBJECTSDN(CN('ITSO.IBM.COM')   - 1
       O('IBM CORPORATION')                          -
       OU('ITSO CS19 SHARED SITE')                   -
       C('US'))                                      -
       WITHLABEL('CS19 ITSO SHAREDSITE1')            - 2
       SIGNWITH(CERTAUTH LABEL('CS19 ITSO CA1') 3
  RACDCERT SITE LIST
/*
```

Figure 3-8 Batch job to create SITE certificate and sign with internal CA certificate

1 Instead of a user ID, the SITE ID is used to indicate that this certificate is to be used as a site certificate, and is not associated with a specific user. The SITE parameter is used in this example because the private key of the certificate being generated is to be shared by multiple servers. It is useful (but not required) to make sure that the common name (CN) is the same as the domain name of the site.

2 The LABEL implies that the certificate is a shared site certificate

3 The SIGNWITH parameter indicates that the internally signed CA certificate that we created previously is used to sign this site certificate. The label of the CA certificate is specified to identify the CA certificate. This indicates that the site certificate should be digitally signed with the internal CA's private key.

3. Create a shared RACF key ring for the servers. We suggest that a new user ID and new key ring be created for this purpose. Both the user ID and the key ring could be given names that imply they are related to the shared site certificate.

   For our example, however, we simply associated the new key ring with the FTP server's user ID and then connected the shared certificate to the new ring and marked it as the default certificate.

   Figure 3-9 shows the three steps necessary to create the shared key ring and connect the two certificates.

```
//KEYRINGS  JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRINGS EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//* Step 3:                                                         *
//* Add a new keyring to the various clients' RACF ID , then ...    *
//* Add the SITE certifiate to the servers' keyring.  The           *
//* keyring name in the server configuraton must be changed to      *
//* point to the new ring name as in "KEYRING SAF <userid>/SharedRing *
//* This job assumes that keyrings are associated with userid 'TCPIP' *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT ID(TCPIP) ADDRING(SHAREDRING1) 1
  RACDCERT ID(TCPIP) CONNECT(CERTAUTH -   2
                     LABEL('CS19 ITSO CA1')          -
                     RING(SHAREDRING1)               -
                     USAGE(CERTAUTH)
  RACDCERT ID(TCPIP) CONNECT(SITE -   3
  LABEL('CS19 ITSO SHAREDSITE1')     -
                     RING(SHAREDRING1)             -
                     DEFAULT                       -
                     USAGE(PERSONAL)
  SETROPTS RACLIST(DIGTRING) REFRESH
  SETROPTS RACLIST(DIGTCERT) REFRESH
  RACDCERT LISTRING(*) ID(TCPIP)
/*
```

Figure 3-9 Batch job to add the shared key ring

1 Create a new RACF shared key ring using the RACDCERT ADDRING command.

2 Connect the internal CA certificate to the new key ring using the RACDCERT CONNECT command.

3 Connect the site certificate (which was signed by the internal CA certificate) to the new key ring using the RACDCERT CONNECT command. Even though the certificate was created as a site certificate, the `USAGE` must be specified as `PERSONAL` because the servers use it to authenticate themselves to the clients. It is *this* certificate that the servers send to the client during server authentication.

4. Update the server configurations to point to the new shared key ring and user ID TCPIP:

   Using TN3270:

   ```
   KEYRING SAF TCPIP/SharedRing1
   ```

   Using FTP:

   ```
   KEYRING TCPIP/SharedRing1
   ```

5. Protect the shared key ring and permit the user IDs of each server to read it. If there are any regular user IDs that need to be able to list the key rings, grant them permission at this time.

Figure 3-10 shows the RACF PERMIT commands that are necessary to grant key ring access to the servers sharing the key ring. Because the key ring is associated with the FTPD and TN3270 user ID, the user ID for the servers **1** needs only `READ` access. But any other started task IDs **2** or user IDs **3** need `UPDATE` access because the key ring belongs to a different user ID.

```
//PERMRING  JOB MSGCLASS=X,NOTIFY=&SYSUID
//PERMRING EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//* Step 4:                                                         *
//* Permitting access to the keyring                                *
//*     Owners of KEYRING need READ access                          *
//*         FTP and TN3270 PROCS are owned by Userid 'TCPIP'        *
//*         Other PROCS may have different owners                   *
//* Non-Owners of KEYRING need UPDATE access                        *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TCPIP) ACCESS(READ)   1
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(PAGENT) ACCESS(UPDATE) 2
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS01)  ACCESS(UPDATE)  3
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS02)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS03)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS04)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS05)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS06)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS07)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS08)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS09)  ACCESS(UPDATE)
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS10)  ACCESS(UPDATE)
/*
```

Figure 3-10 Permit TN3270 and FTPD access to the shared key ring

6. Protect the private key and permit the user IDs of each server to access it. The private key is represented by the facility named `IRR.DIGTCERT.GENCERT`. They all need CONTROL access.

Figure 3-11 demonstrates permitting access to the private key of the shared SITE certificate to the servers for FTP, TN3270, and Policy Agent, as well as to any other user IDs that need access to the private key facility.

```
//PERMKEY  JOB MSGCLASS=X,NOTIFY=&SYSUID
//PERMKEY EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*  Step 5:                                                        *
//*Permitting access to the private key of shared SITE cert in keyring*
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(TCPIP) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(PAGENT) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS01) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS02) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS03) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS04) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS05) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS06) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS07) ACCESS(CONTROL)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS08) ACCESS(CONTROL)
  SETROPTS RACLIST(FACILITY) REFRESH
/*
```

Figure 3-11 Grant access to the private key to the servers

7. Export the internal CA certificate to an MVS data set.

Figure 3-12 shows the RACDCERT EXPORT command being used to export the internal CA certificate to an MVS data set. The MVS data set will be sent via FTP in CERTB64 ASCII format to any client that needs to use that certificate to validate a server (in this case, a TN3270 client and an FTP client).

```
//EXPORT    JOB MSGCLASS=X,NOTIFY=&SYSUID
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Step 6:                                                     *
//*     Export the Self-signed Certificate Authority certificate    *
//*     from the RACF database in base-64 encoded format.  This is   *
//*     then FTP'd to the clients so that they can verify the server  *
//*     certificiates when passed in the SSL exchange.              *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT CERTAUTH EXPORT(LABEL('CS19 ITSO CA1')) -
    FORMAT(CERTB64) DSN('TCPIP.CS19.CACERT')
/*
```

Figure 3-12 Batch job to write the internal CA certificate to an MVS data set

Note that the export command used will not export the private key. Only the public key needs to be exported.

One way to reduce the number of file transfers to clients is for them to pick up their key databases from a LAN drive prepared on a server. Some clients dynamically allow the upload of the certificate during the handshake itself.

8. FTP the certificate exported in step 4 on page 72 to clients that will use it.

   This step is not illustrated, because any FTP client is able to perform this step. However, if the format is B64 (Base64), the FTP from z/OS to any other platform must be done in ASCII mode. This is because Base64 encoding is a character-based encoding, so the translation from EBCDIC to ASCII is necessary. If the exported certificate was in any other format (for example, DER), then a BINARY or IMAGE mode of transfer would be required.

Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can paste a B64 certificate from an MVS data set into a simple workstation editor and then save it to disk.

The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

9. At the client, the certificate received from step 5 on page 73 must be imported into the key database as a trusted certificate.

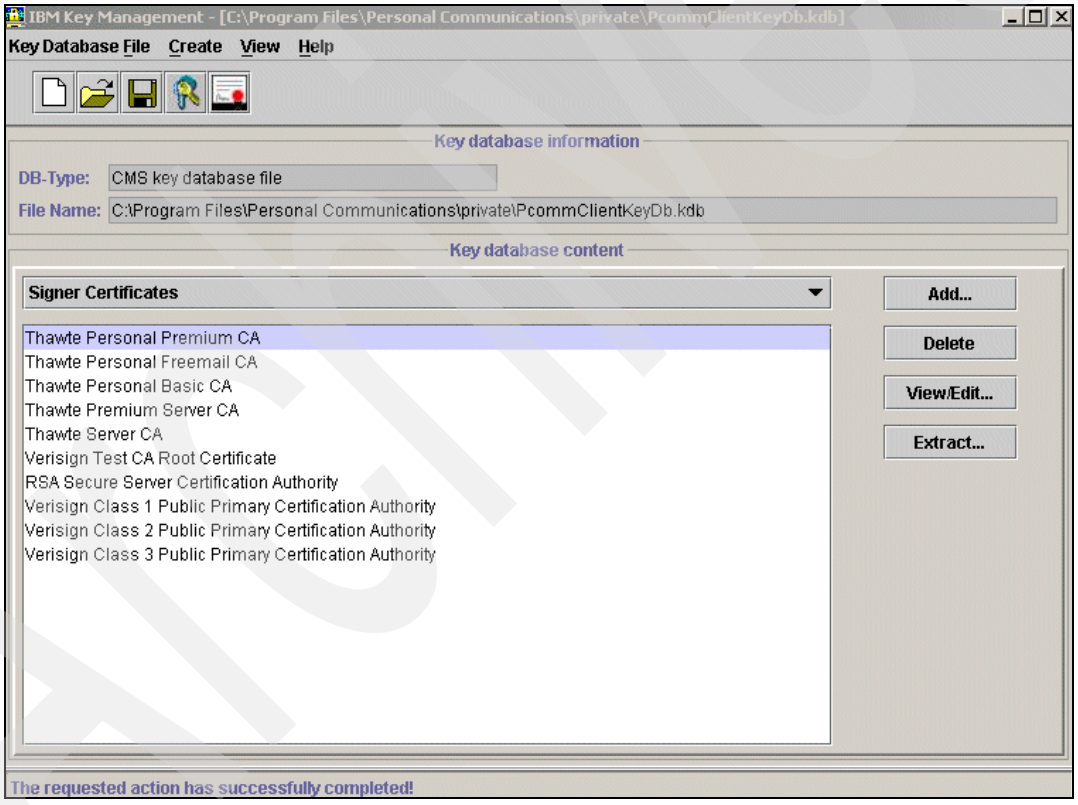


Figure 3-13 Personal Communications client certificate management window

Depending on the type of client, there are a number of different ways to do this. In the case of a Windows Personal Communications client (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Utilities folder. Open the key database by clicking the **Open** icon and entering the database password. This displays the window (after the key database file is opened) in Figure 3-13. Import the certificate using the **Add...** button.

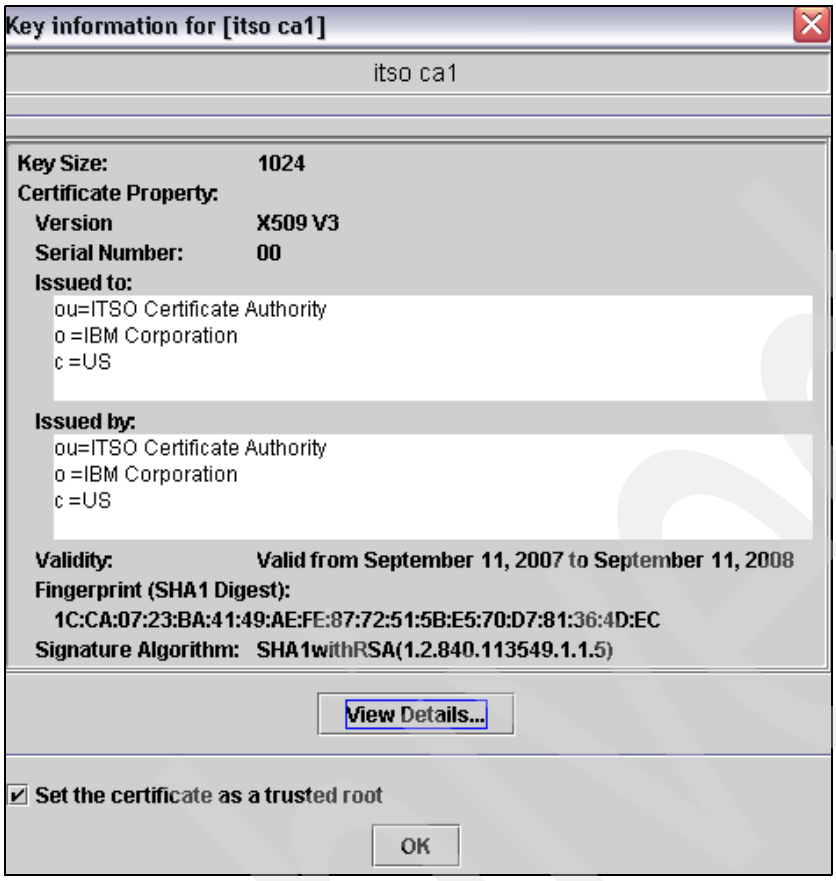


Figure 3-14 Personal Communications client: display of imported server certificate

When the certificate is added, the window shown in Figure 3-14 displays the detail from the certificate, showing the key size of 1024 (set by default in the RACDCERT GENCERT command), the certificate version, the `Issued To` name, the `Issued By` name (which is the same as the `Issued To` name), the valid date range, and the encryption algorithm to be used. Notice that the certificate must be marked as a trusted root, as indicated by the checkmark.

10. Test the client-to-server connection.

    As the example in Figure 3-15 on page 55 shows, a personal communications client was instructed to connect to the TN3270 server using SSL. In our case, we used the IBM Personal Communications client (PComm).

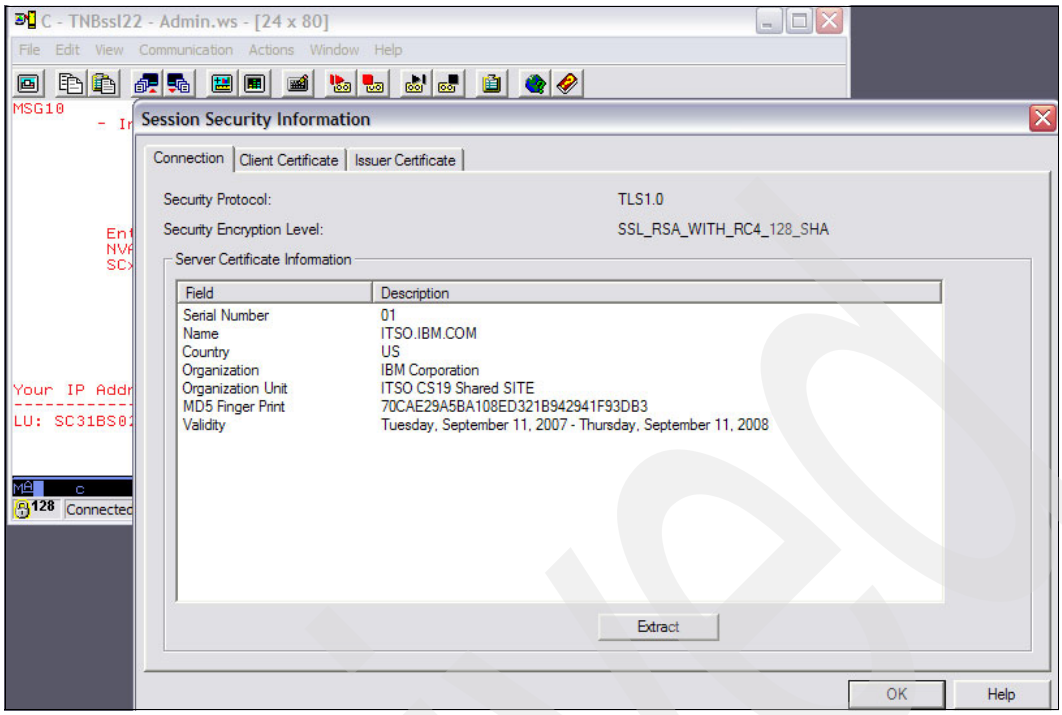


Figure 3-15 Personal Communications client: site certificate and signer's details

Figure 3-15 shows Personal Communications displaying the certificate (by clicking **Communication** -> **Security Information...**), which shows that the certificate subject is the original site certificate, and the certificate issuer is the internal CA set up in step 1 on page 71.

Note however, that the concept of a site certificate is not applicable to Personal Communications. As far as SSL/TLS is concerned, this certificate is a normal user or personal certificate sent out by a server during the handshake.

This discussion showed the generation of a site certificate to be shared by multiple servers and signed by the internal CA certificate. The internal CA certificate was exported to the client and placed in the client's key database as a trusted certificate. The procedure for any site certificate is similar.

To see how the individual servers are configured to make use of the site certificate and shared key ring, refer to their respective sections:

- Chapter 14, "Telnet security" on page 539
- Chapter 15, "Secure File Transfer Protocol" on page 579

3.4.3 Self-signed certificates

This section demonstrates how to use the TSO RACDCERT command and the UNIX **gskkyman** command to store and use self-signed certificates. Each example assumes the server is on z/OS and the client is not.

As mentioned earlier, self-signed certificates are not often used. However, since some organizations do require the use of self-signed certificates, we cover the topic here. It is recommended that a locally signed certificate, as detailed in 3.4.4, "Internal (local) Certificate Authority" on page 70, be used instead of a self-signed certificate. One reason that using a local CA is preferred over a self-signed certificate is because the local CA implementation

more closely resembles the well-known CA environment. Also, there is very little extra work involved in using a local CA as opposed to a self-signed certificate.

Self-signed server authentication, RACDCERT procedure

In this section, we illustrate how to generate and store a certificate for use by a TN3270 server. Once generated, the server certificate is placed in the server's RACF key ring. The public key and certificate are exported and placed in the client's key database as a trusted CA certificate.

There is no client authentication involved in this section. For an example of using client authentication, see "Self-signed client authentication, RACDCERT procedure" on page 59.

Here are the steps required to generate and store a certificate for use by a TN3270 server:

1. Generate a self-signed certificate for the server, as shown in Figure 3-16.

```
//CS081   JOB 'SET UP TN3270 CERT','CS08',CLASS=A,MSGCLASS=X
//SERVCRT  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* set up the TN3270 server certificate, and self-sign it.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
1 RACDCERT ID(TCPIP) GENCERT SUBJECTSDN(CN('ITSO.IBM.COM') -
        O('IBM CORPORATION')                               -
        OU('ITSO  TN3270 SERVER')                          -
        C('US'))                                           -
        WITHLABEL('TN3270 SERVER')
/*
```

Figure 3-16 Batch job to create a self-signed server certificate

The ID(TCPIP) parameter 1 associates the certificate being generated with the RACF user ID TCPIP. This is the user ID that the server in our example is running under. Yours will probably be different. For an explanation of the rest of the RACDCERT parameters, see 3.3.2, "Digital certificate field formats" on page 40.

Note that because there is no RACDCERT SIGNWITH parameter specified on the GENCERT command, the certificate will be digitally signed by the private key owned by the subject of the certificate. This is the definition of a self-signed (or root) certificate. It is good practice to make the common name (CN) the same as the host (DNS) domain name of the server.

2. Create a RACF key ring for the server.

Figure 3-17 shows the two steps necessary to create the key ring for the server.

```
//CSO81   JOB 'SET UP TN3270 CERT','CSO8',CLASS=A,MSGCLASS=X
//key ring  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add a new key ring to the TN3270 servers RACF ID (TCPIP), then....
//* Add TN3270 server certificate to the user 'TCPIP's key ring. the
//*   key ring name is from the TN3270 configuration statement as below
//*      'key ring SAF TN3270Ring'
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(TCPIP) ADDRING(TN3270RING) 1
  RACDCERT ID(TCPIP) CONNECT(ID(TCPIP)            - 2
                          LABEL('TN3270 SERVER') -
                          RING(TN3270RING)       - 3
                          DEFAULT                - 4
                          USAGE(PERSONAL))
/*
```

Figure 3-17 Batch job to add a key ring for the self-signed certificate

a. Create a new RACF key ring using the RACDCERT ADDRING command 1.

b. Then, as shown in 2, connect the self-signed server's certificate to the new key ring using the RACDCERT CONNECT command.

Note that the RING parameter 3 specifies the same ring name as you configured into the server. For the TN3270 server, this is specified on the TELNETPARMS KEYRING SAF *ringname* statement. For the FTP server, it is on the KEYRING statement in FTP.DATA.

The DEFAULT statement 4 is needed when using the native SSL/TLS capabilities of the FTP or TN3270 server. These servers will look for and send out (if it is found) the default certificate during the handshake. If there is no default certificate designated in the key ring, the handshake will fail.

When using AT-TLS, the default designation is not necessary, or can be overridden, by specifying CertificateLabel on the TTLSConnectionAdvancedParms statement.

> **Note:** It is possible to override the DEFAULT designation for the native FTP server only by passing the GSK_KEY_LABEL environment variable. The TN3270 server does not accept environment variables in this fashion, and therefore this method is not available.

3. Export the self-signed server certificate to an MVS database.

Figure 3-18 shows the RACDCERT EXPORT command being used to export the self-signed server certificate to an MVS data set. Note that the private key is not included in this export. A private key should never be exported if the certificate is only being exported as a CA certificate.

```
//CS081   JOB 'EXPORT SERVER CERT','CS08',CLASS=A,MSGCLASS=X
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Export the Self-signed Server certificate from the RACF database
//*   in base-64 encoded format. This is then FTP'd to the TN3270
//*      client so that it can verify the same certificate
//*      when passed in the SSL exchange.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(TCPIP) EXPORT(LABEL('TN3270 SERVER'))  -
           FORMAT(CERTB64) DSN('CS08.RACDCERT.TN32CERT')
/*
```

Figure 3-18 Batch job to write the internal CA certificate to an MVS data set

4. FTP the certificate exported to the MVS data set in step 3 on page 57 to the client that will use it.

   This step is not illustrated, because any FTP client will be able to perform this step. Note that in the example, the exported certificate in the MVS data set is in Base 64 format. Therefore, the FTP must perform EBCDIC-to-ASCII translation if the client is on an ASCII host.

   Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can paste a B64 certificate from an MVS data set into a simple workstation editor and then save it to disk.

   The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

   The MVS data set must be downloaded to any client that needs to use that certificate in order to validate the same certificate when presented by a server in an SSL exchange. Depending on the number of clients in an enterprise, this may result in a large number of transfers. One way to reduce the number of file transfers to clients is for all clients to pick up their key database from a LAN drive that has been prepared on a server.

5. At the client, the certificate received from step 4 must be imported into the key database as a trusted certificate.

   Depending on the type of client, there are a number of ways to do this. In the case of a Windows personal communications client, such as the IBM Personal Communications or PComm (a TN3270 client), select the Certificate Management or Certificate Wizard icon from the PComm Utilities folder. This displays the window (after the key database file is opened) shown in Figure 3-13 on page 53. Import the certificate by clicking the **Add** button. When the certificate is added, the window seen in Figure 3-19 on page 59 will show the detail display from the certificate. Note the key size of 1024 (set by default with the RACDCERT GENCERT command), the certificate version, and the `Issued To` name are the same as the `Issued By` name (this is a self-signed certificate).

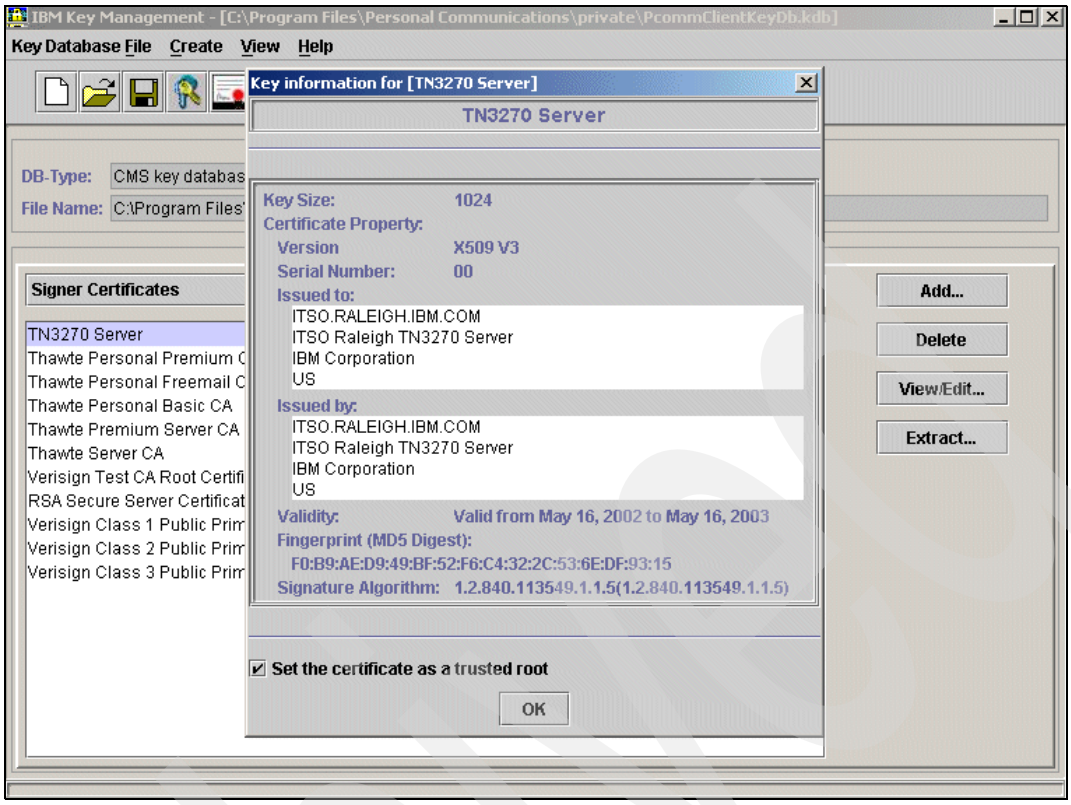


Figure 3-19 Personal Communications client: display of imported self-signed server certificate

6. Test the client-to-server connection.

   We activated the personal communications client to connect to the TN3270 server using SSL. If you want to know more about the SSL/TLS session, click **Communication** -> **Security Information...**.) See Figure 3-15 on page 55 for an example.

Although this section showed the certificate being generated for a TN3270 server, followed by an export to the client and placement in the client's key database as a trusted certificate, the procedure for any client/server is similar.

Self-signed client authentication, RACDCERT procedure

This section assumes that your server on z/OS has already been configured for server authentication, as outlined in "Self-signed server authentication, RACDCERT procedure" on page 56. Although RACDCERT is used on the z/OS endpoint, the certificate will be created at the client endpoint.

It is good practice to implement server authentication without client authentication during setup and testing. Server authentication is independent of, and happens prior to, client authentication. After server authentication is working correctly, you can then implement client authentication using the steps provided here.

A client public root CA certificate must be added to RACF and associated with the appropriate RACF user ID using the RACDCERT ID(servers-user ID) ADD... command.

The basic procedure to follow is described here:

1. Get the client certificate. For a self-signed certificate, this is usually generated at the client end. (Because client programs use different ways to generate a client certificate, this is a generic example.)

   In the case of Personal Communications, which provides a TN3270 client, use the Windows menu (click **Start** -> **Programs** -> **IBM Personal Communications** -> **Utilities** -> **Certificate Management**) to open the client's key database. Then click **Create** -> **New Self-signed Certificate** to generate the certificate. See Figure 3-20 for an example of a self-signed client certificate that has just been generated by Personal Communications into the client key database.

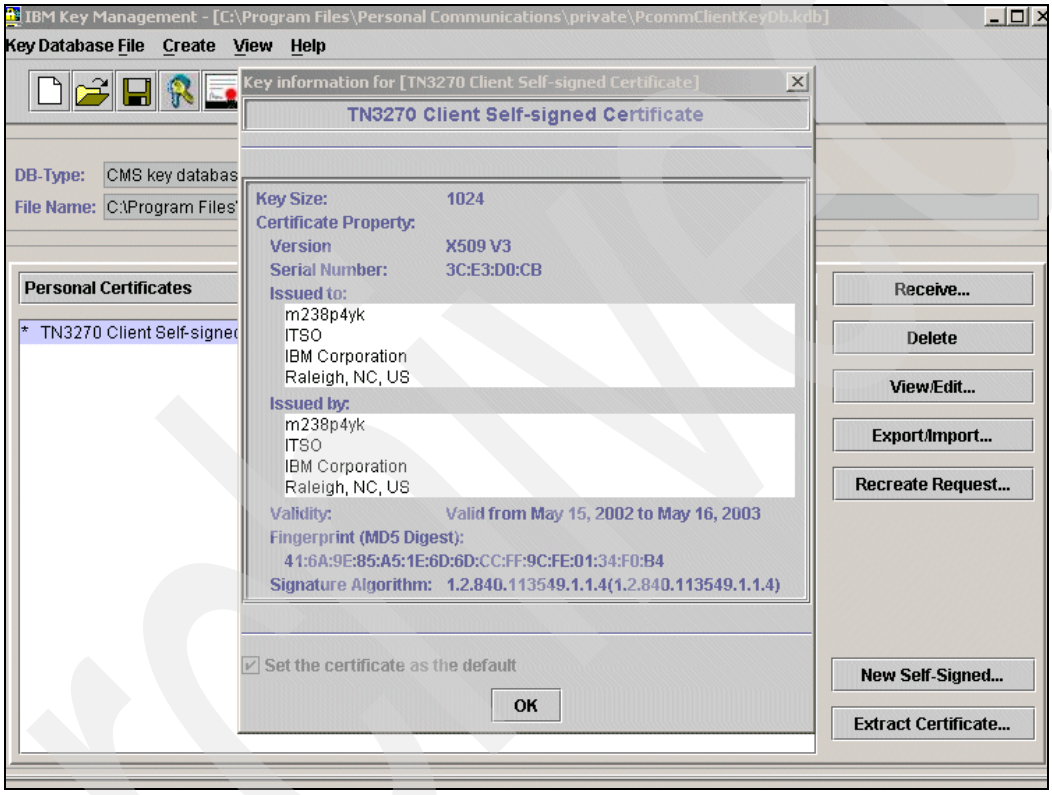


Figure 3-20 Personal Communications client: newly added self-signed client certificate

2. Export the certificate from the client's key database to a certificate file.

   Most certificate management utilities allow the export of a certificate in at least two formats. The most common is Base64-encoded ASCII, but there is also the binary DER format. The choice depends on what your certificate management utility at the server end can use as an import format.

   We used Base64 ASCII format. At the lower right side of the window shown in Figure 3-20 is the **Extract Certificate...** button. This is used to write a certificate to a data set. The window that is presented is shown in Figure 3-21 on page 61. Note that the Data Type field specifies Base64-encoded ASCII, and that the certificate will be written to the cert.arm file.

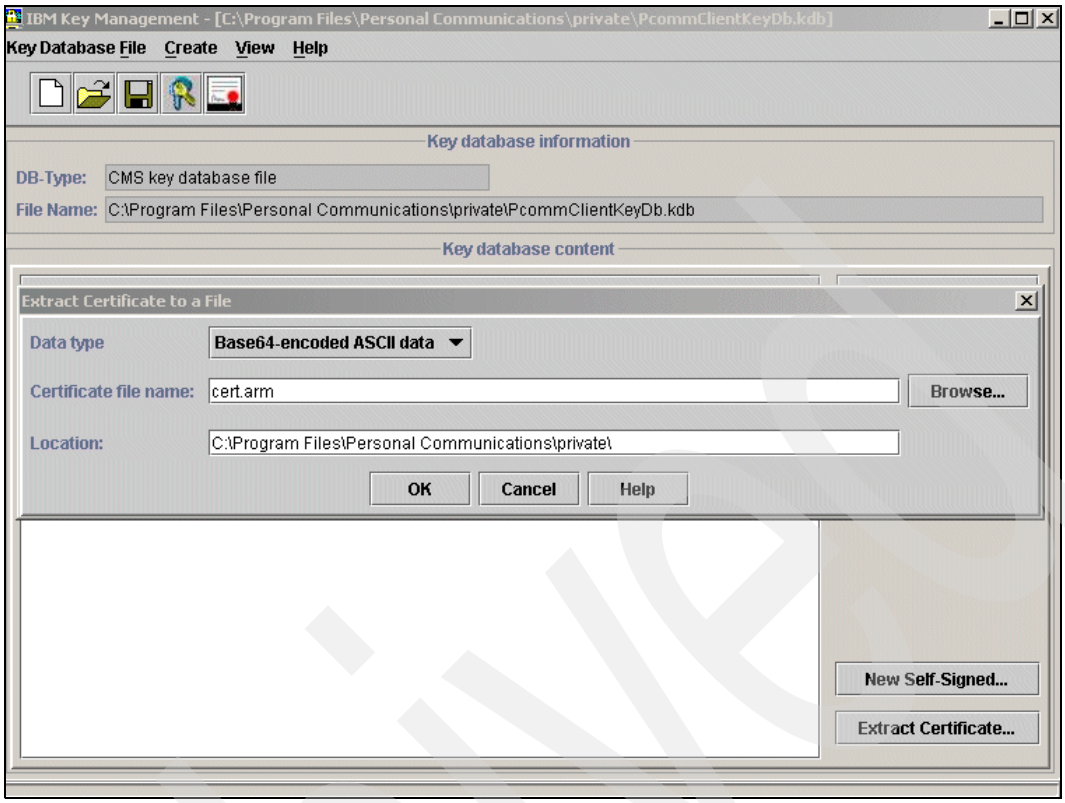


Figure 3-21 Exporting a client certificate to a file

3. FTP the certificate file from the client to the z/OS server side.

   This step is not illustrated. You simply need to perform an ASCII FTP on the Base64 encoded certificate file created in step 2 on page 60 (cert.arm, in this example) to an MVS data set at the server side as a text file. The MVS data set must have variable blocking in order for RACF to recognize it.

   Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can paste a B64 certificate from an MVS data set into a simple workstation editor, and then save it to disk.

   The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

4. Add the client certificate into the RACF database and associate it with a user.

   Figure 3-22 on page 62 shows the batch job used to add the client certificate to the RACF database and associate it with the RACF user ID CS08 (the owner of the client certificate) using the `ID()` parameter. The `RACDCERT ADD` command specifies the MVS data set name that contains the client certificate; this was the data set name that was created by the FTP in step 3. The certificate is set to trusted status, which means that any certificate that is signed by this certificate should pass authentication.

   This operation would also work if the certificate is imported as a CA certificate using `CERTAUTH` instead of `ID()`. This is because this certificate is going to be used to authenticate the personal certificate that this client will send out during the handshake.

```
//CSO81   JOB 'IMPORT SERVER CERT','CSO8',CLASS=A,MSGCLASS=X
//CLIENT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Import the Self-signed Client certificate into the RACF database
//*   This was FTP'd from the workstation TN3270 client, that
//*      generated it with the local PCOMM utility. It must be
//*      imported as a trusted CA certificate
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(CSO8) ADD('CSO8.RACDCERT.CLIENT.CERT') -
               TRUST WITHLABEL('TN3270 CLIENT CERTIFICATE')
/*
```

Figure 3-22 Batch job to add a client certificate into the RACF database as TRUSTED

5. Connect the client public CA certificate to the server's RACF key ring.

   Figure 3-23 shows the batch job used to connect the client certificate, added in step 4 on page 61, to the server's RACF key database. In the `RACDCERT CONNECT` statement, the key ring name is whatever is coded in the server's configuration file (in TN3270's case, it is specified on the `KEYRING SAF` statement). This assumes that the key ring is already set up (probably because you have the server's certificate there already). If the ring is not set up, you must create it before this job is run. See Figure 3-9 on page 50 for an illustration of the RACDCERT ADDRING command.

```
//CSO81   JOB 'CONNECT SERVER CERT','CSO8',CLASS=A,MSGCLASS=X
//CLIENT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Connect the Self-signed Client certificate into the RACF database
//*   This was FTP's from the workstation TN3270 client, that
//*      generated it with the local PCOMM utility. It must be
//*      imported as a trusted CA certificate
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
RACDCERT ID(TCPIP) CONNECT(ID(CSO8)             -
                        LABEL('TN3270 CLIENT CERTIFICATE') -
                        RING(TN3270RING)       -
                        USAGE(PERSONAL))
/*
```

Figure 3-23 Batch job to import client certificate into RACF

The connection can be made after these steps are completed and both the client and server are configured with the appropriate parameters for client authentication.

Self-signed server authentication, gskkyman procedure

The following steps demonstrate how to use the **`gskkyman`** command to set up a self-signed certificate to be used for server authentication only.

It is assumed that you have set up a key database and produced a stash file for the server as discussed in 3.3.4, “Using the gskkyman command” on page 44; that you have set the correct UNIX permissions so that the server can read the files; and that the database is named matt.kdb. After the database is created, generate a self-signed certificate into it, and set it as the default certificate. The self-signed certificate must then be exported to the client workstation, where it is imported into the client’s key database as a trusted CA certificate.

Note: If the server supports AT-TLS, the default certificate is not the only choice for sending to the client. Another certificate (non-default) can be specified within the AT-TLS configuration for this server, and sent to the client to identify this server.

1. Open the key database using **gskkyman** by selecting option 2 from the database menu. Figure 3-24 shows **gskkyman** being used to open the existing database in preparation for generating the certificate.

```
 Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

  11 - Create new token
  12 - Delete token
  13 - Manage token
  14 - Manage token from list of tokens

   0 - Exit program

Enter option number: 2
Enter key database name (press ENTER to return to menu): matt.kdb
Enter database password (press ENTER to return to menu):

Database: /u/cs08/matt.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Figure 3-24 Opening the key database

After the key database is opened, you are presented with another list of options. Option 6 creates a self-signed certificate by prompting for details. Selecting option 6, as shown in Figure 3-25, continues the process of generating the self-signed certificate.

2. Create the self-signed certificate and export it to a z/OS UNIX file.

```
Enter option number (press ENTER to return to previous menu): 6

       Certificate Type

   1 - CA certificate with 1024-bit RSA key
   2 - CA certificate with 2048-bit RSA key
   3 - CA certificate with 4096-bit RSA key
   4 - CA certificate with 1024-bit DSA key
   5 - User or server certificate with 1024-bit RSA key
   6 - User or server certificate with 2048-bit RSA key
   7 - User or server certificate with 4096-bit RSA key
   8 - User or server certificate with 1024-bit DSA key

Select certificate type (press ENTER to return to menu): 6
Enter label (press ENTER to return to menu): TN3270 gskkyman certificate
Enter subject name for certificate
  Common name (required): ITSO.IBM.COM
  Organizational unit (optional): ITSO TN3270 Server
  Organization (required): IBM
  City/Locality (optional): Poughkeepsie
  State/Province (optional): NY
  Country/Region (2 characters - required): US
Enter number of days certificate will be valid (default 365): 3650

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate created.
```

Figure 3-25 Generating a self-signed certificate using gskkyman

Figure 3-25 shows the dialog between the user and **gskkyman** to set up a self-signed certificate. The first choice to be made is the key size (we selected option 6). At present, anything over 512 bits (which is not even an option with gskkyman) is considered to be reasonably secure.

In this example, an RSA key (the most commonly used, at present) of 2048 bits was chosen. A label for the certificate is entered as is the common name, organizational unit and name, city, state and country. The lifetime of the certificate was entered as 3650 days or 10 years.

Next, we made this certificate the default for this key database by entering zero (0) and then pressing Enter to return to the Key Management Menu.

Figure 3-26 on page 65 illustrates the command sequence. We began by selecting **1** to manage keys and certificates. We then selected our certificate from the list. After selecting this certificate, we can see in **1** the option for setting it as the default for the ring (option 3).

```
 Key Management Menu

       Database: /u/cs08/matt.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 1

Key and Certificate List

       Database: /u/cs08/matt.kdb

   1 - TN3270 gskkyman certificate

   0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list): 1

Key and Certificate Menu

       Label: TN3270 gskkyman certificate

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default 1
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file 2
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 3

Default key set.
```

Figure 3-26 Setting the newly created certificate as the default

In addition, in Figure 3-26, at 2 we can see that option 6 allows us to export this certificate to a file. (Note: Do not use option 7 - the private key is only required for the endpoint that wishes to identify itself. In a server authentication-only context, the client will never need a private key.)

After selecting option 6, a prompt for export format was received. PKCS #7 is usually a good format for being acceptable to any platform. Either Base64 encoding or binary is acceptable.

This exported certificate is what is transferred to the client for importation into the client's key database. In this example, the exported file is called matt.cer. The contents of the exported file are shown in Figure 3-27.

Note: If the server supports AT-TLS, the default certificate is not the only choice for sending to the client. Another certificate (non-default) can be specified within the AT-TLS configuration for this server, and sent to the client to identify this server.

```
>cat matt.cer
-----BEGIN CERTIFICATE-----
MIID6gYJKoZIhvcNAQcCoIID2zCCA9cCAQExADALBgkqhkiG9w0BBwGgggO/MIID
uzCCAqOgAwIBAgIIRvv/9wAEmyEwDQYJKoZIhvcNAQEFBQAwczELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAk5DMRUwEwYDVQQHEwxQb3VnaGtlZXBzaWUxDDAKBgNVBAoT
A0lCTTEbMBkGA1UECxMSSVRTTyBUTjMyNzAgU2VydmVyMRUwEwYDVQQDEwxJVFNP
LklCTS5DTO0wHhcNMDcwOTI3MTkwOTQzWhcNMTcwOTI0MTkwOTQzWjBzMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCTkMxFTATBgNVBAcTDFBvdWdoa2VlcHNpZTEMMAoG
A1UEChMDSUJNMRswGQYDVQQLExJJVFNPIFROMzI3MCBTZXJ2ZXIxFTATBgNVBAMT
DElUUO8uSUJNLkNPTTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKYM
AfAaVB/eAG3Z7iw8gEnn8epssKm5PlibWVXvYm81WzJYnbbI+Gv+bU52xBMVxIrb
N3IQOBcAzayFLKLugfhWczWzcsVOX2Ea62q4w3kgdhvjhF+vHq69NhBKdphjypCU
bJDMwKimU+CVIrMpco/FJ7IjmA5h971K+OmlxxGnMEu/EsiOTwV/cG1Oy4ylOHU4
CU39zrrvQD3seDM/SNgkZd9edAofqD+t+bJ2ZdvUSOwPSxlFmKyqqJbZkt6rKQdq
OmSJRLgJSVsRQjX9OA5Ah+7toPAgv9pUGNA+EwUwMEjnflCadhc4caCbwbuKMryD
6NmbD2/yeDpsYd+hsgsCAwEAAaNTMFEwHQYDVR0OBBYEFEcKBLLH71Slb+dHGgOb
OgSChyyJMB8GA1UdIwQYMBaAFEcKBLLH71Slb+dHGgObOgSChyyJMA8GA1UdEwEB
/wQFMAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAIfJWteLlw2SkFWOwzBs5B5cmCm9
jNZhmbtzxJZkw9l2pAaXyAWLOmmi9J1aVlRNRpFO4ru2VMBKacICsOPlrfAlwwKn
3DliBfHRplp7N+UO/wxcwxZC59OJyDzUedg+pCOyYE+GEQW2ykw5VGSFJTGtOAMq
W1z7vp6EJxqxpQ93eKDQhuXL6OqOeow9NR/PY9yWtku8t/XatObFaO1BkzzRlqeG
U6mEOOgXIirbmfZButepnr/PBWuMybZGar3NIvljDnn8irGU+1lpOki8Xs+UL8e9
SAGw+UlZp4zXouoieMHf+XE3dvXvyhCZBal8qkpSfwkelIHm5GYDAt9q4pwxAA==
-----END CERTIFICATE-----
```

Figure 3-27 An exported certificate from the gskkyman utility

The z/OS UNIX **cat** command (as shown in Figure 3-27) will show the contents of an EBCDIC file on an EBCDIC host. You will need to FTP this file in ASCII if you want it to be usable on an ASCII host. All Base64 encoded certificates will contain the `BEGIN CERTIFICATE` and `END CERTIFICATE` headers (there may also be others). If you edit or display a Base64 certificate and you do not see both of these headers, it means that the certificate has been corrupted somehow.

3. FTP the file exported in step 2 on page 64 to the client that will be using it.

Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can paste a Base64 certificate from an MVS data set into a simple workstation editor, and then save it to disk.

The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid additional unwanted formatting.

4. Import the certificate into the client's key database.

   At the client, the certificate received from step 3 on page 66 must be imported into the key database as a trusted certificate. Depending on the type of client, there are a number of different ways to do this.

   In the case of a Windows Personal Communications client (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Utilities folder. This displays the window (after the key database file is opened) shown in Figure 3-13 on page 53.

   Now import the certificate by clicking the **Add** button. When the certificate is added, the window in Figure 3-28 (the detail display from the certificate) will appear. It shows the key size of 1024 (set by **gskkyman** when creating the certificate in Figure 3-25 on page 64), the certificate version, and the `Issued To` name are the same as the `Issued By` name (this is a self-signed certificate).

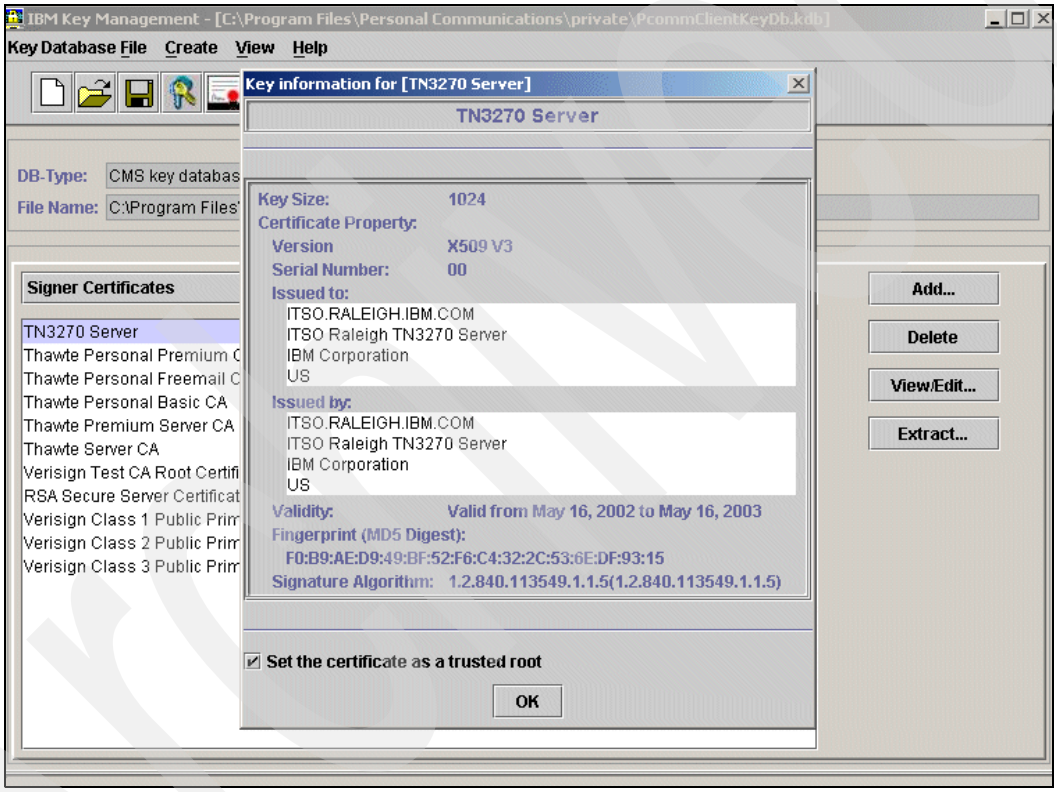


Figure 3-28 Personal Communications client: display of imported self-signed server certificate

Now the client should be able to connect to the server. When the server passes its certificate to the client during the SSL exchange, the client will be able to validate it using the same certificate that is now stored in the client's key database as a CA certificate.

Self-signed client authentication, gskkyman procedure

It is assumed that you have set up a key database (`matt.kdb`) and the server's certificate is in the key database.

The steps to implement client authentication in a **gskkyman** environment are similar as in the RACDCERT environment, except that the certificates are stored in a z/OS UNIX key database rather than the RACF database.

These steps are:

1. Get the client certificate. For a self-signed certificate, this is often generated at the client end. (Because different client programs have different ways to generate a client certificate, this should be thought of as a generic example.)

   In the case of IBM Personal Communications, which provides a TN3270 client, use the Windows menu (click **Start** -> **Programs** -> **IBM Personal Communications** -> **Utilities** -> **Certificate Management**) to open the client's key database. Then click **Create** -> **New Self-signed Certificate** to generate the certificate. See Figure 3-29 for an example of a self-signed client certificate that has just been generated by IBM Personal Communications into the client key database.

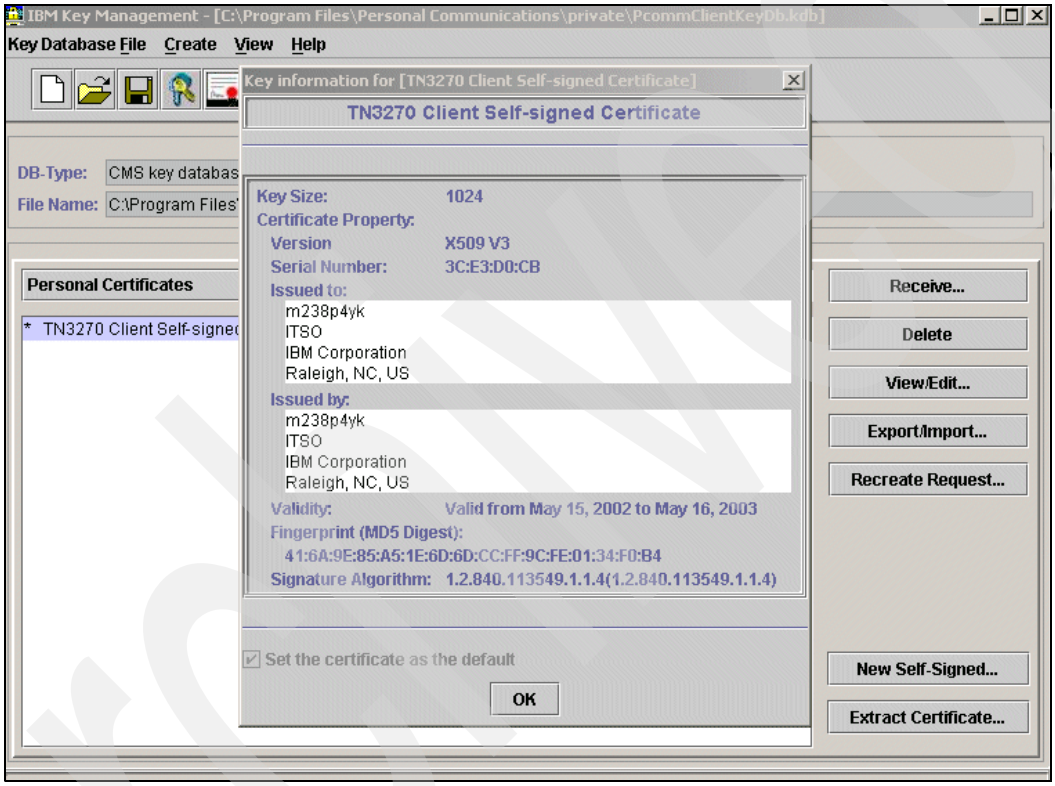


Figure 3-29 Personal Communications client: newly added self-signed client certificate

2. Export the client certificate to a file.

   We used Base64 ASCII format. At the lower right side of the window shown in Figure 3-29 is the **Extract Certificate...** button. This is used to write a certificate to a file. The window that is presented is shown in Figure 3-30 on page 69. Note that the `Data Type` field specifies Base64-encoded ASCII, and that the certificate will be written to the cert.arm file.

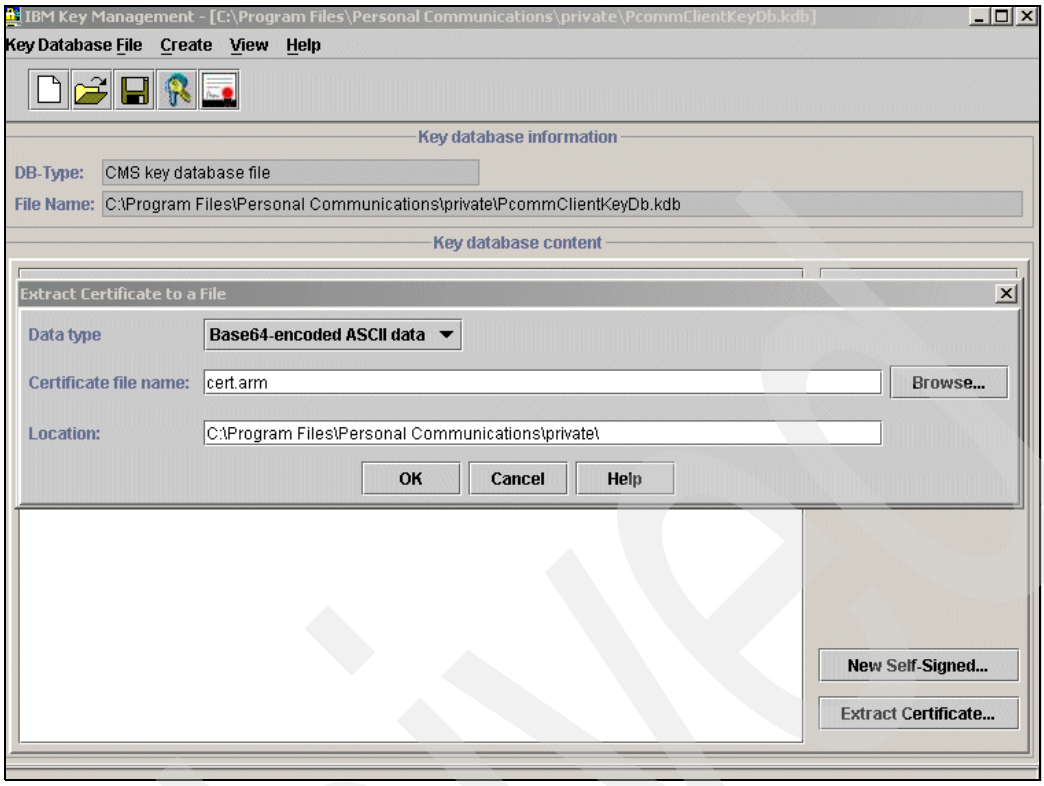


Figure 3-30 Exporting a client certificate to a file

3. FTP the certificate file from the client to the z/OS server side.

   To do this, you only need to FTP the certificate file created in step 2 on page 68 (cert.arm in this example) to a z/OS UNIX file at the server side as a text file.

   Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can copy a Base64 certificate from an MVS data set and paste it into a simple workstation editor, and then save it to disk.

   The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

4. Add the client certificate into the server's z/OS UNIX key database using **`gskkyman`**.

   Refer to Figure 3-24 on page 63 for instructions about opening an existing database using **`gskkyman`**.

   When the key database is opened, you are presented with a list of options. Choose option **7 Import a certificate** to receive a certificate and store it as a trusted CA certificate.

```
Database: /u/cs08/matt.kdb

  1 - Manage keys and certificates
  2 - Manage certificates
  3 - Manage certificate requests
  4 - Create new certificate request
  5 - Receive requested certificate or a renewal certificate
  6 - Create a self-signed certificate
  7 - Import a certificate
  8 - Import a certificate and a private key
  9 - Show the default key
 10 - Store database password
 11 - Show database record length

  0 - Exit program

Enter option number (press ENTER to return to previous menu): 7
Enter import file name (press ENTER to return to menu): telnetcert.arm
Enter label (press ENTER to return to menu): TN3270 Client Cert for Workstation1

Certificate imported.
```

Figure 3-31 Adding a self-signed client certificate as a CA certificate using gskkyman

Figure 3-31 shows the dialog between the user and **gskkyman** to add the client certificate. The client certificate file in the example is named `telnetcert.arm`; it was placed there in the FTP transfer in step 3 on page 69.

Now the client should be able to connect to the server. The server will be able to validate the client certificate with its copy of the client's certificate in the server's key database.

3.4.4 Internal (local) Certificate Authority

One possibility in setting up your certificate management scheme is to set up as a local CA. As a local CA, you will be signing digital certificates for other entities, and anybody who uses the certificates that you sign will have to have a copy of your root CA certificate in their key databases.

You might choose to be a CA if you have multiple SSL/TLS-enabled servers in your system. When you have more than one server with its own self-signed certificate, each certificate must be exported to the clients that will use them. Therefore, if you had an FTP server, a TN3270 server, and an LDAP server, each using self-signed certificates and all being used from one workstation, that workstation will need all three certificates in its key database. With an internal CA, you can sign each of the server's certificates, and export the one certificate (the internal root CA certificate) to the clients. (Note that this assumes a client's key database can be shared between client programs. This is sometimes not the case, but a saving can still be made in that only the one key needs to be distributed.)

In this section, we explain how to set up an internal CA, and create and sign a server certificate with that CA certificate. You will see that the process is similar to the self-signing process described in 3.4.3, "Self-signed certificates" on page 55, except that the certificate being distributed to the clients is the root CA certificate, not the user certificate.

Internal-CA signed server certificate RACDCERT procedure

This procedure is similar for any z/OS server. In this example, we are producing a certificate for use by a TN3270 server. This certificate is needed by TN3270 clients using SSL/TLS. In order for the client to validate the server certificate, the internal CA certificate will be needed at the client. The following steps demonstrate how to do this.

1. Create a self-signed certificate for the local (internal) CA, as shown in Figure 3-32.

```
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add the top-level self-signed certificate for the certificate
//*   authority (ourselves)
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  racdcert certauth gencert                          -
     subjectsdn( o('IBM Corporation')                -
                 ou('ITSO Certificate Authority')    -
                  C('US'))                           -
      WITHLABEL('ITSO Certificate Authority')
/*
```

Figure 3-32 Batch job to create internal CA certificate in RACF database

2. Generate a certificate for the server, as shown in Figure 3-33.

```
//SERVCRT  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* set up the TN3270 server certificate, and sign it with the
//*   self-signed certificate-authority certificate set up previously
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(TCPIP) GENCERT SUBJECTSDN(CN('ITSO.IBM.COM') -
        O('IBM Corporation')                                -
       OU('ITSO  TN3270 Server')                      -
        C('US'))                                            -
 WITHLABEL('TN3270 Server')                                 -
  SIGNWITH(CERTAUTH LABEL('ITSO Certificate Authority')) 1
/*
```

Figure 3-33 Batch job to create server certificate and sign with internal CA certificate

Note that the RACDCERT SIGNWITH command **1** specifies the label of the internal CA certificate we set up in step 1. This indicates that the server certificate should be digitally signed with the internal CA's private key. The `ID(TCPIP)` parameter is used in this example because the TN3270 server, for whom the certificate is being generated, is associated with RACF user ID `TCPIP`. Again, it is good practice to ensure that the common name (CN) is the same as the host or domain name of the server.

3. Create a RACF key ring for the server.

```
//keyring  EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Add a new key ring to the TN3270 servers RACF ID (TCPIP), then....
//* Add TN3270 server certificate to the user 'TCPIP's key ring. the
//*   key ring name is from the TN3270 configuration statement as below
//*      'key ring SAF TN3270Ring'
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT ID(TCPIP) ADDRING(TN3270Ring)
  RACDCERT ID(TCPIP) CONNECT(CERTAUTH              -
                          LABEL('ITSO Certificate Authority') -
                          RING(TN3270Ring)       -
                          USAGE(CERTAUTH))
  RACDCERT ID(TCPIP) CONNECT(ID(TCPIP)               -
                          LABEL('TN3270 Server') -
                          RING(TN3270Ring)       -
                          DEFAULT                -
                          USAGE(PERSONAL))
/*
```

Figure 3-34 Batch job to add a key ring

Figure 3-34 shows the three steps necessary to create the key ring for the server:

a. Create a new RACF key ring using the RACDCERT ADDRING command.

b. Connect the internal root CA certificate to the new key ring using the RACDCERT CONNECT command.

c. Connect the server's certificate (which was signed by the internal root CA certificate) to the new key ring using the RACDCERT CONNECT command.

4. Export the internal CA certificate to an MVS data set.

```
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*
//* Export the Self-signed Certificate Authority certificate from the
//*   RACF database in base-64 encoded format. This is then FTP'd to
//*      the TN3270 client so that it can verify the TN3270 server's
//*      certificate when passed in the SSL exchange.
//*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  RACDCERT CERTAUTH EXPORT(label('ITSO Certificate Authority')) -
             FORMAT(CERTB64) DSN('CS08.RACDCERT.SERVCERT')
/*
```

Figure 3-35 Batch job to write the internal CA certificate to an MVS data set

Figure 3-35 shows the RACDCERT EXPORT command being used to export the internal CA certificate to an MVS data set. The MVS data set will be ASCII, sent via FTP to any client that needs to use that certificate to validate a server (in this case, a TN3270 client). One way to reduce the number of file transfers to clients is for them to pick up their key databases from a LAN drive prepared on a server.

5. FTP the certificate exported in step 4 on page 72 to the client that will use it.

   This step is not illustrated, because any FTP client will be able to perform this step. If the certificate is not exported as Base64, however, make sure you use a binary mode of transfer instead of text/ASCII.

   Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can copy a Base64 certificate from an MVS data set and paste it into a simple workstation editor, and then save it to disk.

   The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

6. At the client, the certificate received from step 5 must be imported into the key database as a trusted certificate.

   Depending on the type of client, there are a number of different ways to do this. In the case of a Windows client such as IBM PComm (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Utilities folder. This displays the window (after the key database file is opened) that was shown previously in Figure 3-13 on page 53. Import the certificate using the **Add** button.

   When the certificate is added, the window shown in Figure 3-36 shows the detailed display from the certificate. You see that the key size of 1024 (set by default in the RACDCERT GENCERT command), the certificate version, and the `Issued to` information are the same as the `Issued by` information (all root CA certificates are self-signed).

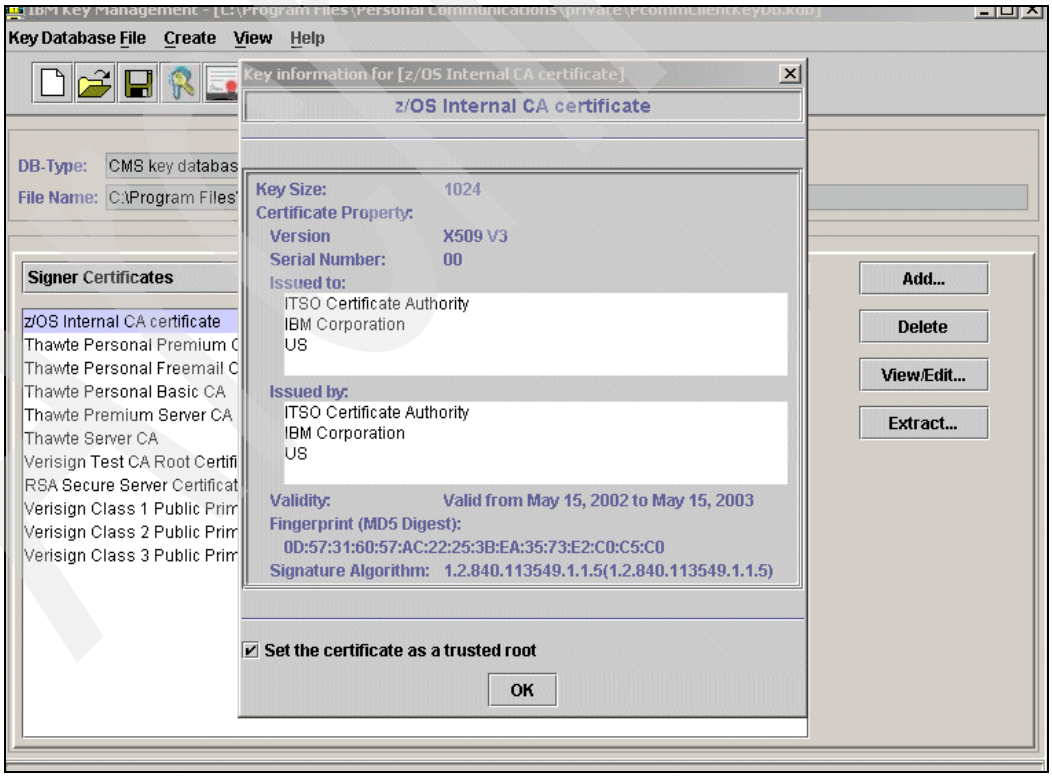


Figure 3-36 The IBM Personal Communications client: display of imported server certificate

7. Test the client-to-server connection.

   The communications client was instructed to connect to the TN3270 server using SSL. To verify the SSL/TLS characteristics of the session using PComm, select **Communication -> Security Information...** (you should see something similar to what is shown in Figure 3-15 on page 55.

This discussion showed certificates being generated for a TN3270 server, and the internal CA certificate being exported to the client, then placed in the client's key database as a trusted certificate. The procedure for any client/server is similar.

3.4.5 External (well-known) Certificate Authority

This section demonstrates the use of the TSO RACDCERT command and the UNIX **gskkyman** command to store and use well-known CA-signed certificates. It is assumed that the server is on z/OS and the client is not.

The following step-by-step examples can be used to create a z/OS UNIX key database or RACF key ring for the IBM HTTP Server for z/OS, the TN3270 server, or other servers that are SSL/TLS-enabled. The SSL/TLS support can be provided internally by the application, or it can be provided through AT-TLS configuration through the Policy Agent. The TN3270 and FTP servers on z/OS provide support for both internal SSL/TLS and external TTLS through the AT-TLS Policy Agent configuration.

Note: Even though the TN3270 and FTP servers on z/OS provide support for both internal SSL/TLS and external TTLS through the AT-TLS Policy Agent configuration, implementation of TTLS through AT-TLS is now the preferred method for these two servers. AT-TLS provides more functionality and flexibility.

External CA-signed server certificate RACDCERT procedure

The steps required to implement the SSL environment for the IBM HTTP Server with a server certificate signed by a public CA are described here. A similar procedure can be used for other SSL-enabled application servers. The assumption in the examples is that the Web server's RACF user ID is WEBSRV.

1. Generate a self-signed certificate. A self-signed certificate is required as a temporary, placeholder user certificate that will eventually be signed by the external CA.

   We use the following certificate as a base for the certificate request we will be creating:

```
RACDCERT  ID(WEBSRV)  GENCERT
      SUBJECTSDN(CN('itso.ibm.com')
            O('IBM Corporation')
            OU('ITSO  Webserver')
            C('US'))
      WITHLABEL('Web Server Certificate')
```

   Because some clients will check for a match between the CN and the DNS host name of the server, you can set the CN to be 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 named CS07.WEBSERV.GENREQ. Use this command:

```
RACDCERT  ID(WEBSRV)  GENCERT
      GENREQ(LABEL('Web Server Certificate'))
      DSN('CS07.WEBSERV.GENREQ')
```

This certificate request must be sent to the CA. 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 (and now signed by the well-known CA) personal certificate into a variable blocked MVS data set.

   The CA usually returns the certificate using e-mail or similar means. The certificate should be in Base64-encoded text format. Transfer the certificate received from the CA into a VB MVS data set named, for instance, CS07.WEBSERV.CERT.

   Tip: Base64-encoded certificates are composed entirely of text data. This means they can be moved around by a simple cut and paste. For example, you can copy a Base64 certificate from an MVS data set and paste it into a simple workstation editor, and then save it to disk.

   The reverse is also true: you can copy from a text editor on the workstation and paste it into a VB MVS data set. Make sure you use a simple workstation editor, to avoid unwanted additional formatting.

4. If the CA vendor requires any intermediate certificates, you would add them to the data base at this time. You may need to go to the CA vendor's Web site to pick up any intermediates, or they may have been sent to you along with your personal certificate. The following command can be used to add an intermediate CA certificate.

   ```
   RACDCERT CERTAUTH ADD('CSO7.intermediate.cert') TRUST WITHLABEL('CA
   Intermediate')
   ```

   Note that the intermediates function as CA (CERTAUTH) certificates.

5. Replace the self-signed certificate with the personal certificate received from and signed by the CA.

   ```
   RACDCERT  ID(WEBSRV)
         ADD('CSO7.WEBSERV.CERT')
         WITHLABEL('Web Server Certificate')
   ```

   Use the same values for `ID()` and `WITHLABEL` for the RACDCERT ADD command that you used with the RACDCERT GENCERT in step 2. This is how you tell RACF that you are replacing the original self-signed certificate with the CA-signed certificate.

6. Create a key ring for the server, if it does not already exist. If it already exists, bypass this step.

   Key ring names become 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 key ring names, a single asterisk (*) is not allowed.

   ```
   RACDCERT ID(WEBSRV) ADDRING(WEBSERVER)
   ```

7. Connect the certificate to the key ring. For native FTP and TN3270 SSL/TLS, make sure you set the certificate as the default.

   In our case, we can now create the connection between the digital certificate and the key ring using the RACDCERT CONNECT command and associating it with the HTTP started task user ID.

   ```
   RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Web Server Certificate')
            RING(WEBSERVER) DEFAULT USAGE(PERSONAL))
   ```

Note: If the server supports AT-TLS, the default certificate is not the only choice for sending to the client. Another certificate (non-default) can be specified within the AT-TLS configuration for this server, and sent to the client to identify this server.

8. Connect any intermediates, if necessary:

```
RACDCERT ID(WEBSRV) CONNECT(ID(CERTAUTH LABEL('CA Intermediate')
         RING(WEBSERVER)
```

Because we implemented a user certificate that is signed by a well-known CA, the root CA certificate will already be in the browser clients' key rings. There is no need to do any updating to the key repositories of any of the clients.

External CA-signed server certificate, gskkyman procedure

The following step-by-step example shows how to create a key database in z/OS UNIX and how to create certificate requests for external CAs for the IBM HTTP Server for z/OS. A similar procedure may be used for other server applications, including the TN3270 server, the policy agent and AT-TLS.

In our test environment, we elected to have our certificate issued by the public CA company, VeriSign. In the following discussion, we show how we created the certificate request for our server certificate, and how we received it into the key database.

It is assumed that you have set up a key database and produced a stash file for the server, as discussed in 3.3.4, "Using the gskkyman command" on page 44; that you have set the correct UNIX permission bits so that the server can read the files; and that the database is named /u/takada/sslkey/server.kdb.

1. Create a public-private key pair and certificate request.

   Figure 3-37 on page 77 shows the menu panel of the **gskkyman** utility. To create a public-private key pair and a certificate request, select **4 - Create new certificate request** from this panel.

```
          Database: /u/takada/sslkey/server.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 4 1

Certificate Type

   1 - Certificate with 1024-bit RSA key
   2 - Certificate with 2048-bit RSA key
   3 - Certificate with 4096-bit RSA key
   4 - Certificate with 1024-bit DSA key

Enter certificate type (press ENTER to return to menu): 1 2

Enter request file name (press ENTER to return to menu): server.arm 3
Enter label (press ENTER to return to menu): ITSO  Webserver Cert 4
Enter subject name for certificate 5
  Common name (required): mvs03c.itso.ral.ibm.com
  Organizational unit (optional): ITSO
  Organization (required): IBM Corp.
  City/Locality (optional): Research Triangle Park
  State/Province (optional): North Carolina
  Country/Region (2 characters - required): US

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate request created.
```

Figure 3-37 Create a certificate request

1 Select option **4- Create new certificate request** to create a certificate request.

2 Select the key size that you want. Option 1 (1024) is still considered secure at the time of writing and was selected.

3 Specify a file name for the certificate request. Later you have to send the contents of this file to the CA vendor you selected.

4 Enter a label related to this key and certificate.

5 Enter the certificate subject name fields. Common Name should be your server's host name.

Some Web browsers perform an extra check of the server's identity by comparing the host name in the URL with the host name in the Common Name attribute in the certificate. If they do not match, the browser could issue a warning or deny the connection. Somewhere within the browser's preferences, there should be the ability to turn off this DNS-to-Common Name comparison. As mentioned, this comparison is not part of the SSL/TLS handshake.

2. Request the certificate from the CA.

   After step 1 on page 76, you have three files in addition to the key database:

   – A certificate request file (*.arm)
   – A stash file (*.sth)
   – A request file (*.rdb)

   Figure 3-38 shows the contents of the certificate request file. This file must be sent to the CA vendor to be signed. We sent this certificate request to VeriSign.

```
>cat server.arm
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBNTCB4AIBADB7MQswCQYDVQQGEwJVUzELMAkGA1UECBMCTkMxDTALBgNVBAcT
BENhcnkxGDAWBgNVBAoTDOlCTSBDb3Jwb3JhdGlvbjEUMBIGA1UECxMLSVRTTyBS
YWxpZ2gxIDAeBgNVBAMTF212czAzYy5pdHNvLnJhbC5pYmOuY29tMFwwDQYJKoZI
hvcNAQEBBQADSwAwSAJBAJaDyGjFOxIvb3FXm68t66tDQ+dn9B/zLthCS7dc7nor
KT6YpfjnI7duvw/zXXMrrJP99y4oLIGafHIZq1qAHoOCAwEAAaAAMAOGCSqGSIb3
DQEBBAUAA0EAc7OFskVCHrzZXkyoIa6NnDdrtt6CHhMKLJKtIitStFPXZVIMQxPK
1ER2vdsdzpQtIqgTromX2Jf4l4qm47gcWA==
-----END NEW CERTIFICATE REQUEST----
```

Figure 3-38 The content of the certificate request file

As shown in Figure 3-39 on page 79, you can copy and paste the contents of the request file into the VeriSign form. The process should be similar for any other well-known CA vendors, such as thawte, Entrust, RSA or Equifax.

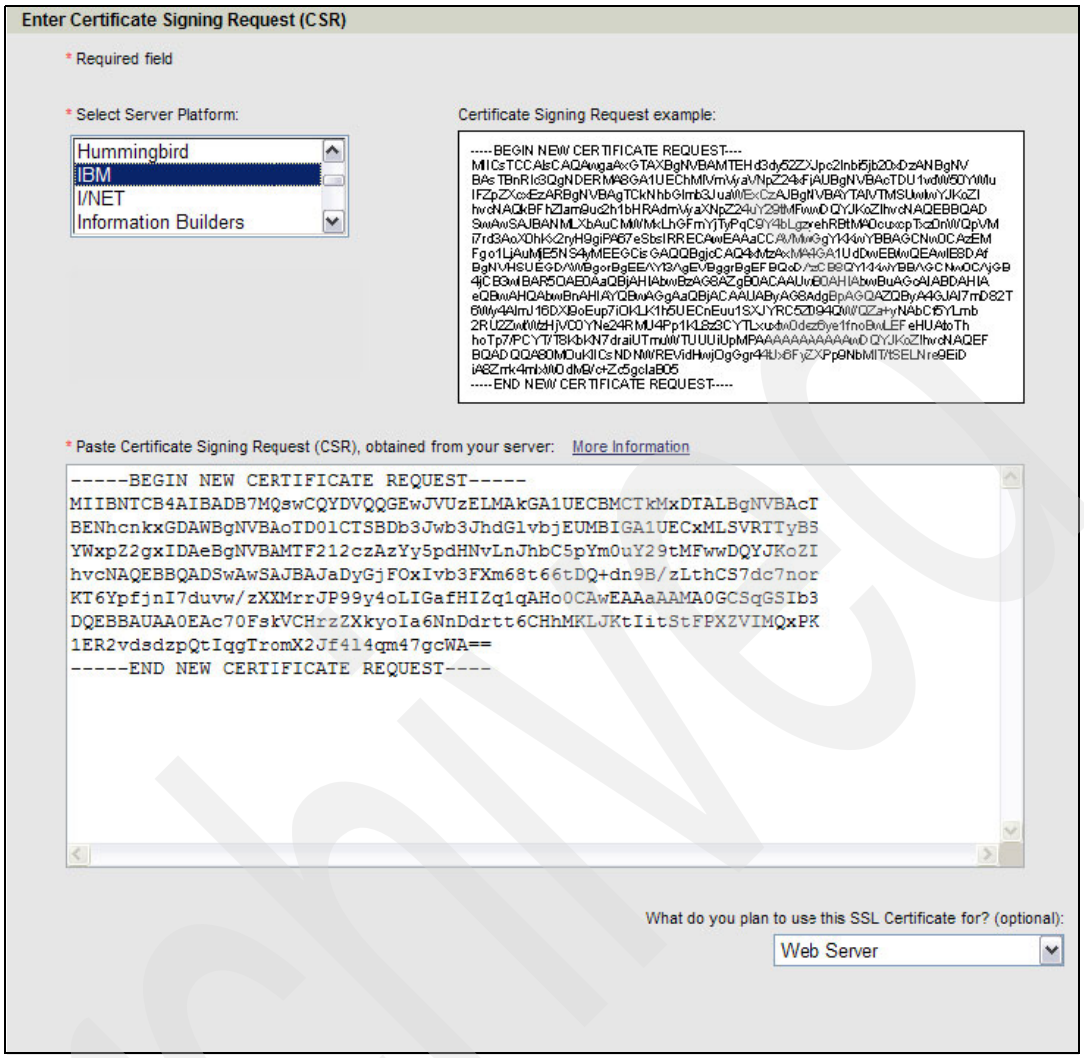


Figure 3-39 Certificate request submit form at VeriSign Web site

After you send a certificate request to an external CA, it can take some time 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 z/OS SSL function in our 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, therefore we received it almost immediately after submitting the certificate request.

3. Receive the certificate from the CA.

   After receiving a certificate from the CA via e-mail, copy and paste it to a z/OS UNIX file (other transfer methods are acceptable, as long as they are text-based). As shown in Figure 3-40 on page 80, 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 cmVwb3NpdG9yeS9UZXNOQ1BTIEluY29ycC4gQnkgUmVmLiBMaWFiLiBMVEQuMUYw
000005 RAYDVQQLEz1Gb3IgVmVyaVNpZ24gYXVOaG9yaXplZCBOZXNOaW5nIG9ubHkuIE5v
000006 IGFzc3VyYW5jZXMgKEMpVlMxOTk3MB4XDTk5MDgwNjAwMDAwMFoXDTk5MDgyMDIz
000007 NTk1OVowgYExCzAJBgNVBAYTAlVTMRcwFQYDVQQIEw5Ob3JOaCBDYXJvbGluYTEN
000008 MAsGA1UEBxQEQ2FyeTESMBAGA1UEChQJSUJNIENvcnAuMRQwEgYDVQQLFAtJVFNP
000009 IFJhbGlnaDEgMB4GA1UEAxQXbXZzMDNjLmlOc28ucmFsLmlibS5jb2OwXDANBgkq
000010 hkiG9wOBAQEFAANLADBIAkEA4P/8r7jWD27V1XWTP1l2GgOqcakpxrTaXZ78x/Sr
000011 EMydBymOnxhrRzK21DFbpTlbM9mT+juOav9mKiUxf19WswIDAQABMAOGCSqGSIb3
000012 DQEBBAUAAOEAr2OtpJvdpN4NcR6Lzx3eBGUZ4VtwtwkvKeU2AU6N9/JXOMGS2r+m
000013 IckUeu4+pRF+cHZY8uLjL1hA+cOBux4RKA==
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 3-40 The content of the server certificate issued by the trusted CA

Because the **gskkyman** utility accepts only z/OS UNIX files, you have to create a z/OS UNIX file for your certificate.

4. Import the CA-signed personal server certificate into the key database.

Note: If your CA vendor requires an intermediate certificate, you should obtain one. The intermediate may have been delivered via an e-mail, or you may need to go to the vendor's Web site to cut and paste the certificate into an HFS file (as described in step 3).

From the **gskkyman** main menu, you can use option **7 - Import a certificate**, to add an intermediate certificate.

Figure 3-41 shows **gskkyman** being used to import the certificate into your key database.

```
 Key Management Menu

       Database: /u/cs08/matt.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 5 1

Enter certificate file name (press ENTER to return to menu): server.cert 2

Certificate received.

Press ENTER to continue.
```

Figure 3-41 Store the server certificate into the key database

Open the key database (refer to Figure 3-24 on page 63 for an example).

1. Select option **5 - Receive requested certificate or renewal certificate** to store your server (user) certificate file.
2. Specify your server certificate file created in Figure 3-40 on page 80.

You should set this certificate as the default.(refer to Figure 3-26 on page 65 for an example of setting a default certificate).

Note: If the server supports AT-TLS, the default certificate is not the only choice for sending to the client. Another certificate (non-default) can be specified within the AT-TLS configuration for this server, and sent to the client to identify this server.

External CA-signed client authentication, RACDCERT procedure

The procedure for a client to request a CA-signed user (personal) certificate is dependent on the type of client software being used. Most SSL/TLS-enabled clients will have a method to create a file with a certificate request for submission to a CA.

A client user certificate, in addition to being stored in the client's key database, must be added to RACF and associated with the appropriate RACF user ID using the RACDCERT ID(clients-user ID) ADD command. The client certificate's CA must be connected to the server's RACF key ring using RACDCERT ID(servers-user ID) CONNECT.

Here is the basic procedure to follow:

1. Generate a certificate request at the client.

   Clients have different ways to generate a certificate request from an external CA. For this example, we will be requesting a client certificate for a personal communications (TN3270) client such as IBM Personal Communications (PComm). Figure 3-42 shows the PComm window to request a certificate.

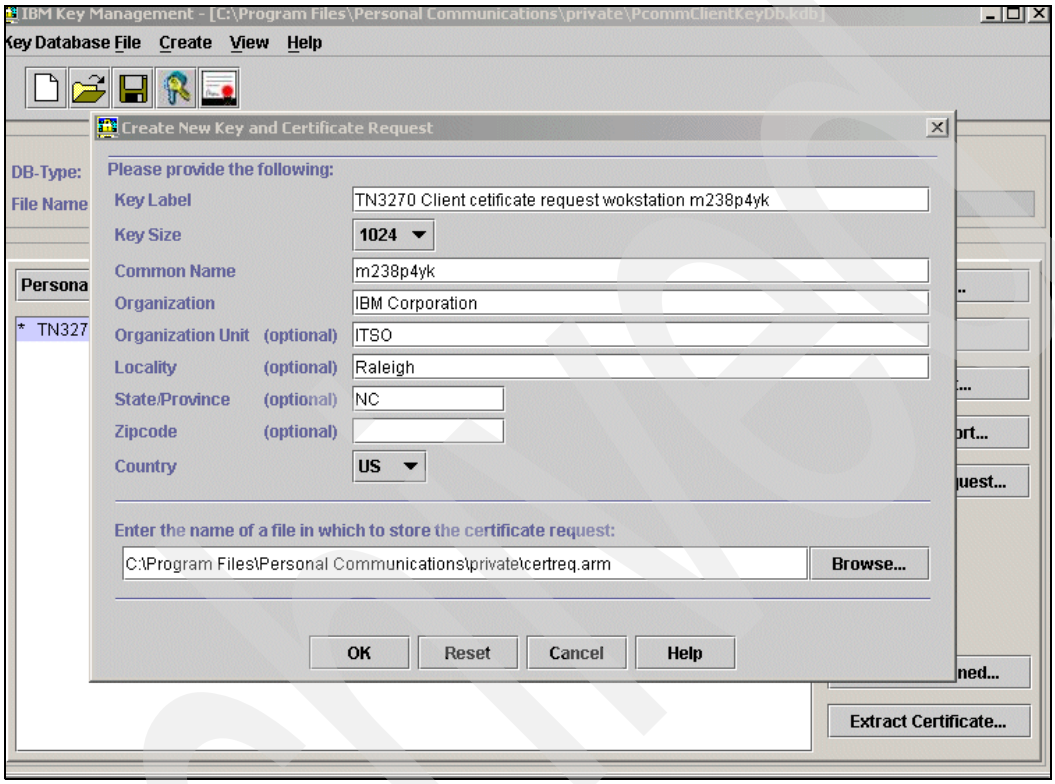


Figure 3-42 Using a client to request a certificate

Figure 3-43 illustrates an example of the certificate request file that is created.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqDCCARECAQAwaDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQH
EwdSYWxlaWdoMRgwFgYDVQQKEw9JQkOgQ29ycG9yYXRpb24xDTALBgNVBAsTBElU
UO8xETAPBgNVBAMTCGOyMzhwNHlrMIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQC81RfEw/spXrdZY/eSES6kFkrI+BvO1lVhYQ+X/+lgsA/Bbb85e75hsPAHU/+q
xeDC2JDqJrjPIChbwxbOMRofxwYhSpu51grQJIYYMehbWlmz9BvF3V+I8SV2fp+A
uPXtjwl7cTC1tNO+mbBnlxgYVDaygOgkh8XhlM4QMderIwIDAQABoAAwDQYJKoZI
hvcNAQEEBQADgYEAHdJbb3R3i7a2WJgQKnl+TdbeJxX9D8bdufXfzwCRckLqBPNi
kVeh6Hg5z+UeLX7O+Cr3TsPmYJHAXZYQCNATCsHIRj1p5XC5OVDrckEG/RpVLvfO
36Y2fYOT4f86sOy8L2RwhRSm3V2mC5vG9JjlB1MS2hkQ13ZWFkYrFMvwczo=
-----END NEW CERTIFICATE REQUEST-----
```

Figure 3-43 Certificate request generated by client for external CA

2. Send the certificate request to the vendor CA.

   You e-mail the certificate request output from the client, either by using cut and paste, or as an attachment. The CA may take some time to generate and send the certificate and private key back to you.

3. Receive the certificate from the CA.

   Depending on the CA, you may have to go to a secure Web site to download your client certificate and key, or it may be sent to you in a secure e-mail. Whatever method is chosen, you must end up with a file, probably in PKCS #12 format, which contains both a digital certificate and a private key. This file will be password-protected.

4. Import the client certificate (and private key) into the client key database.

   In step 3, the certificate and key were received and detached into a workstation file. That certificate and key must now be imported into your client's key database. As an example, we show how to import the client certificate and key into the PComm key database.

   First, start the Certificate Management utility. Click **Start** -> **Programs** -> **IBM-Personal-Communications** -> **Utilities** -> **Certificate Management**.

   Open the PComm key database and enter the password (the default installation password is PCOMM).

   After starting the IBM Certificate Management utility, you should see something like the example illustrated in Figure 3-13 on page 53.

   Just below the heading `Key database content`, there is a drop-down box containing the text `Signer Certificates`; the certificates listed are the CA certificates. Click the drop-down box and select **Personal Certificates**. You will then be presented with a list of personal certificates. If you have never imported any, the list will be blank.

   Click the **Import** button and select the certificate file that you exported in step 1, and click **OK**. You will be asked for a password, which will have been provided by your CA.

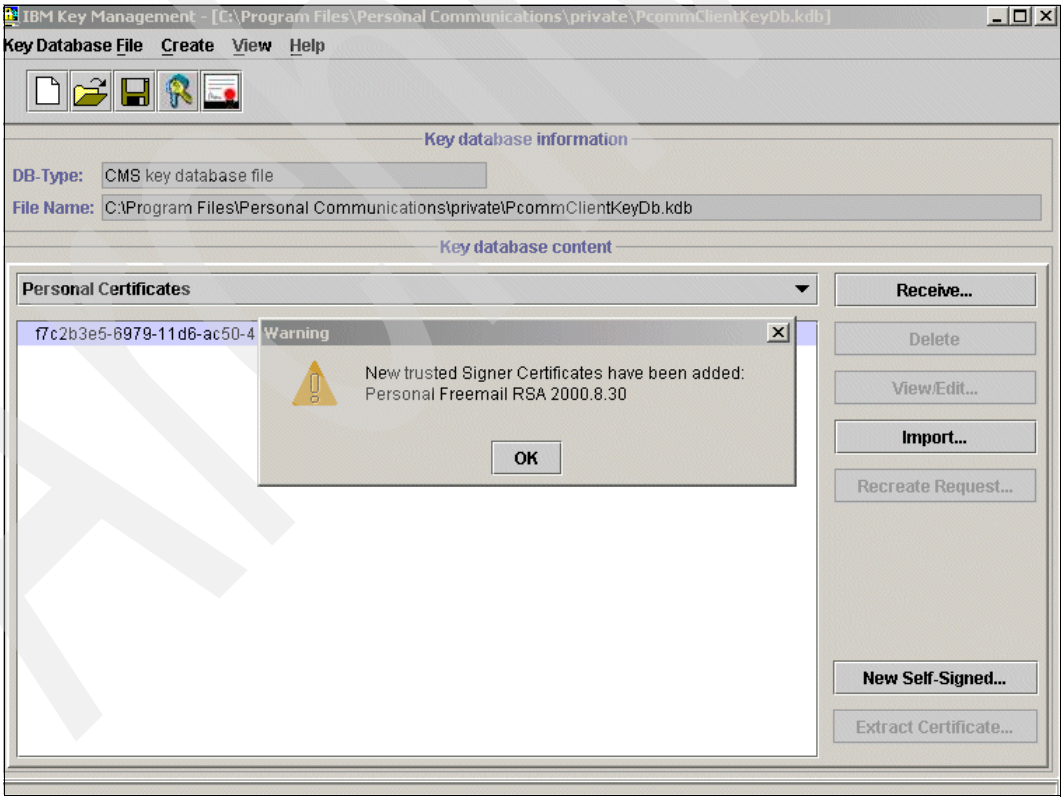


Figure 3-44 Personal Communications certificate dialog after importing a client certificate and key

Figure 3-44 on page 83 shows the window after you have imported the key/certificate into the key database. The file received from the CA may contain not only the client certificate; it may also contain all intermediate certificates, up to and including the root signer's certificate. The import function may have added several certificates to the client key database, one in the Personal Certificates section and one or more in the Signer Certificates section, as indicated in the informational window.

5. This step might not need to be performed at all. Your well-known CA vendor should already have the signer root CA certificate in the certificate repository that is shipped by IBM; that is the definition of a well-known CA.

   Consequently, there two possible variations of step 5:

   – If your CA vendor has provided you with the issuer's and subject's DN of the root CA, then compare your root CAs DNs against the list found in Appendix C of z/OS *Security Server RACF Security Administrator's Guide*, SA22-7683. Scan for a certificate label that sounds like your root CA. Then, check the DNs of that certificate for an exact match.

     You should find your vendor's root CA certificate listed. If you do, the next step is to issue a RACDCERT CERTAUTH ALTER (LABEL('labelname')) TRUST to change this certificate to a status of trusted. Then, issue RACDCERT ID(userID) CONNECT(CERTAUTH LABEL('labelname') RING(serverringname) to connect the certificate to your server's key ring. Remember that server authentication must be set up first, so you should already have completed the steps described earlier to establish a RACF key ring.

   Otherwise:

   – If you did not find your root CA among those supported by RACF, then export the client root CA certificate to a workstation file and import it into RACF. We explain how to perform this task in step 2 through step 5 of "Self-signed client authentication, RACDCERT procedure" on page 59.

External CA-signed client authentication, gskkyman procedure

The basic procedure to follow for **gskkyman** is:

Steps 1 through 4 are exactly the same as those outlined in "External CA-signed client authentication, RACDCERT procedure" on page 81:

1. Generate a certificate request at the client.
2. Send the certificate request to the CA.
3. Receive the client certificate from the CA.
4. Import the certificate into the client's key database.
5. Again, this step has two possible variations:

   – If your root CA is already in the **gskkyman** database, then from the **gskkyman** main menu, select **2 - Manage certificates**. Page through the list (using the enter key) looking for a label that looks like your root CA. Select that certificate's number and check the DNs against your root CA's DNs.

     If you find a match, you do not need to do anything further. Root CAs are automatically connected and trusted in a **gskkyman** environment.

   Otherwise:

   – If you do not find your vendor's root CA, then import the root CA following the example found in Figure 3-31 on page 70.

4

Network Security Services

In this chapter, we explain how you can use Network Security Services (NSS) by implementing a Network Security Services Daemon (NSSD) and a Network Security Services Client.

The following topics are discussed in this chapter.

Section	Topic
4.1, “Basic concepts” on page 86	Basic concepts behind a Network Security Services implementation
4.2, “Configuring IKE in RSA signature mode” on page 96	How to build an IPSec dynamic tunnel with IKE in RSA signature mode
4.3, “Configuring Network Security Services (NSS)” on page 128	The preliminary and integral steps needed to configure the NSS environment
4.4, “Verifying the NSS environment” on page 193	The commands to us to verify your working environment
4.5, “Diagnosing the NSSD environment” on page 211	The references, commands, and log entries that are helpful in diagnosing NSS problems
4.6, “Worksheet for NSSD implementation” on page 214	Worksheet to help you design for the NSS environment on your installation, to simplify configuration of the implementation.
4.7, “References” on page 215	References to additional information that is helpful in implementing NSS

4.1 Basic concepts

To understand Network Security Services (NSS), you must first understand IPSec and the role of the Internet Key Exchange Daemon (IKED). Therefore, we encourage you to read Chapter 9, “IP Security” on page 373 before continuing with this chapter on NSS.

4.1.1 Review of IKED

IP Security (IPSec) is a protocol that provides authentication, data integrity, and data privacy for IP packets flowing between two IPSec endpoints. It is the technology used to provide security between endpoints by over what is called a Virtual Private Network (VPN). It is protocol-independent, allowing security to be applied not only to TCP packets but also to UDP and other IP flows.

The VPN requires the management of cryptographic keys and security associations across VPN tunnels between the endpoints. The tunnels can be built manually or dynamically, as explained here:

- Manual tunnels are built when security parameters and encryption keys are statically configured at each end of the tunnel; the tunnels are managed manually by a security administrator. Keys are refreshed when the tunnels are manually recycled.
- Dynamic tunnels utilize the Internet Key Exchange (IKE) protocol to negotiate security parameters and to generate encryption keys dynamically in order to lower the risk of secret key compromise. The keys are dynamically refreshed during the lifetime of the tunnel.

With dynamic tunnels the Internet Key Exchange (IKE) protocol can work with pre-shared keys or with “RSA signature mode” during what is called Phase I, as you see in Figure 4-1.

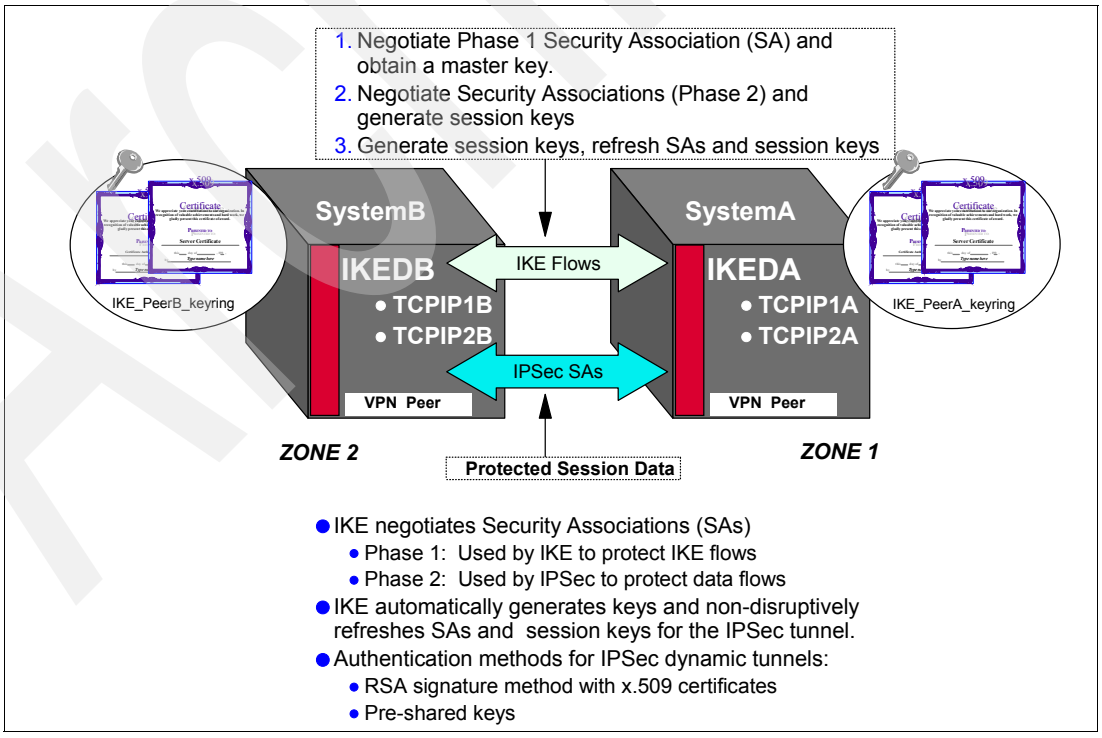


Figure 4-1 Internet Key Exchange (IKE) concepts

With pre-shared keys, IKE authenticates the peers by means of a secret key that has been shared between the IKE peers; this is called “pre-shared key mode.” With “RSA signature mode,” the IKE protocol utilizes x.509 certificates to authenticate the IKE peers. Therefore, in RSA signature mode, each IKE partner has access to a key ring; the key ring holds private keys, at least one certificate associated with the local server, and at least one Certificate Authority certificate to authenticate the peer.

Figure 4-1 on page 86 shows that each IKE peer has access to such a key ring: at SystemA the key ring is named IKE_PeerA_keyring; at the VPN Peer, SystemB, the key ring is named IKE_PeerB_keyring.

Note that NSS plays a role in RSA signature mode authentication only; NSS is not concerned at all with pre-shared key mode because RSA signature mode is a much more scalable approach than that of pre-shared key mode. Generally speaking, each security endpoint, represented by an IKE agent, may reside in a separate zone of the IP network. Some zones are more trusted than others. For example, in Figure 4-1 on page 86, Zone 1 may be within a highly secure enterprise data center. Zone 2 may be outside of any corporate firewall and connected to the “open Internet.”

After the IKE (phase 1) tunnel is established, other protocols are used to generate unique keys for each IPSec (also called the phase 2) tunnel. Phase 2 does not require any further RSA signature operations. Because each IKE and IPsec tunnel has a limited lifetime (either based on time or on the amount of data flowed), nondisruptive tunnel refreshes take place over time. For IKE tunnels, this again requires RSA signature operations.

Managing certificates and signatures in this environment can become onerous if the network is large. Not only is the administration of certificates across many security endpoints cumbersome and error-prone, but even the configuration and deployment of the many certificates is subject to the same burdens. Then there are the security vulnerabilities themselves. Sensitive data like private keys can be compromised when stored in less trusted zones.

The Network Security Services solution can overcome these challenges.

4.1.2 The Network Security Services solution

The Network Security Services (NSS) server provides a set of network security services for IPSec. These include the certificate (and digital signature) service for IKE Phase I negotiation and the network management service. The certificate service and network management service are used by NSS clients, represented in Figure 4-2 on page 88 by the Internet Key Exchange Daemon (IKED) in SystemB. IKED has been enhanced with NSS client functionality on SystemA.

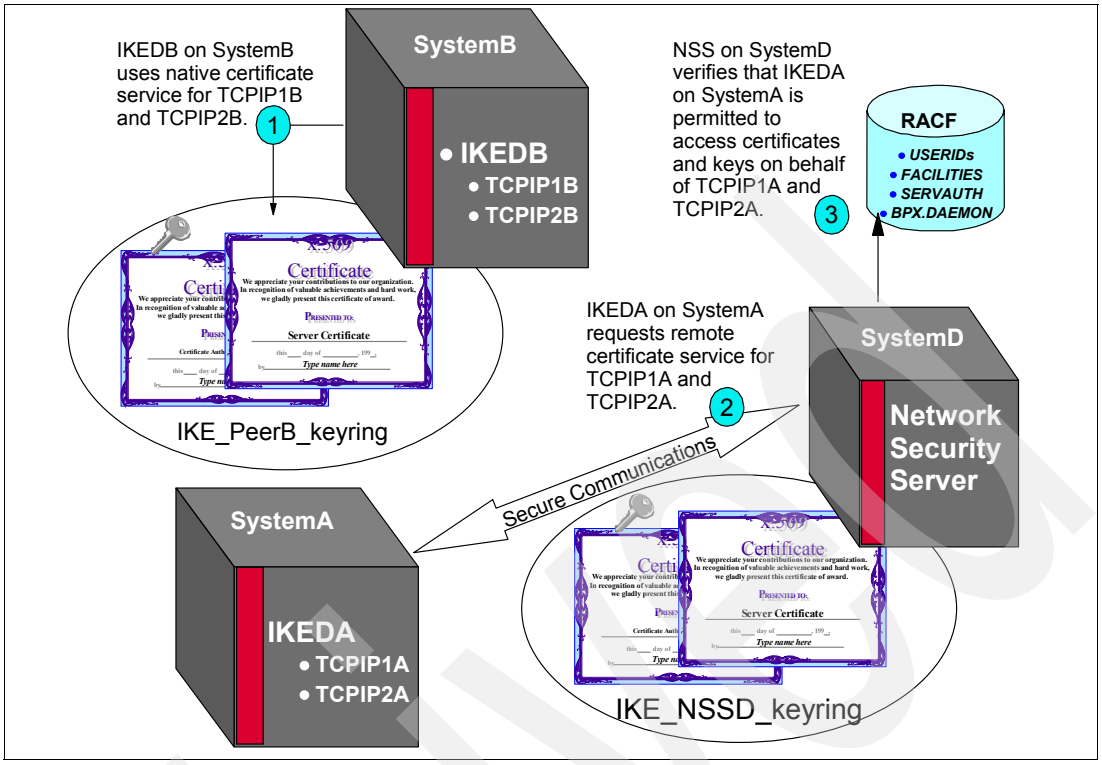


Figure 4-2 Overview of Network Security Services: certificate management

In Figure 4-2, IKEDB on SystemB at (1) uses the RSA certificates and private keys that are stored natively on the local key ring (IKE_PeerB_keyring). On the other hand, IKEDA in SystemA is a client of the NSS on SystemD and therefore no longer requires local certificate services; at (2) you see how it uses the RSA certificates and private keys that are stored on the NSS server's key ring (NSSD_keyring). In fact, the NSSD key ring takes the place of the former IKE_PeerA_keyring. The information that the client needs is sent by the NSS server across a connection secured with TLS/SSL.

Note: Although we define IKEDA as the NSS client, in reality NSS maintains a separate connection for each NSS-enabled TCP/IP stack. In SystemA, TCPIP1A and TCPIP2A are both NSS-enabled. Each stack appears as a separate NSS client to the NSS server.

It is possible to have a third stack (for example, TCPIP3A) in System A that is not NSS-enabled. (TCPIP3A is not depicted in Figure 4-2 on page 88.) In such a case, TCPIP3A would still need access to a local key ring. This type of configuration could be necessary in a migration scenario, in which stacks are enabled for NSS one by one.

When an NSS client uses the NSS certificate service, the NSS server consults SERVAUTH resources in the RACF database (3) to verify IKEDA's privileges to access the NSSD_keyring and its contents on behalf of TCPIP1A and TCPIP2A. After that is accomplished, the NSS creates and verifies RSA signatures on behalf of the NSS client using private keys and RSA certificates that are stored only at the NSS server on the remote ring (NSSD_keyring). Thus you see that the NSSD_keyring, although owned by NSS, contains the same certificates that formerly populated the IKE_PeerA_keyring to which we referred in Figure 4-1 on page 86.

Figure 4-3 on page 89 illustrates several of these points.

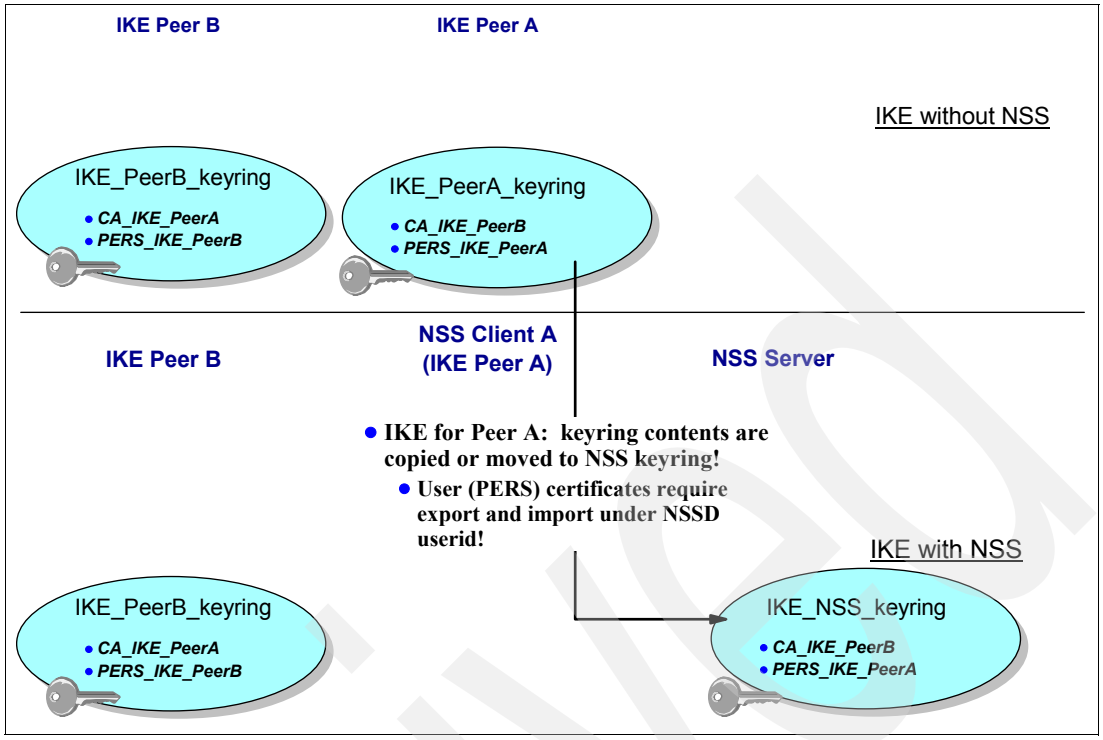


Figure 4-3 Contrasting key ring usage: IKE without NSS and IKE with NSS

Because the NSS key ring is functionally equivalent to the IKE key ring, you see in Figure 4-3 that it contains the same certificates that the IKE peer needs for IKE processing. Thus, the certificates that formerly resided on the IKE key ring of the new NSS client are copied or moved to the key ring of the NSS server.

Note: Our examples use RACF as the external security manager. Equivalent definitions in other external security managers should also be able to provide this function.

RACF also contains user ID and BPX.DAEMON definitions necessary to the implementation of NSS. We discuss these later when we describe the implementation steps for Network Security Services.

NSS also provides network management services for NSS clients, as depicted in Figure 4-4 on page 90.

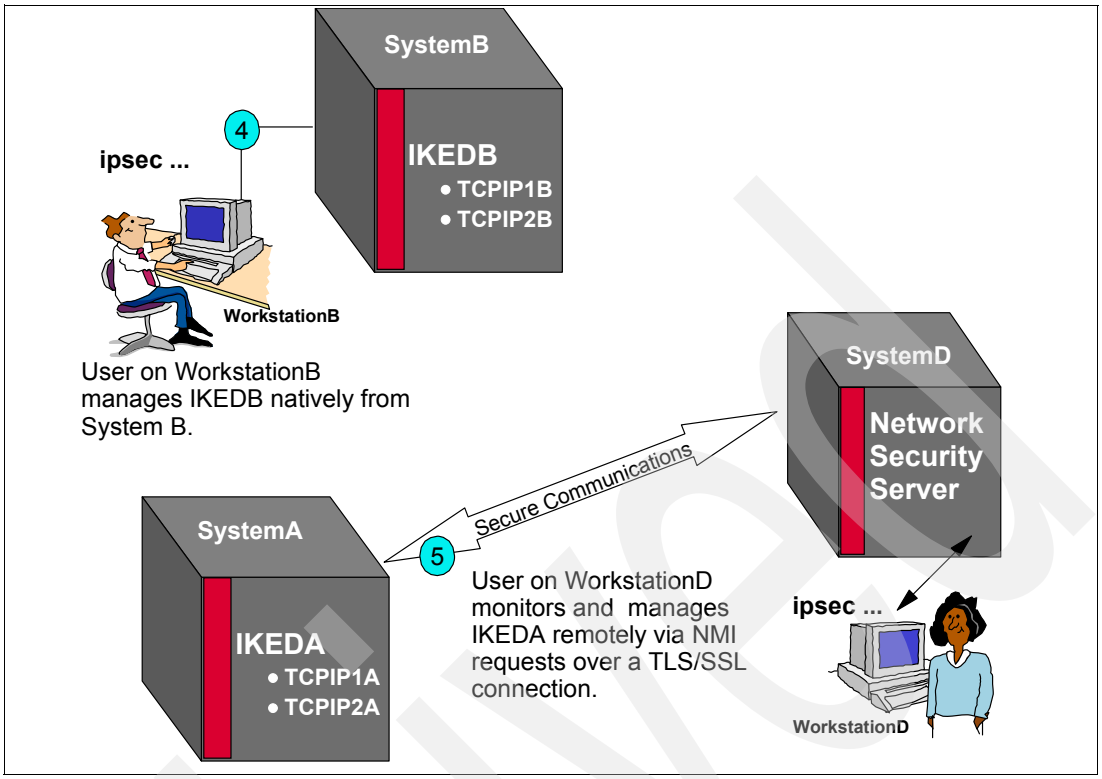


Figure 4-4 Overview of Network Security Services: Network Management Service

When an NSS client uses the network management service, the NSS server routes IPSec network management interface (NMI) requests to that NSS client, which enables the NSS client to be managed remotely. The NSS client provides the NSS server with responses to these requests. The **ipsec** command is enhanced to use NSS remote management services to monitor and control remote IPsec endpoints. The user on Workstation D may issue commands to view the IPSec tunnel with an endpoint on SystemA, to activate, deactivate, or refresh a security association or to switch between the TCP/IP stack's default IP filter policy and the PAGENT IP security filter policy.

In Figure 4-4 at (4) you see that the user on WorkstationB sends **ipsec** commands to IKEDB at SystemB while the user is connected to SystemB. Conversely, at (5), the user at WorkstationD is connected to SystemD but can manage IKEDA at SystemA remotely due to the connection between the NSS daemon (NSSD) and the NSS client, IKEDA on SystemA. These communications take place across a connection secured with TLS/SSL.

The IKE daemon can be configured to act as an NSS client on behalf of multiple TCP/IP stacks. Later we show how to use the **-z** option of the **ipsec** command or the IPSec NMI to manage NSS clients that use the NSS network management service.

For details about using the **ipsec** command to manage NSS clients, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781. For details about using the IPSec NMI to manage NSS clients, see *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787.

NSS benefits

NSS centralizes the sensitive key ring material that would otherwise need to reside in less secure zones of the network onto a single location in the most secure zone of the network. In addition, NSS allows for centralized configuration and administration of certificates. It

provides a central, SAF-enabled repository for RSA certificates along with signature services within the most trusted zones.

We list here several other advantages inherent in the NSS solution, which allows you to:

- Eliminate the need to distribute certificates to security endpoints
- Centralize and reduce configuration and deployment complexity, especially when used with Centralized Policy Services
- Offload digital signature operations from the IKE daemon (the NSS client)
- Enable monitoring and management of remote IPsec endpoints through the **`ipsec`** command and a network management programming interface

Figure 4-5 expands on our view of the NSS solution shown in earlier figures.

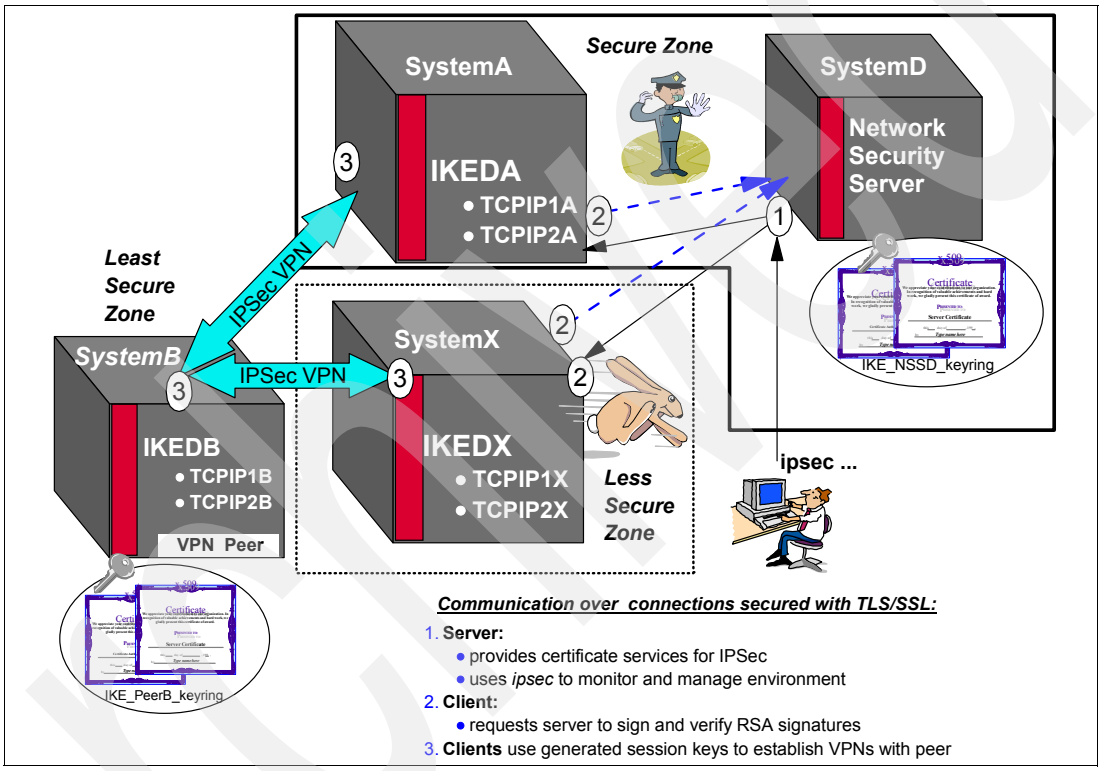


Figure 4-5 Benefits of Network Security Services solution

Figure 4-5 shows a basic NSS setup involving a single NSS server and two clients. IKEDA in SystemA has become an NSS client; IKEDX in SystemX is a new NSS client. The NSS still resides in SystemD. The clients are all z/OS IKE daemons. Each IKE daemon can act as an NSS client for up to 8 TCP/IP stacks running on that system. Communication between client and server systems is secured by TLS/SSL connections.

As we have mentioned before, any of these z/OS systems may have additional TCP/IP stacks which are not enabled for NSS.

- Also note that multiple IKE daemons can simultaneously access network security services. These IKE daemons do not have to reside within the same sysplex as the NSS server. In Figure 4-5, we have two different sysplexes:
  - The Secure Zone sysplex, which contains the NSS server system and SystemA. This sysplex resides in a highly secured data center.
  - The Less Secure Zone sysplex, which contains system SystemX. This is possibly a test sysplex in the data center.

- Then you see the Least Secure Zone, one that may lie across the Internet in another company's intranet and contains the VPN peer, SystemB.

Key rings for the NSS implementation

You have seen in Figure 4-2 on page 88 that IKE daemons and the NSS server require key rings to hold the private keys and certificates for RSA authentication during IKE negotiations.

Note: The NSSD key ring contains the certificates that would normally have resided on the IKE keyring. The NSSD key ring takes the place of the IKE key ring for any NSS clients.

You have also seen in Figure 4-2 on page 88 and in Figure 4-4 on page 90 that the communications between the NSS clients and the NSS server are secured by TLS/SSL protocols. Thus, the NSS solution also requires TLS/SSL policies that point to key rings that hold TLS/SSL certificates and private keys. Refer to Figure 4-6.

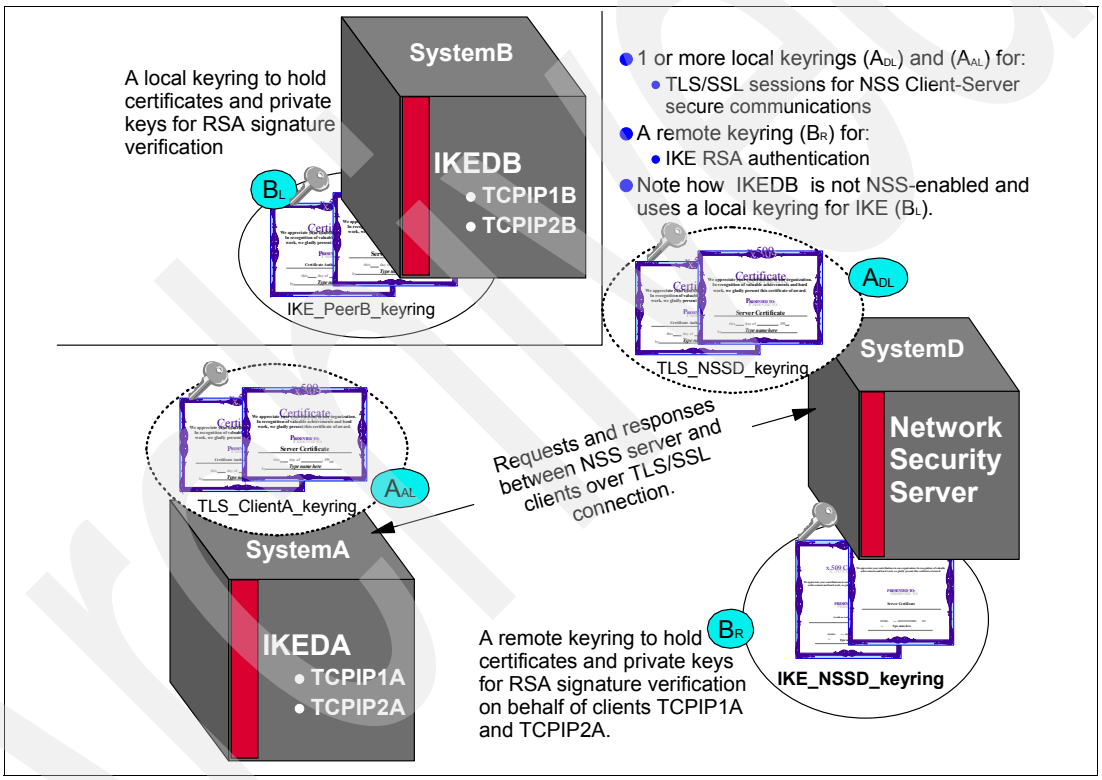


Figure 4-6 How NSS uses key rings for IKE processes and for secure client - server communication

Figure 4-6 illustrates how IKEDB at SystemB, the IKE peer to IKEDA at SystemA, relies on a local, native key ring for the RSA signature verification (B_L). It also illustrates how IKEDA at SystemA relies on the remote IKE keyring managed by the NSS server (B_R). In addition, you see that we require certificates for the secure communication between client and NSS server. These certificates are stored in a different set of key rings (A_R and A_L).

As discussed in Chapter 3, "Certificate management in z/OS" on page 31, certificates can be stored on key rings in many different ways. So, here we look at different strategies for managing key rings, as illustrated in Figure 4-7 on page 93.

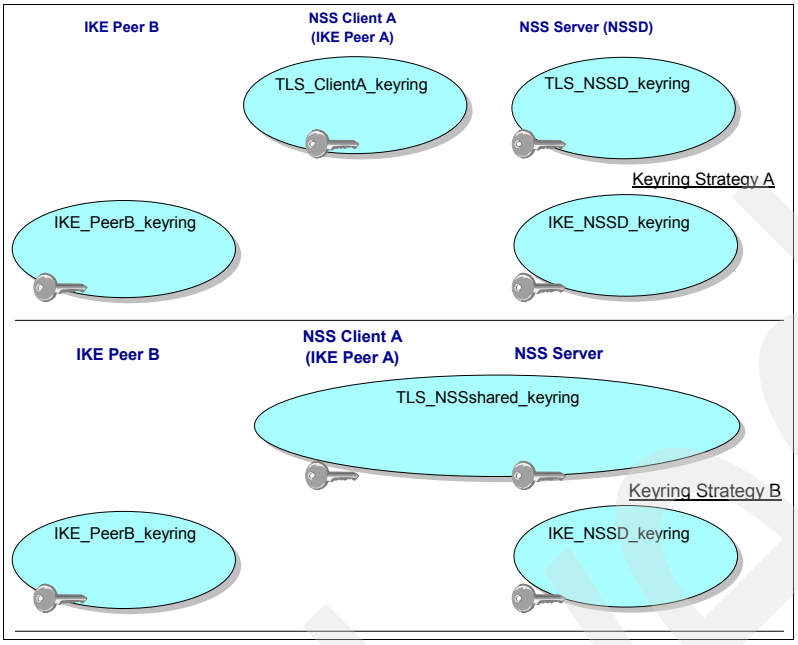


Figure 4-7 Key ring strategies for an NSS implementation

IPSec, IKE, and TLS key ring strategies for the NSS implementation

1. Key ring Strategy A, shown in Figure 4-7, consists of the following:
   - A key ring solely for TLS/SSL at the NSS server side (TLS_NSSD_keyring).
   - A separate key ring for TLS/SSL processing at the NSS client node (TLS_ClientA_keyring).
   - One key ring solely for IKE remote certificate processing at the NSS server side (IKE_NSSD_keyring). This key ring *must* be owned by the user ID associated with the *NSSD* procedure.
   - A separate IKE key ring solely for native IKE certificate processing at any node not serviced by the NSS server (IKE_PeerB_keyring). This key ring *should* be owned by the user ID associated with the *IKED* procedure.

   Strategy A is the simpler strategy to implement if you are unfamiliar with certificate management on key rings. All key rings are managed on a node-by-node basis. There should be no confusion as to the role a particular certificate on such a key ring plays.

2. Key ring Strategy B, which is also shown in Figure 4-7, consists of the following:
   - A shared TLS key ring for both the NSS client and the NSS server (TLS_NSSshared_keyring).
   - An NSS key ring at the NSS server site used for remote certificate processing on behalf of NSS clients (IKE_NSSD_keyring). This key ring *must* be owned by the user ID associated with the *NSSD* procedure.
   - A local IKE key ring for any peer node not serviced by the NSS server (IKE_PeerB_keyring. This key ring *must* be owned by the user ID associated with the *IKED* procedure.

   Strategy B is also simple to implement. It is common at many installations to share an SSLS/TLS key ring among multiple servers that require security and across LPARs within a sysplex. All TLS key rings are managed on a sysplex-wide basis. Note that the IKE key rings in Strategy B continue to be maintained separately. This represents the logical way to support IKE and NSS key rings, because the theory behind NSS is the centralization and isolation of IKE certificate management for a specific client set.

Note: In our RSA scenario, we used Strategy B because we already had TLS key rings that satisfied our needs in the sysplex. We did not implement TLS client authentication.

Next, we focus on the types of certificates that are stored on the individual key rings; see Figure 4-8.

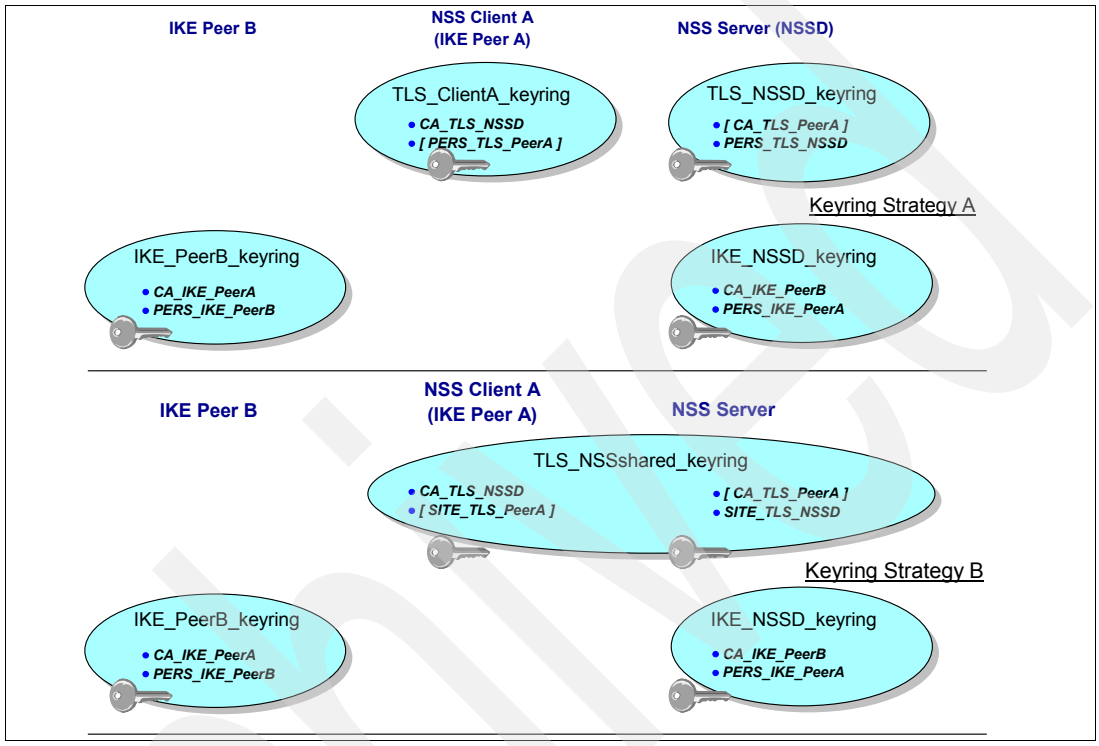


Figure 4-8 Key ring contents for an NSS implementation

3. Figure 4-8 shows the contents of the key rings illustrated in Key ring Strategy A.
   - For SSL/TLS, only server authentication is required; client authentication is optional.
     - TLS_NSSD_keyring

       On the NSS Server's shared key ring (TLS_NSSshared_keyring), you see a server certificate that represents the NSSD for TLS processing: PERS_TLS_NSSD.

       You also see the Certificate Authority (CA) certificate that was used to sign the NSSD server's individual, personal certificate; this is used by the client in authenticating the server certificate. You may optionally see the Certificate Authority (CA) Certificate, CA_TLS_PeerA, that has signed the TLS client's personal certificate if client authentication is necessary.

       Note: In this example, we employed user (PERSONAL) certificates. Such certificates must be owned by the user ID that requires access to the key ring. If IKE requires access, its user ID must own the user (PERSONAL) certificate for TLS; if NSS requires access, its user ID must own the certificate for TLS negotiations.

       CA and SITE certificates may be used by any user ID or procedure, as long as the proper RACF permissions to the key ring and its keys have been established.

- TLS_PeerA_keyring

  On the NSS Client A's key ring you see the required Certificate Authority (CA) Certificate, CA_TLS_NSSD, that has signed NSSD's server certificate, which is PERS_TLS_NSSD. You may optionally see the user (PERSONAL) client certificate, PERS_TLS_PeerA, that would be necessary if the TLS server requests client authentication.

– For IKE protocols, both partners must authenticate to each other.

  - IKE_NSSD_keyring

    You see here both the certificate of the NSS Client, PERS_IKE_PeerA, and any CA certificates that may have signed the IKE certificates of any peers to IKE_PeerA. In this case, we see the CA certificate, CA_IKE_PeerB.

  - IKE_PeerB_keyring

    You see here the certificate that identifies IKE peer, PeerB: PERS_IKE_PeerB. You also see the CA certificate that authenticates IKE_PeerA: CA_IKE_PeerA.

4. In Figure 4-8 on page 94, you also see the contents of the key rings illustrated in Key ring Strategy B.

   – For SSL/TLS, only server authentication is required; client authentication is optional.

     - TLS_NSSshared_keyring

       On the NSS Server's shared key ring you see a SITE certificate that represents the NSSD for TLS processing, SITE_TLS_NSSD. You also see the Certificate Authority (CA) certificate that was used to sign the NSSD server's site certificate; this is used by the client in authenticating the server certificate.

       You may optionally see the client's SITE certificate, SITE_TLS_PeerA, as well as the Certificate Authority (CA) Certificate, CA_TLS_PeerA, that has signed the TLS client's SITE certificate if client authentication is necessary.

       Note: In this example, you see a shared key ring that is populated with SITE and CA certificates. Refer to x.509 certificate generation and management if you need assistance in understanding the differences between user (PERSONAL) server or client certificates versus SITE and CA certificates.

   – For IKE protocols, both partners must authenticate to each other. The IKE key rings in Strategy B are populated the same way they are with Strategy A:

     - IKE_NSSD_keyring

       You see here both the certificate of the NSS Client, PERS_IKE_PeerA, and any and all CA certificates that may have signed the IKE certificates of any peers to IKE_PeerA. In this case, we see the CA certificate, CA_IKE_PeerB.

     - IKE_PeerB_keyring

       You see here the certificate that identifies IKE peer, PeerB: PERS_IKE_PeerB. You also see the CA certificate that authenticates IKE_PeerA: CA_IKE_PeerA.

Recall that we have implemented Key ring Strategy B in our scenario.

Later in this chapter, we show you how to implement the complete secured environment for both RSA signature verification and for TLS. We assume you are already an expert in SSL/TLS protocols, in creating key rings and certificates, and in IKE in general. Therefore, we make the following recommendation for the implementation of NSSD.

Recommendation: The implementation of NSSD relies on several projects that can have been previously implemented. For example, prior hands-on experience with certificate management, IPSec, IKE, and TLS/SSL greatly simplifies the implementation of NSSD, because your only concern at that point is to create the NSS client and NSS server relationship.

Therefore, we recommend that you reduce your learning curve with NSSD by migrating an existing IPSec and IKE scenario to one using NSSD. We also recommend that you already have experience with the Network Configuration Assistant for building the AT-TLS environment. This is the approach we took in our NSSD scenario.

4.2 Configuring IKE in RSA signature mode

Chapter 9, “IP Security” on page 373, describes how to configure IKE with pre-shared keys only. Figure 4-9 illustrates that scenario.

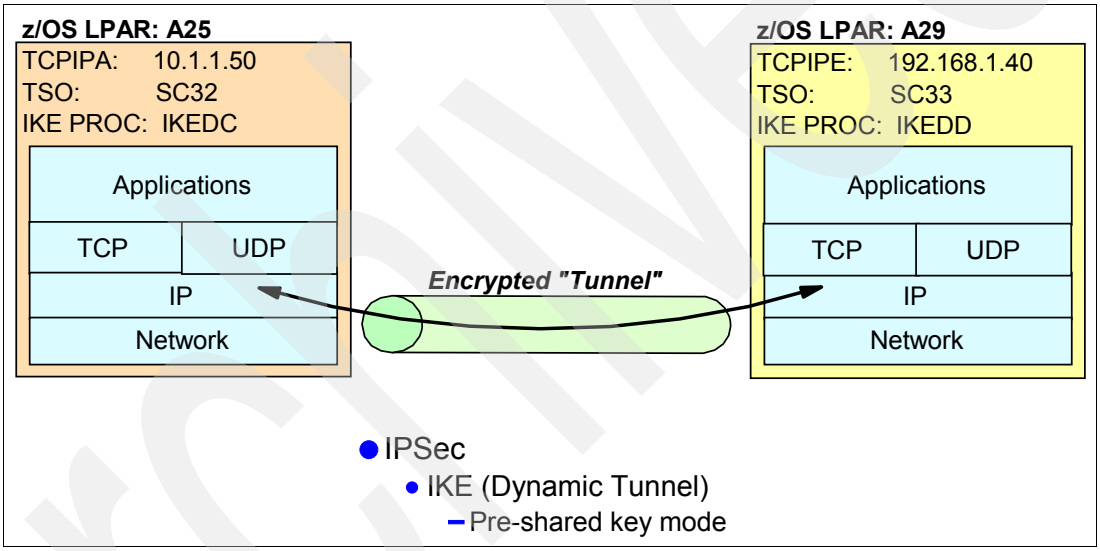


Figure 4-9 VPN tunnel between TCPIPA on SC32 and TCPIPE on SC33: pre-shared key mode

However, you now know that the NSS solution relies on IKE with RSA signature authentication. Therefore, in this section we describe how to change the IKE configuration using pre-shared key (Figure 4-9) to a configuration that builds the encrypted tunnel using RSA signature mode.

After we build dynamic tunnels using RSA signature mode, we then show you how to configure the NSS solution.

4.2.1 Generating certificates for IKE RSA signature mode

Our new scenario is illustrated in Figure 4-10 on page 97.

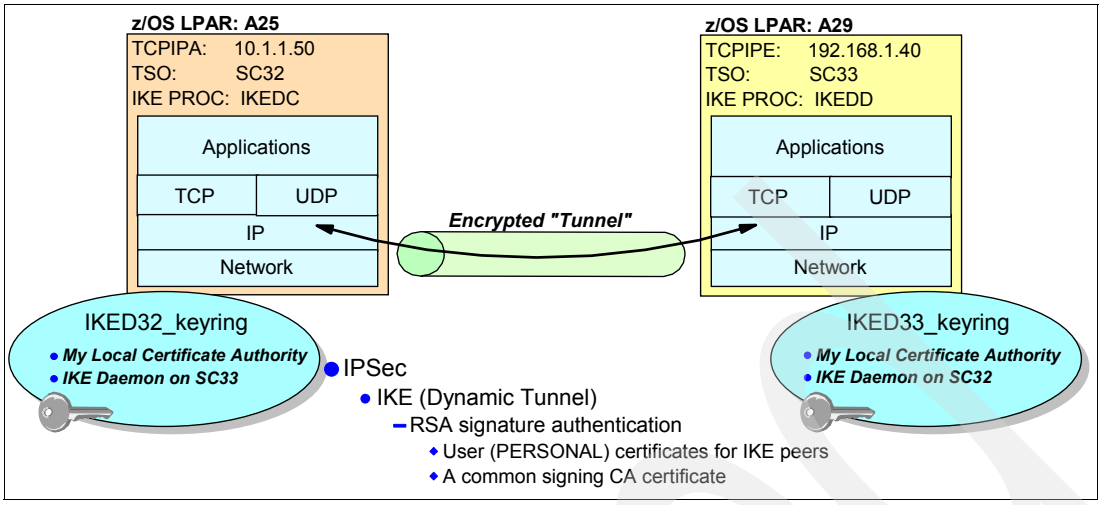


Figure 4-10 VPN tunnel between TCPIPA on SC32 and TCPIPE on SC33: RSA signature mode

With IKE in RSA signature mode we require key ring access, as you see in Figure 4-10. We use two key rings in this scenario, but we could also have shared a key ring between SC32 and SC33. Using two key rings simplifies the explanation of how this scenario, which does not rely on NSS, can be migrated to a scenario with one that does rely on Network Security Services.

Note that IKE procedure running on MVS system SC32 is named IKEDC. The key ring assigned to that procedure is IKED32_keyring. The IKE peer procedure running on MVS system SC33 is named IKEDD. The key ring assigned to IKEDC is IKED33_keyring.

The IKED32_keyring is populated with:

- A user (PERSONAL) certificate representing the IKEDC peer. The label on the certificate is IKE Daemon on SC32.
- A certificate authority certificate that has signed the certificate that resides on IKEDD's key ring. That is, the CA certificate is the one that has signed IKEDD's user (PERSONAL) certificate. The label on the CA certificate is My Local Certificate Authority.

The IKED33_keyring is populated with:

- A user (PERSONAL) certificate representing the IKEDD peer. The label on the certificate is IKE Daemon on SC33.
- A certificate authority certificate that has signed the certificate that resides on IKEDC's keyring. That is, the CA certificate is the one that has signed IKEDC's user (PERSONAL) certificate. The label on the CA certificate is My Local Certificate Authority.

Note: Our scenario uses the same signing certificate authority for both of the PERSONAL certificates. You may encounter situations in which there are two different CAs for the two user certificates, and you will need to ensure that the correct CA is connected to the correct keyring.

JCL to create the key rings and certificates for the IKE RSA scenario

We now show the JCL we used to create the RACF definitions for this scenario. In Example 4-1 on page 98 you see the JCL used to create the user (PERSONAL) certificate for IKEDC in SC32.

Example 4-1 JCL for RACF to create user certificate for SC32 IKEDC process

```
//CERT32    JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERT32   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Create Individual Personal Certificate for SC32              *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
RACDCERT ID(IKED) GENCERT  a                                        -
         SUBJECTSDN  (CN('IKE Daemon on SC32')    b                 -
                      OU('ITSO')                                    -
                      C('US'))                                      -
                      NOTBEFORE(DATE(2007-09-11))   c               -
                      NOTAFTER(DATE(2008-09-11))    c               -
                      WITHLABEL('IKE Daemon on SC32')  d            -
                      SIGNWITH(CERTAUTH                             -
                      Label('My Local Certificate Authority'))  e
  setropts raclist(DIGTCRIT) refresh   f
  racdcert ID(IKED) list(label('IKE Daemon on SC32'))   g
/*
```

Example 4-1 highlights several components of this definition and the commands that follow:

a The ID parameter identifies this certificate that is being generated (GENCERT) as a personal user certificate. (It is not a CA or SITE certificate.)

b SUBJECTSDN identifies several components that comprise the x.509 Distinguished Name (DN) of the certificate owner or holder. Each DN should be unique at least within a RACF database; it should also be unique across the world, because this DN is used to distinguish identities in many cases. We have utilized only three components of the DN for SC32: CN, OU, and C. (Refer to Chapter 3, “Certificate management in z/OS” on page 31, for detailed descriptions of these and other components.)

c These parameters set a time frame for the certificate’s validity. The default time frame is only one year. It is common in a production environment to use a much longer time frame than we used in this scenario.

d The RACF database requires a label for organizing the certificates within a RACF database. The label must be unique.

e This definition tells the GENCERT process which CA should be the signing authority for the user’s personal certificate.

f The **setropts** command refreshes the DIGTCERT class so that the changes are made immediately known in the running RACF environment and operating system.

g The **racdcert** command allows us to verify that our certificate has been properly created; it displays the certificate with its attributes.

Recommendation: Code the validity period for the certificate; otherwise, it defaults to only one year. Also code an ALTNAME parameter. We neglected to do so and were then required to use the full x.500 distinguished name in our IKE definitions in order to authenticate to the IKE peer, as you will see later. Refer to Example 4-3 on page 100 for a discussion about the ALTNAME parameter.

Example 4-2 shows the JCL used to create the user (PERSONAL) certificate for IKEDD in SC33.

Example 4-2 JCL for RACF to create user certificate for SC33 IKEDD process

```
//CERT33    JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERT33    EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Create Individual Personal Certificate for SC33             *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
RACDCERT ID(IKED) GENCERT  a                                        -
         SUBJECTSDN  (CN('IKE Daemon on SC33')    b                 -
                      OU('ITSO')                                    -
                      C('US'))                                      -
                      NOTBEFORE(DATE(2007-09-11))   c               -
                      NOTAFTER(DATE(2008-09-11))    c               -
                      WITHLABEL('IKE Daemon on SC33')  d            -
                      SIGNWITH(CERTAUTH                             -
                      Label('My Local Certificate Authority'))  e
  setropts raclist(DIGTCRIT) refresh   f
  racdcert ID(IKED) list(label('IKE Daemon on SC33'))   g
/*
```

Example 4-2 highlights several components of this definition and the commands that follow the definition:

a The ID parameter identifies this certificate that is being generated (GENCERT) as a personal user certificate. (It is not a CA or SITE certificate.)

b SUBJECTSDN identifies several components that comprise the x.509 Distinguished Name (DN) of the certificate owner or holder. Each DN must be unique at least within a RACF database; it should also be unique across the world, because this DN is used to distinguish identities in many cases. We have utilized only three components of the DN for SC32: CN, OU, and C. In the chapter on Certificate Management all of these components and many more are described. (Refer to Chapter 3, “Certificate management in z/OS” on page 31, for detailed descriptions of these and other components.)

c These parameters set a time frame for the certificate’s validity. The default time frame is only one year. It is common in a production environment to use a much longer time frame than we used in this scenario.

d The RACF database requires a label for organizing the certificates within a RACF database. The label must be unique.

e This definition tells the GENCERT process which CA should be the signing authority for the user’s personal certificate.

f The **setropts** command refreshes the DIGTCRIT class so that the changes are made immediately known in the running RACF environment and operating system.

g The **racdcert** command allows us to verify that our certificate has been properly created; it displays the certificate with its attributes.

Recommendation: Code the validity period for the certificate; otherwise, it defaults to only one year. Also code an ALTNAME parameter. We neglected to do so and were then required to use the full **`x.500 distinguished name`** in our IKE definitions in order to authenticate to the IKE peer, as discussed with Figure 4-17 on page 110.

Previously we recommended coding a “Subject’s Alternate Name” by adding the ALTNAME parameter to a certificate definition. The Alternate Name is often used in IKE as an easy means to identify an IKE peer. You can see how this definition is used in RSA signature mode in a later section of this chapter. Example 4-3 shows a RACF certificate definition that utilizes the Subject Alternate Name field.

Example 4-3 Coding alternate names in a RACF certificate definition

```
racdcert id(iked) gencert           -
  (label('IKE Daemon on SC32'))           -
   ALTNAME ( IP(10.1.1.50) -
               DOMAIN('sc32a.itso.raleigh.ibm.com') -
               EMAIL('sc32a@sc32a.itso.raleigh.ibm.com'))
```

Example 4-3 illustrates how we assigned three alternate names: an IP address, a domain name, and a fully qualified domain name. If we had coded any of the three alternate names in the certificates used in our scenario, we would have had an easier time implementing IKE.

Next, we create the key rings and populate them with the appropriate certificates. Example 4-4 illustrates the RACF commands for IKEDC.

Example 4-4 Keyring JCL for IKEDC

```
//KEYRNG32  JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRNG32 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Add a separate keyring for IKE for SC32                     *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert ID(IKED) addring(IKED32_keyring)
  racdcert ID(IKED) CONNECT(ID(IKED)                      -
                    LABEL('IKE Daemon on SC32')           -
                    RING(IKED32_keyring)                  -
                    USAGE(PERSONAL)
  racdcert ID(IKED) CONNECT(CERTAUTH                   -
                    LABEL('My Local Certificate Authority')       -
                    RING(IKED32_keyring)                  -
                    USAGE(CERTAUTH))
  PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IKED) ACCESS(READ)
  PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(IKED) ACCESS(READ)
  setropts raclist(DIGTRING) refresh
  setropts raclist(DIGTCERT) refresh
  setropts raclist(FACILITY) refresh

  racdcert listring(IKED32_keyring) id(iked)
/*
```

Example 4-5 illustrates the RACF commands for IKEDD.

Example 4-5 Keyring JCL for IKEDD

```
//KEYRNG33  JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRNG33 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Add a separate keyring for IKE for SC33                     *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
    racdcert ID(IKED) addring(IKED33_keyring)
    racdcert ID(IKED) CONNECT(ID(IKED)                      -
                      LABEL('IKE Daemon on SC33')           -
                      RING(IKED33_keyring)                  -
                      USAGE(PERSONAL)
    racdcert ID(IKED) CONNECT(CERTAUTH                  -
                      LABEL('My Local Certificate Authority')       -
                      RING(IKED33_keyring)                  -
                      USAGE(CERTAUTH))
    PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IKED) ACCESS(READ)
    PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(IKED) ACCESS(READ)
    setropts raclist(DIGTRING) refresh
    setropts raclist(DIGTCERT) refresh
    setropts raclist(FACILITY) refresh
    racdcert listring(IKED33_keyring) id(iked)
/*
```

Note: Chapter 9, “IP Security” on page 373, describes how to create certificates (although those scenarios did not use the certificates; we implemented IKE with pre-shared key mode).

That chapter also provides information about hardware encryption.

4.2.2 Configuring the IKE daemon

As discussed in Chapter 9, “IP Security” on page 373, the Internet Key Exchange (IKE) daemon is used for the dynamic management of cryptographic keys. With IKE, the keys are automatically created, distributed, maintained, and refreshed. In addition, the expired security associations are automatically deleted.

Definitions for IKE on SC32

Example 4-6 shows the JCL we used on SC32 for the IKE procedure named IKEDC:

Example 4-6 JCL for IKEDC

```
//IKEDC    PROC
//*
//IKEDC    EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   IKED_FILE=/etc/security/iked.conf2
//*   IKED_CTRACE_MEMBER=CTIIKE01
//*
```

```
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//STDENV   DD DSN=TCPIP.SC32.STDENV(IKED32),DISP=SHR  a
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/security/iked.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

The standard environment file for IKE, highlighted at a, contains the entries depicted in Example 4-7.

Example 4-7 TCPIP.SC32.STDENV(IKED32) for IKEDC procedure

```
IKED_FILE=/etc/security/iked32RSA.conf     b
IKED_CTRACE_MEMBER=CTIIKE00
```

Example 4-7 shows that we are using an IKE configuration file named iked32RSA.conf b, which is stored in /etc/security. This file can be created by the IBM Configuration Assistant, or you can directly edit it on the mainframe.

Note: We show you how to create this file using the IBM Configuration Assistant in “IKE Daemon configuration files” on page 120.

The IKE configuration file contains the entries shown in Example 4-8.

Example 4-8 IKE configuration file for IKEDC

```
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAIKCFG
#

IkeConfig
{
  IkeSyslogLevel    255     a
  PagentSyslogLevel 128     b
  Keyring           IKED/IKED32_keyring  c
  KeyRetries        10
  KeyWait           30
  DataRetries       10
  DataWait          15
  Echo              no
  PagentWait        0
}
```

We initially used high SyslogLevels for both IKE and for Pagent (a and b). We added a task to our plan to lower the logging levels after our testing was complete. Because we were using RSA signature authentication, we included the name of our IKE keyring (c), prefaced by the owner of the keyring, IKED. This name correlates to the diagram shown in Figure 4-10 on page 97.

Definitions for IKE on SC33

Example 4-9 on page 103 shows the JCL we used on SC33 for the IKE procedure named IKEDD.

Example 4-9 JCL for IKEDD

```
//IKEDD PROC
//*
//IKEDD EXEC PGM=IKED,REGION=0K,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   IKED_FILE=/etc/security/iked.conf2
//*   IKED_CTRACE_MEMBER=CTIIKE01
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//STDENV   DD DSN=TCPIP.SC33.STDENV(IKED33),DISP=SHR  a
//* Sample HFS file containing environment variables:
//*STDENV   DD PATH='/etc/security/iked.env',PATHOPTS=(ORDONLY)
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

At a, we are referencing a standard environment variable file named IKED33. Example 4-10 shows the contents of this file.

Example 4-10 TCPIP.SC33.STDENV(IKED33) for IKEDD procedure

```
IKED_FILE=/etc/security/iked33RSA.conf      b
IKED_CTRACE_MEMBER=CTIIKE00
```

Example 4-10 shows that we are using an IKE configuration file named iked33RSA.conf (b), which is stored in /etc/security. This file can be created by the IBM Configuration Assistant, or you can directly edit it on the mainframe.

Note: We explain how to create this file using the IBM Configuration Assistant in “IKE Daemon configuration files” on page 120.

The IKE configuration file contains the entries shown in Example 4-11.

Example 4-11 IKE configuration file for IKEDD

```
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAIKCFG
#
IkeConfig
{
  IkeSyslogLevel    255     a
  PagentSyslogLevel 128     b
  Keyring           IKED/IKED33_keyring  c
  KeyRetries        10
  KeyWait           30
  DataRetries       10
  DataWait          15
  Echo              no
  PagentWait        0
```

```
}
```

As you saw for IKEDC, we initially used high SyslogLevels for both IKE and for Pagent (a and b). We added a task to our plan to lower the logging levels after our testing was complete. Because we were using RSA signature authentication, we included the name of our IKE key ring (c), prefaced by the owner of the keyring, IKED. This name correlates to the diagram in Figure 4-10 on page 97.

4.2.3 Creating the IPSec filters and policies for the IPSec tunnel

We have prepared the IKE processes for the two IPSec partners: one on SC32 and one on SC33. Now we need to prepare the IPSec filter definitions and policies for the two systems. We use the same policies that were configured in Chapter 9, “IP Security” on page 373.

Consult the following sections of that chapter:

- “Updating the TCP/IP stack to activate IPSec” on page 378
- “Restricting the use of the ipsec command” on page 379
- “In the following sections, we explain these steps in more detail.” on page 378
- “Installing the IBM Configuration Assistant for z/OS” on page 379
- “Defining the IPSec policies to PAGENT” on page 380

In our scenario for RSA signature mode, we need to make a change to the Configuration Assistant files, as explained in the following section.

4.2.4 Modifying existing policies to use RSA signature mode

If you have built a scenario similar to that defined in Chapter 9, “IP Security” on page 373, you can modify that policy by editing the existing IPSec Policy File, or you can change the policy with the help of the Configuration Assistant GUI. We chose to import the existing Configuration Assistant backing store file and to make the necessary changes there.

We followed the instructions in Chapter 5, “Policy Agent” on page 221 for importing an existing backing store file (see 5.3, “Backup and migration considerations” on page 240.)

The IPSec implementation between SC32_TCPIPA and for SC33_TCPIPE described in Chapter 9, “IP Security” on page 373, is defined in the backing store file named Between-2zOS. Our plan was to keep that file as our backup and to create a new backing store file that includes RSA signature mode.

Therefore, we created a new backing store in the IBM Configuration Assistant. From the Main Perspective, we executed the following steps from the File pull-down:

```
File --> Open --> Create New Backing Store
```

We provided the name `Between-2zOS-RSA.backingstore`. See Figure 4-11 on page 105.

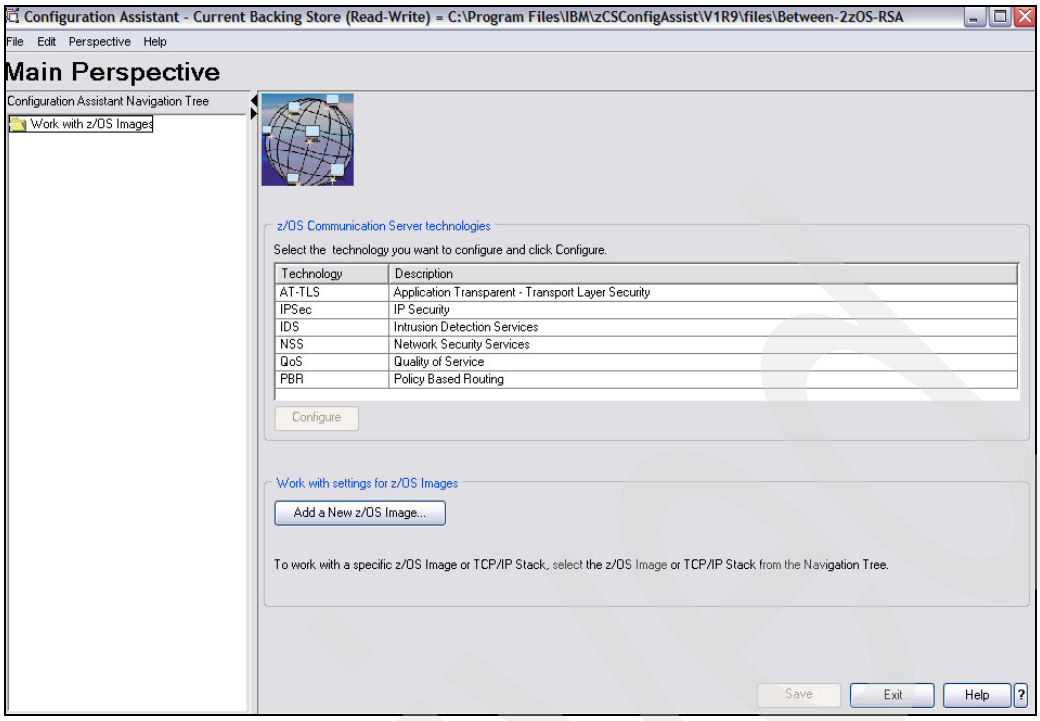


Figure 4-11 Creating a new backing store file for RSA signature mode: Between-2zOS-RSA

Next, we imported the old backing store file, named `Between-2zOS.backingstore`, into the new file. From the Configuration Assistant, we selected:

`File --> Import`

This presents a panel like Figure 4-12 on page 106. We selected **Browse** from this panel to point to the old backingstore file.

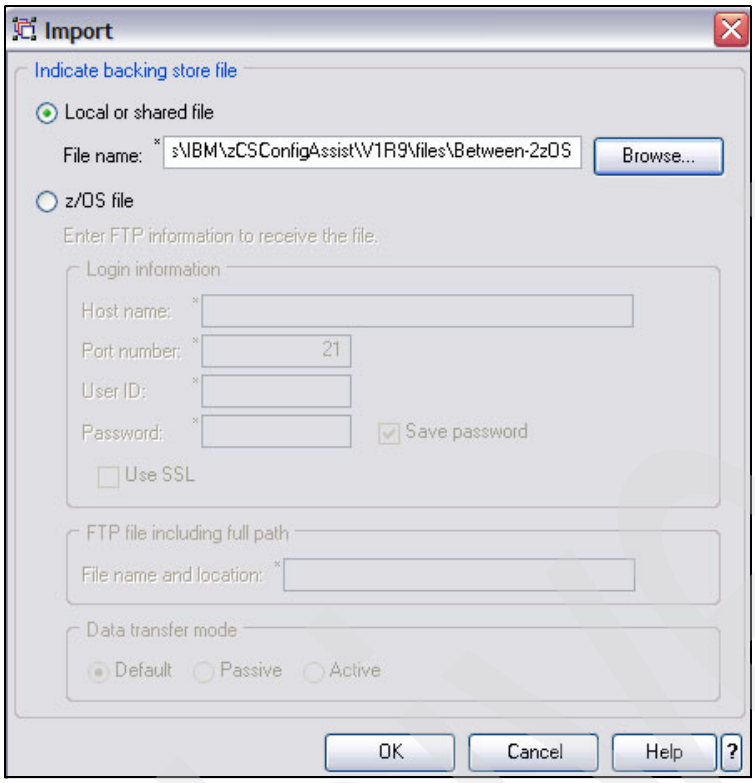


Figure 4-12 Importing previous backingstore file into new file for RSA work

After selecting the old backingstore we clicked **OK**, which took us to the following panel.

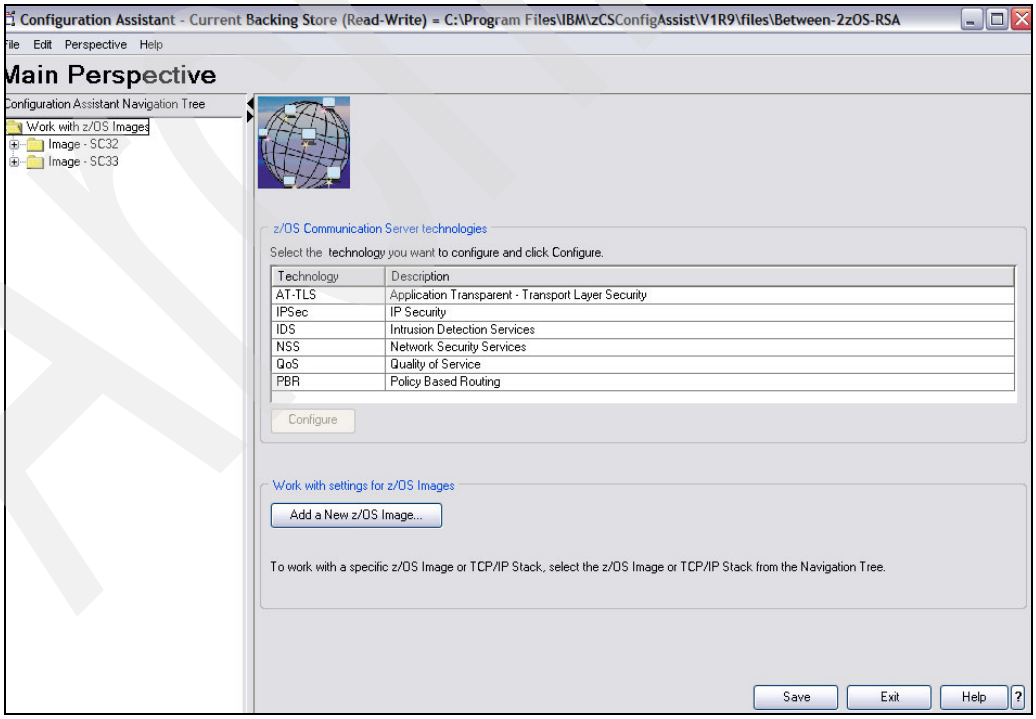


Figure 4-13 Successful import of previous backingstore into Between-2zOS-RSA.backingstore

From here we needed to change our IPSec policy for TCPIPA on SC32 and for TCPIPD on SC33 to one that supports RSA signature mode. Therefore, we expanded the Main Perspective panel so that we could highlight TCPIPA in the left menu and IPSec in the Technology menu in the center; see Figure 4-14.

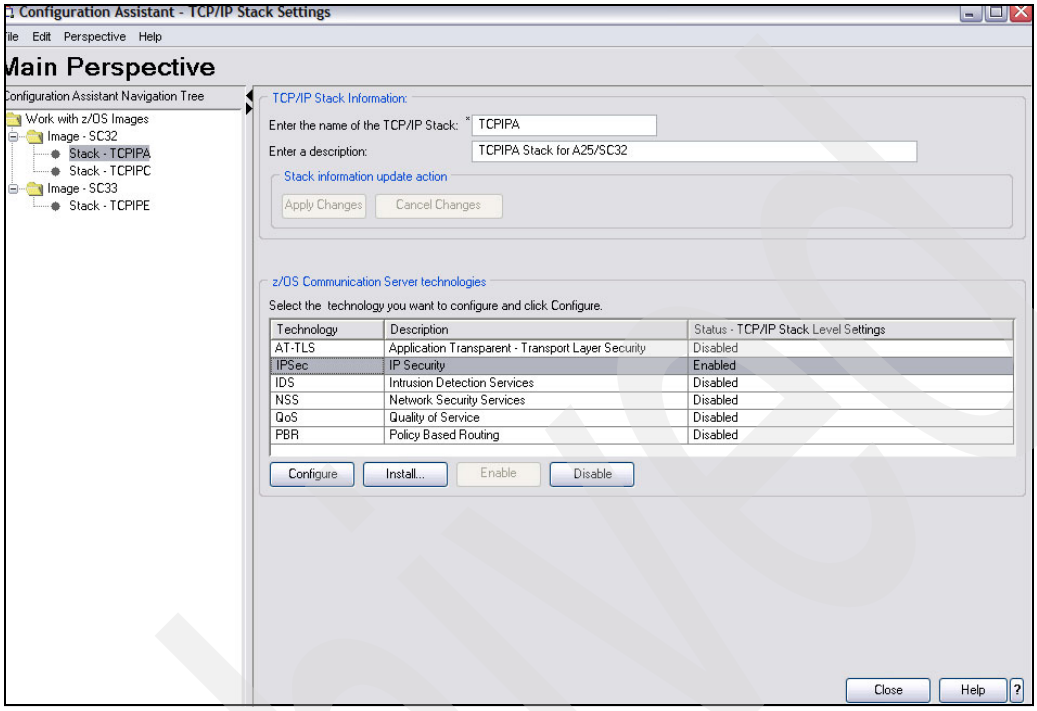


Figure 4-14 Selecting TCP/IP stack and policy that needs revisions

After we selected the entries we needed to revise (TCPIPA and IPSec), we clicked **Configure** on the Main Perspective panel. This presented the IPSec Perspective for TCPIPA, showing the Connectivity Rules tab; see Figure 4-15 on page 108.

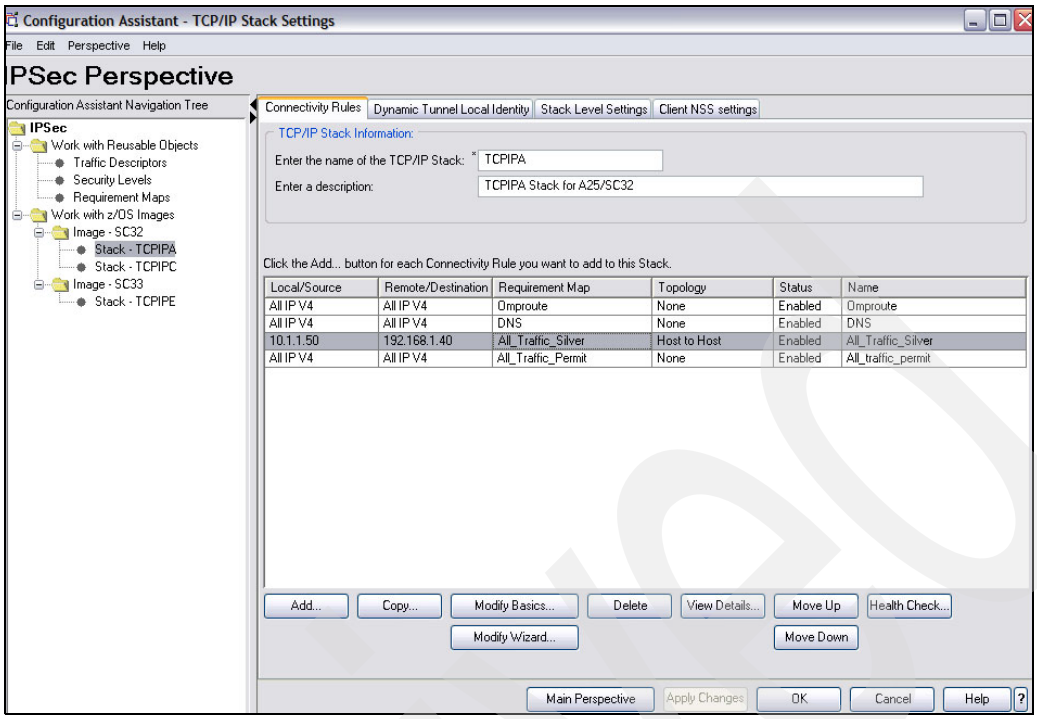


Figure 4-15 IPSec Perspective for TCPIPA

We selected **All_Traffic_Silver**, as shown in Figure 4-15. Then we clicked **Modify Basics**. Our next step is to select the tab **IPSec: Remote Security Endpoint**; see Figure 4-16.

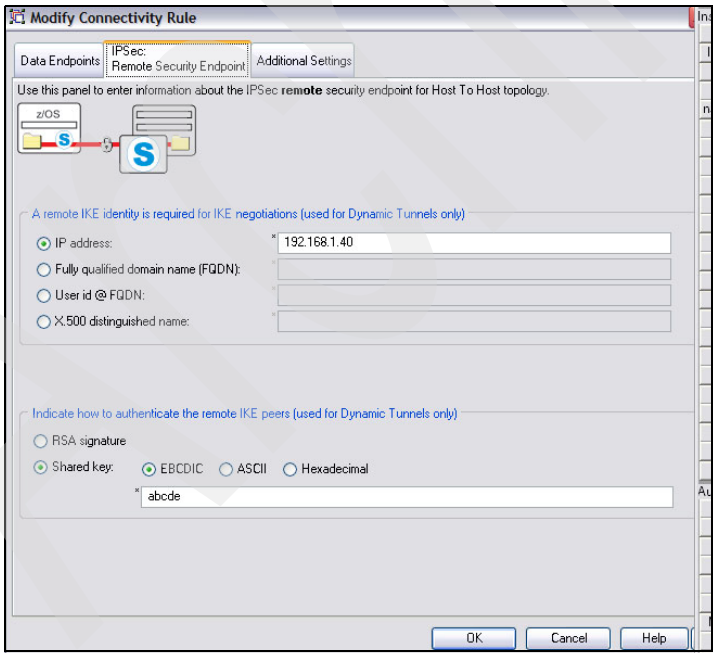


Figure 4-16 Previous IKE mode must be changed from pre-shared key to RSA signature mode

We observed in Figure 4-16 that the authentication of remote IKE peers in the imported configuration was to take place with pre-shared keys. We needed to change this selection to RSA signature mode for our scenario.

We also observed that the local identity is depicted by means of an IP address in the previous, pre-shared key scenario. RSA signature mode requires authentication through the exchange of identities that have been coded in an x.509 certificate. Note that any one of four different identities can be selected:

1. IP address

   This is equivalent to the RACF Subject *ALTNAME* field, *IP*, shown previously in this chapter.

2. Fully qualified domain name (FQDN)

   This is equivalent to the RACF Subject *ALTNAME* field, *DOMAIN*, shown previously in this chapter.

3. User id @ FQDN

   This is equivalent to the RACF Subject *ALTNAME* field, *EMAIL*, shown previously in this chapter.

4. x.500 distinguished name

   This is the full name of the Subject and not an alternate name.

The first three identities correspond to RACF definitions that we failed to include in our certificate definitions. We demonstrate how to code these alternate identities with the RACF parameter ALTNAME in Example 4-3 on page 100.

Recall that our certificate definition in Example 4-1 on page 98 did not include an IP address as an identity. In fact, we coded no Subject Alternate Name in our racdcert execution. Therefore, we cannot use IP address as our RSA identity in the Configuration Assistant panel depicted in Figure 4-16 on page 108.

In our certificate definition, we used only the fields CN, OU, and C. The combination of these Relative Distinguished Name (RDN™) fields creates the x.500 distinguished name. The x.500 distinguished name field represents a valid identity for an IKE connection; therefore, this is the identity that we will use for our scenario with RSA signature mode.

Note: There are many RDNs that can be used to build the x.500 distinguished name. Review Table 4-1 on page 155 for a list of other valid RDNs.

You can display the x.500 distinguished name field with a RACF command that lists the contents of a certificate stored in the RACF database with the label IKE Daemon on SC32:

```
racdcert id(IKED) list(label('IKE Daemon on SC32'))
```

The display output depicts the x.500 name as the `Subject's Name` at a in Example 4-12.

Example 4-12 Display of certificate for SC32

```
Digital certificate information for user IKED:

  Label: IKE Daemon on SC32
  Certificate ID: 2QTJOsXEydLFQMSBhZSWlUCWlUDiw/Py
  Status: TRUST
***
Start Date: 2007/09/20 23:00:00
End Date:   2008/09/21 22:59:59
Serial Number:
     >01<
Issuer's Name:
     >CN=ITSO z/OS CS.O=I.B.M Corporation.C=US<
```

```
 Subject's Name:
      >CN=IKE Daemon on SC32.OU=ITSO.C=US<
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
>
   Ring Owner: IKED
   Ring:
      >IKED32_keyring<
```

Note that the x.500 distinguished name is `CN=IKE Daemon on SC32.OU=ITSO.C=US`. The sequence is very important. The RACF definition may list the fields of the x.500 distinguished name in any sequence. However, that sequence is altered when the definition is stored in the RACF database and the RACF sequence is used for the IKE identity field.

Important: The racdcert display separates the individual fields (also known as Relative Distinguished Names or RDN) of the Distinguished Name (DN) with periods as delimiters. The display pattern is:

RDN<period>RDN<period>RDN<period>RDN

However, you will see that the syntax required in the IKE definition panel (Figure 4-17 on page 110) requires a comma (,) as a delimiter. The coding pattern in this case is:

RDN<comma>RDN<comma>RDN<comma>RDN

Therefore, the display output of CN=IKE Daemon on SC32.OU=ITSO.C=US must be converted to CN=IKE Daemon on SC32,OU=ITSO,C=US when it is defined as an IKE identity in the IBM Configuration Assistant. (This GUI builds the iked.conf file from these definitions.)

We fill in the fields displayed in Figure 4-17 accordingly.

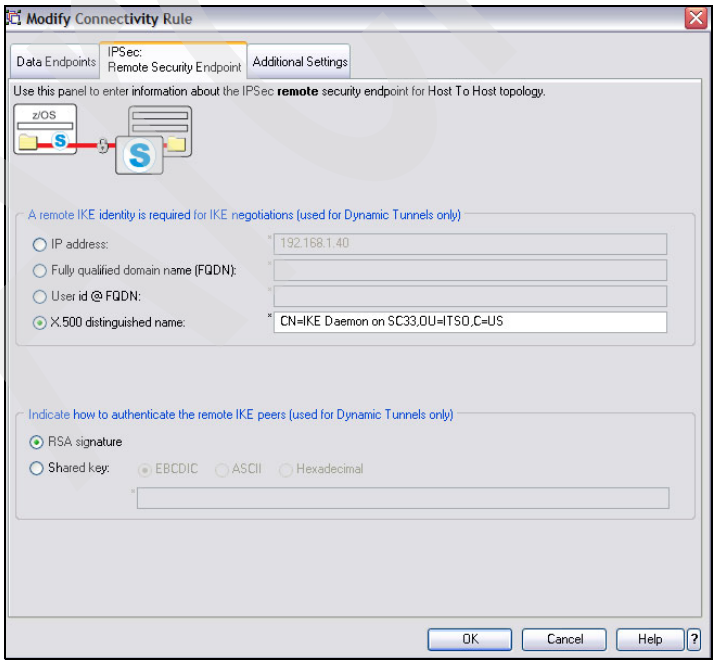


Figure 4-17 RSA signature mode now selected

In Figure 4-17 on page 110, note how the periods from the racdcert display of the certificate have been replaced with commas.

We pressed **OK,** which returned us to the Connectivity Rules panel. We selected the tab for Dynamic Tunnel Local Identity and changed the identity from `IP address` to `x500dn`; see Figure 4-18.

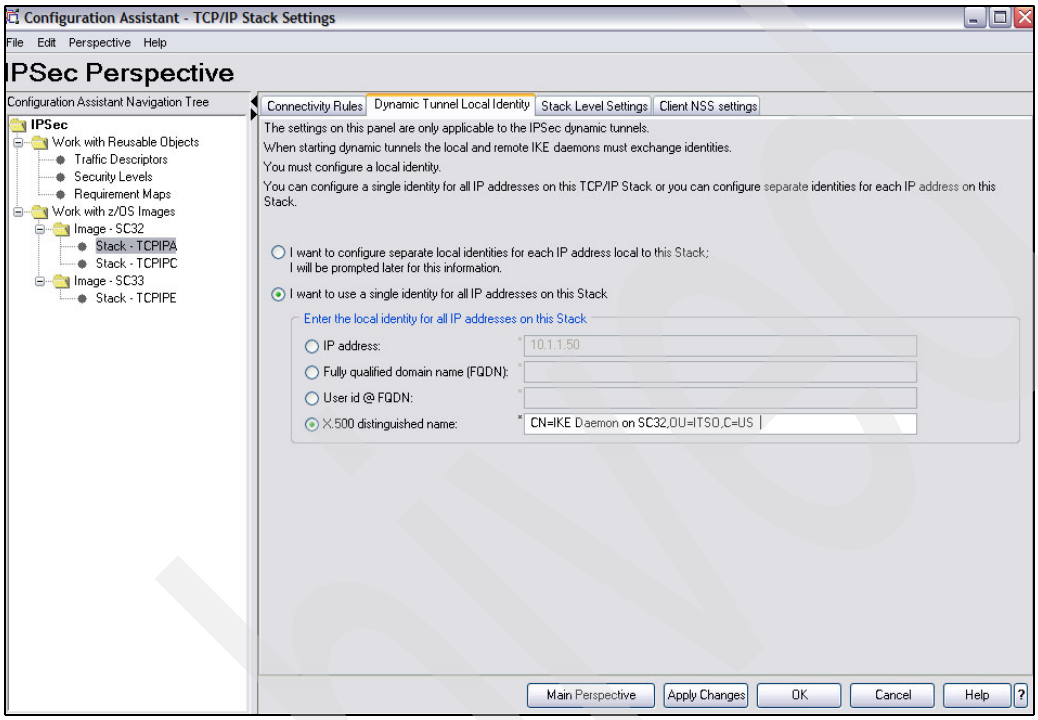


Figure 4-18 Defining the local identity of the IPSec endpoint for SC32

We have chosen to identify the local tunnel endpoint by the x.500 distinguished name from the certificate for SC32. (If we had specified an ALTNAME in the certificate generation, we could have used that instead.) We fill in the field value with `CN=IKE Daemon on SC32,OU=ITSO,C=US`. This value was coded in our certificate definition and we used the sequence that is displayed in the certificate subject name when we issue the command:

```
racdcert id(IKED) list(label('IKE Daemon on SC32'))
```

The output at a in Example 4-13 shows you the certificate Subject's Name. It is in the order that was entered in the `x.500dn` field of Figure 4-18.

Example 4-13 Display of certificate for SC32

```
Digital certificate information for user IKED:

  Label: IKE Daemon on SC32
  Certificate ID: 2QTJOsXEydLFQMSBhZSWlUCWlUDiw/Py
  Status: TRUST
***
Start Date: 2007/09/20 23:00:00
End Date:   2008/09/21 22:59:59
Serial Number:
     >01<
Issuer's Name:
     >CN=ITSO z/OS CS.O=I.B.M Corporation.C=US<
Subject's Name:
```

```
     >CN=IKE Daemon on SC32.OU=ITSO.C=US<  a
 Private Key Type: Non-ICSF
 Private Key Size: 1024
 Ring Associations:
>
   Ring Owner: IKED
   Ring:
      >IKED32_keyring<
```

Selecting **Apply Changes** from the panel returns you to the IPSec Perspective menu. You will perform the same tasks for TCPIPE; see Figure 4-19.

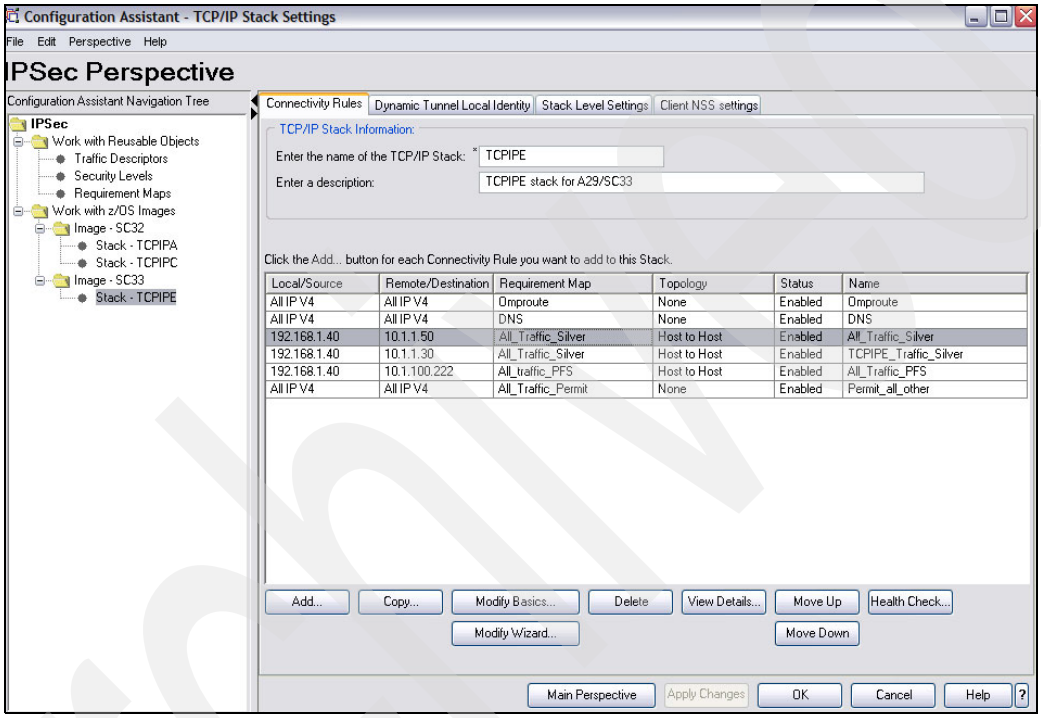


Figure 4-19 Changing IPSec policy for SC33's TCPIPE to SC32's TCPIPA

In Figure 4-19 we have highlighted TCPIPE and the Connectivity Rule that applies to TCPIPA at address 10.1.1.50. We select **Modify Basics** and proceed through the same steps that we executed for TCPIPA on SC32:

- Select **IPSec: Remote Security Endpoint**
  a. Identify the remote endpoint with its x.500 distinguished name and select **RSA signature mode**; see Figure 4-20 on page 113.

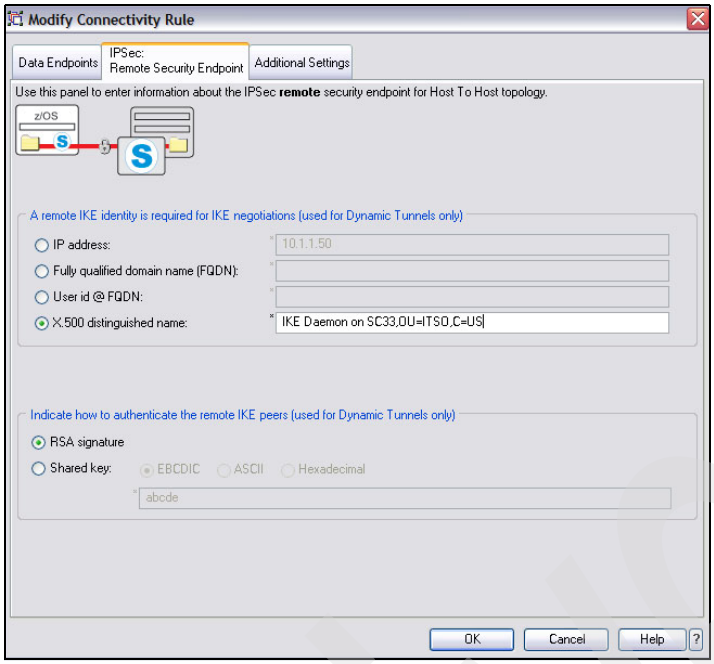


Figure 4-20 Altering remote security endpoint for RSA signature mode

b. Click **OK**.

- Select the tab for Dynamic Tunnel Local Identity and fill in the x.500 distinguished name for TCPIPE on SC33; see Figure 4-21.

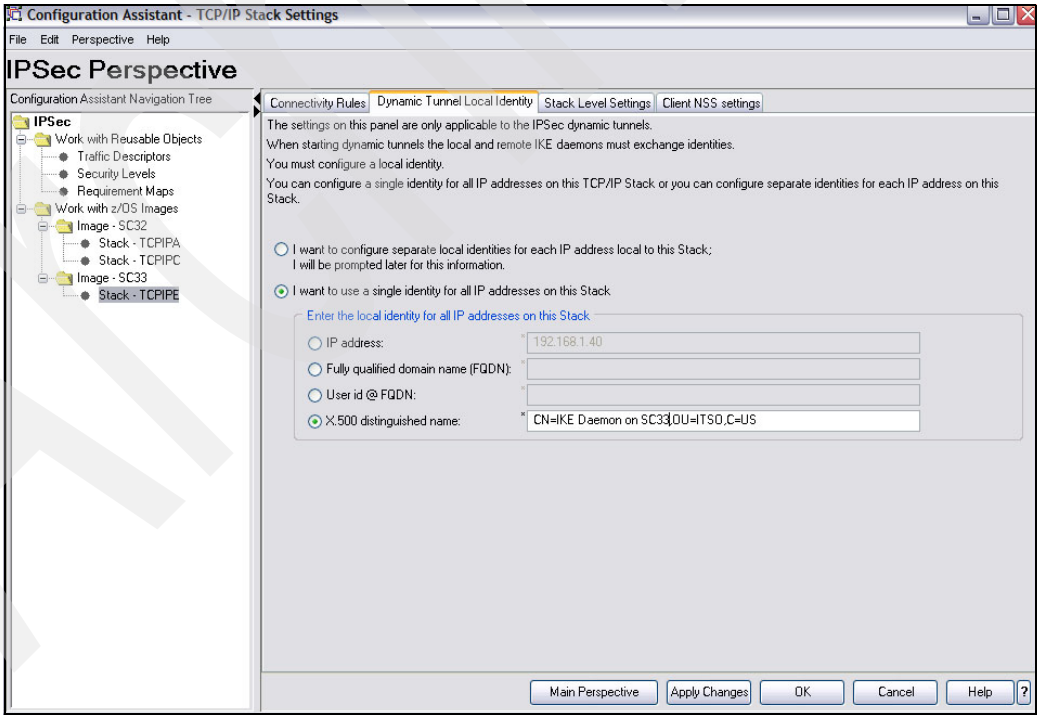


Figure 4-21 RSA signature mode and identity of local IPSec endpoint

Once again, the x.500 distinguished name is taken from the output display of the contents of SC33's certificate that we created earlier:

```
racdcert id(IKED) list(label('IKE Daemon on SC33'))
```

- Select **Apply Changes**.

Building new IPSec policies to use RSA signature mode

We used the existing IPSec policies that we imported from the previous backingstore file. We simply changed the IKE process in that backingstore file from pre-shared key mode to RSA signature mode. If you need to build new IPSec policies because you have no file to import, refer to the description of IPSec policies in Chapter 9, "IP Security" on page 373.

Building the IKE Daemon files for RSA signature mode

In our case, we had already created the IKE Daemon configuration files manually in "Definitions for IKE on SC32" on page 101 and "Definitions for IKE on SC33" on page 102. As mentioned there, you can also create the IKE Daemon configuration files with the help of the IBM Configuration Assistant. We show you how to do this here.

Return to the IPSec Perspective of the IBM Configuration Assistant and select MVS image SC32. This operation presents a panel where we select the tab for the IKE Daemon Settings, as shown in Figure 4-22.

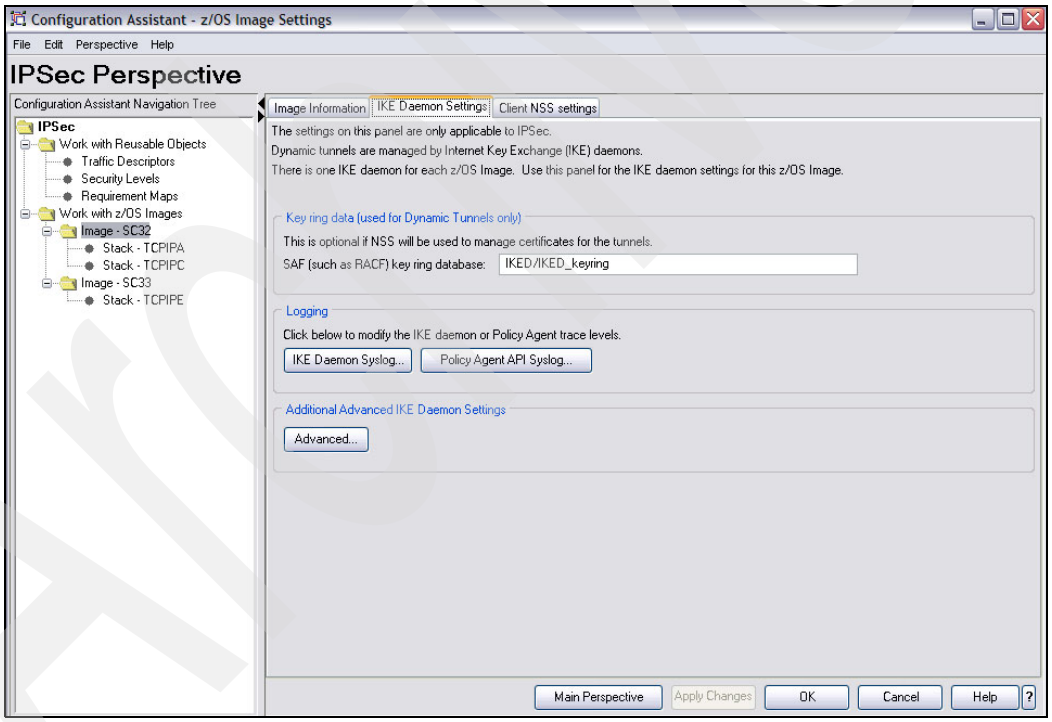


Figure 4-22 Old key ring that was identified but never used with pre-shared key mode

Change the key ring on the panel to reflect the new SC32 keyring that is used for the IKEDC procedure; see Figure 4-23 on page 115.

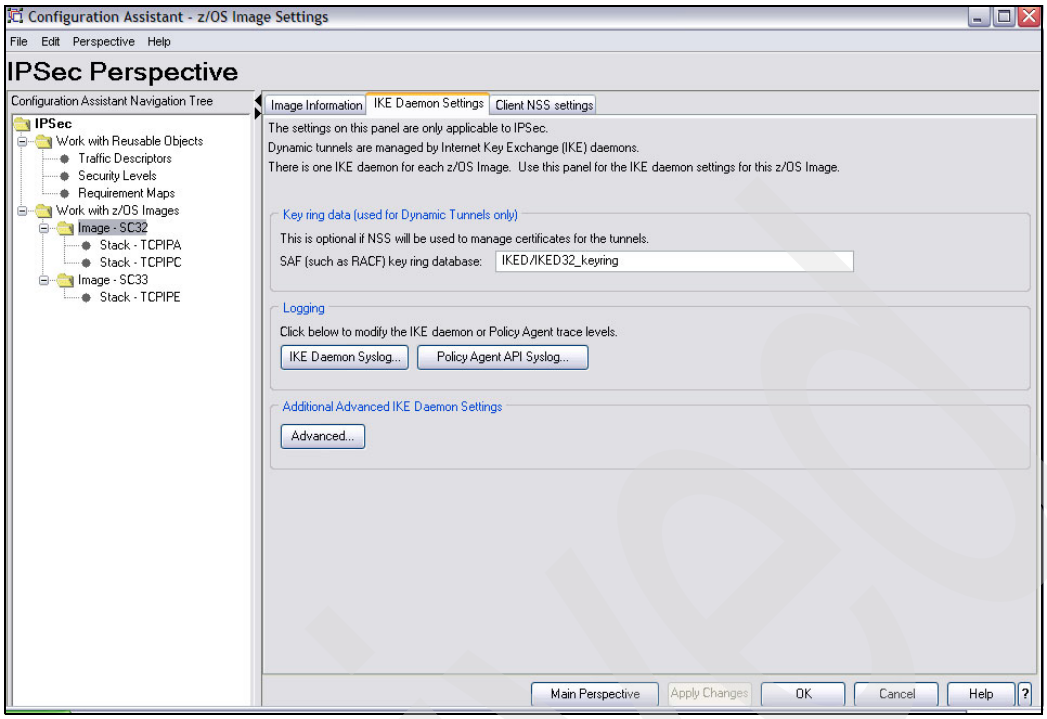


Figure 4-23 Key ring for IKEDC and RSA signature mode at SC32

We replaced the name of the key ring with IKED/IKED32_keyring, which is the key ring we built to contain the x.509 certificates for RSA signature mode. Click **OK**, and then select the SC33 image from the IPSec Perspective. Move to the IKE Daemon Settings panel and again, change the name of the key ring to reflect the ring for RSA signature authentication at IKEDD on SC33; see Figure 4-24.

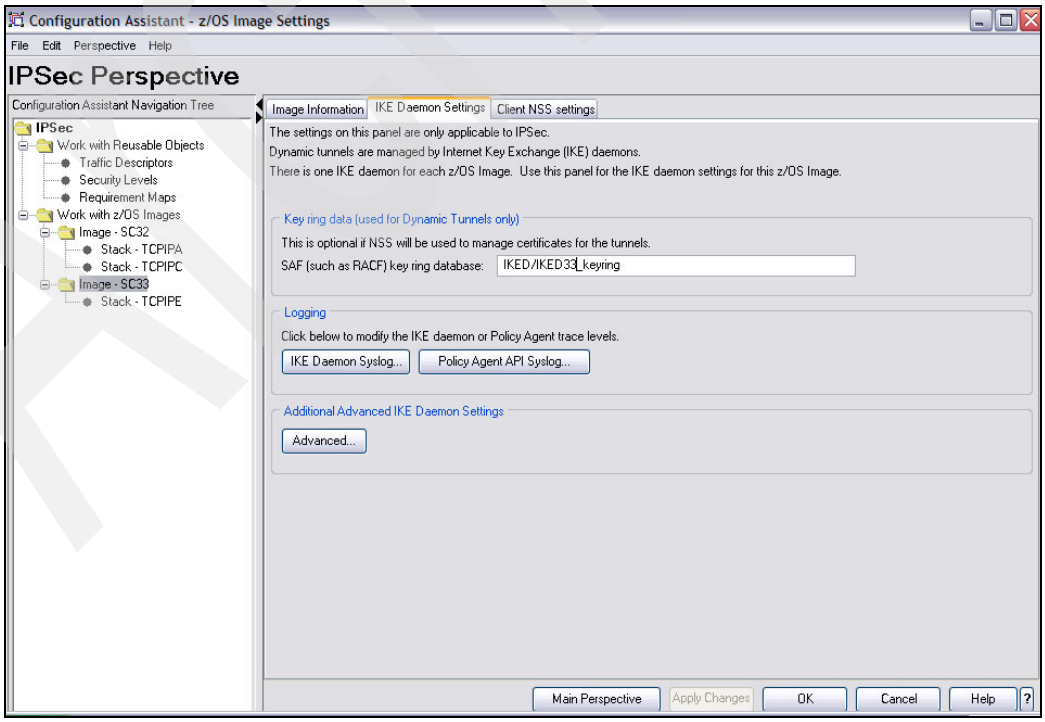


Figure 4-24 Key ring for IKEDD and RSA signature mode at SC33

We replaced the name of the key ring with IKED/IKED33_keyring, which is the key ring we built to contain the x.509 certificates for RSA signature mode at SC33. Click **OK**.

Sending IPSec and IKE configuration files to the mainframe

Here we describe how to send IPSec and IKE configuration files to the mainframe.

Default IPSec policies and Policy Agent IPSec policies

We return to the Main Perspective of the IBM Configuration Assistant and expand the view for SC32. We highlight `TCPIPA`, one of our IKE partners, and also highlight `IPSec`. We select **Install**. This brings up a panel with the files that were changed or added in the IPSec configuration that was altered from pre-shared key mode to RSA signature mode; see Figure 4-25.

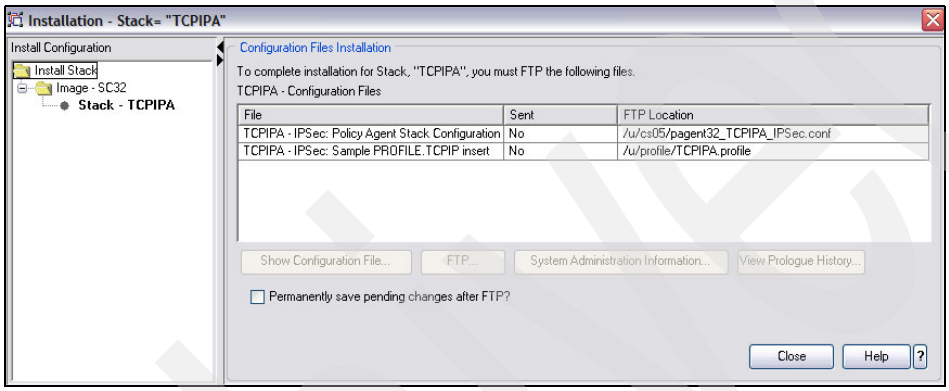


Figure 4-25 Files that were altered during change to RSA signature mode

Note how two files are available to install: the sample file for the default IPSec policy rules that are to be integrated into the running TCP/IP stack, and the IPSec configuration file. Observe that the IPSec policy files were not touched during this process; the same policy files used in Chapter 9, “IP Security” on page 373 are still valid for RSA signature mode.

We highlight the IPSec configuration file and press **Show Configuration File**; see Figure 4-26.

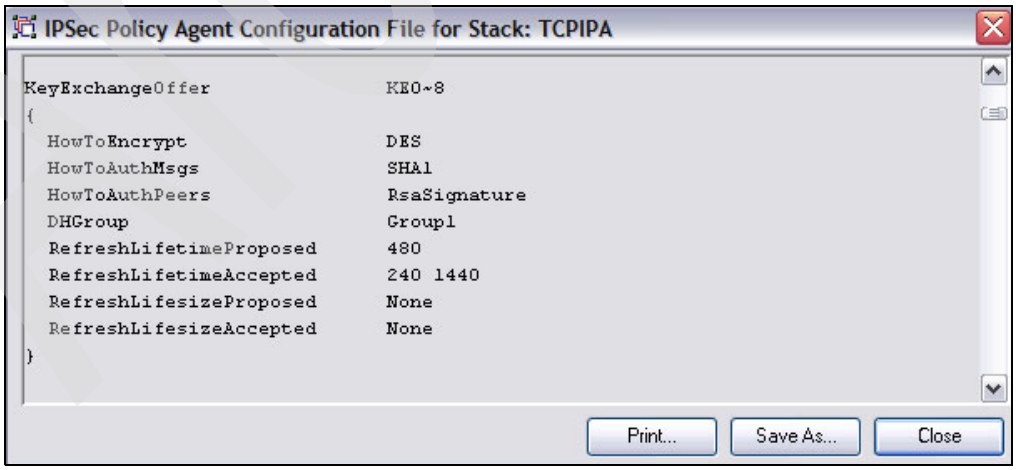


Figure 4-26 TCPIPA: New KeyExchangeOffer KEO~8: Authenticate Peers with RsaSignature

In one portion of the configuration file for the IPSec policy for TCPIPA, we see that there is a KeyExchangeOffer named KEO~8, that designates RsaSignature as the method for IKE peer

authentication. We continue looking at the policy file and find another change to the original policies for TCPIPA; see Figure 4-27.

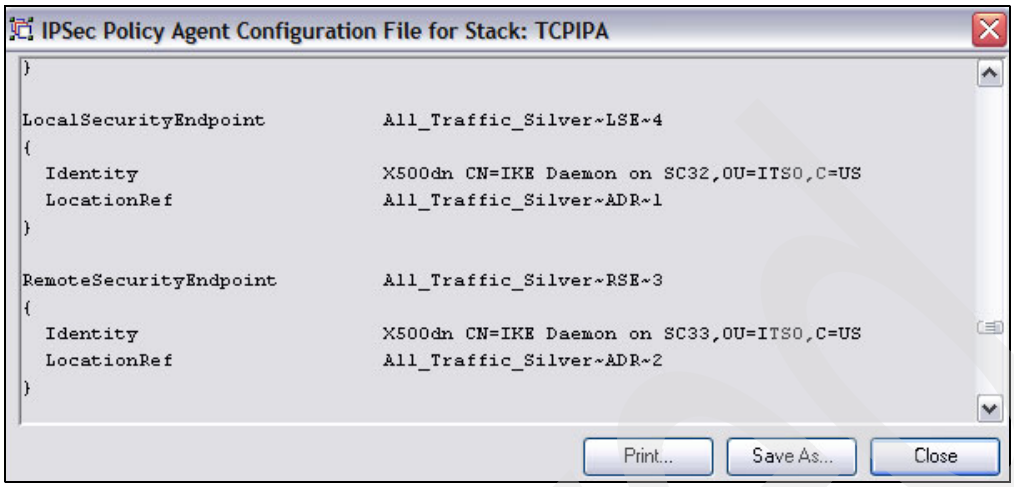


Figure 4-27 SC32's TCPIPA stack: Local and Remote Identity now using x500dn

Observe in Figure 4-27 how the LocalSecurityEndpoint and the RemoteSecurityEndpoint have been changed to reflect a new identity: the `x500dn` value that we extracted from our certificate displays.

We close the view of SC32's files and return to the Main Perspective, where we look at SC33's files. There we see that a new the IKE mode of operation has been added in the KeyExchangeOffer KEO~8 shown in Figure 4-28 on page 118.

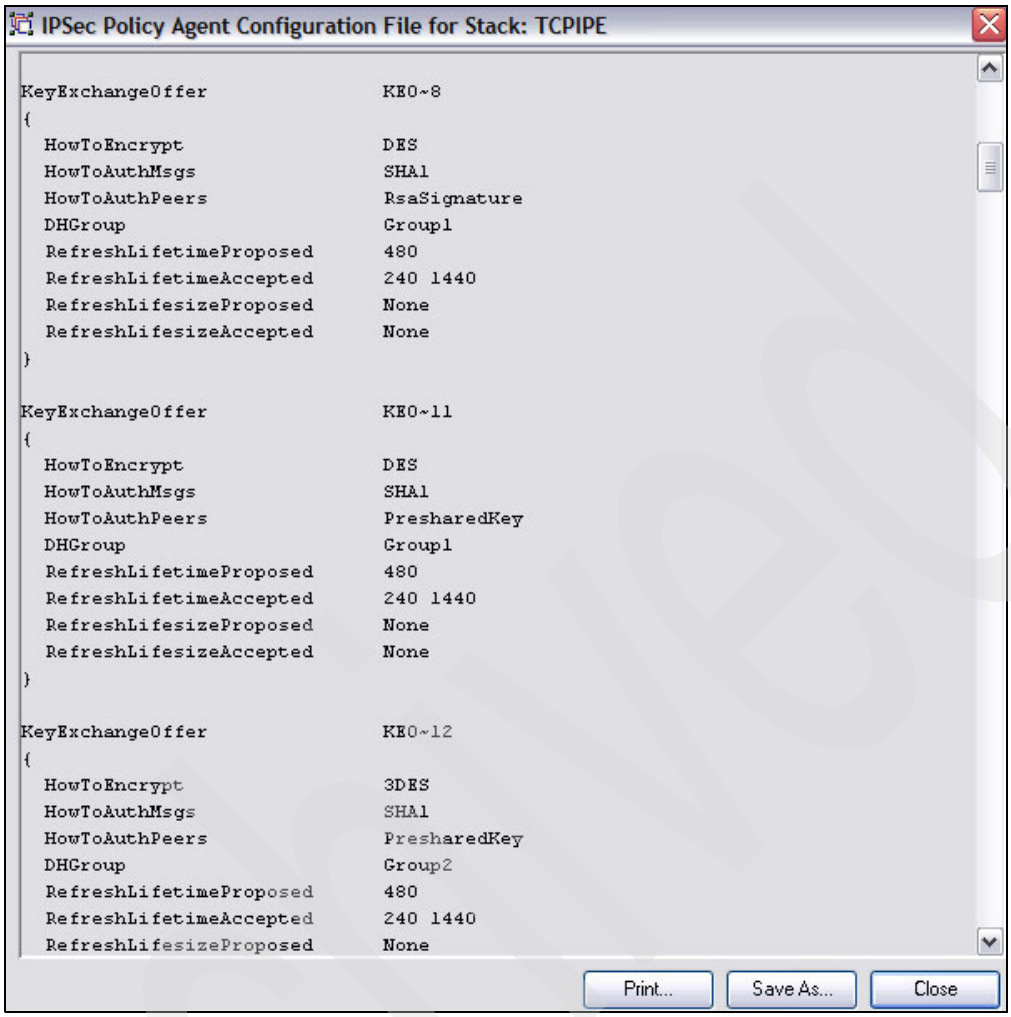


Figure 4-28 TCPIPE: New KeyExchangeOffer KEO~8: Authenticate Peers with RsaSignature

TCPIPE's IPSec policy now includes RsaSignature mode for IKE operation, as you see in the KeyExchangeOffer KEO~8 of Figure 4-28.

We continue to browse through the IPSec policy for TCPIPE and find another section that is different from what was configured when only pre-shared key mode was defined for IKE; see Figure 4-29 on page 119.

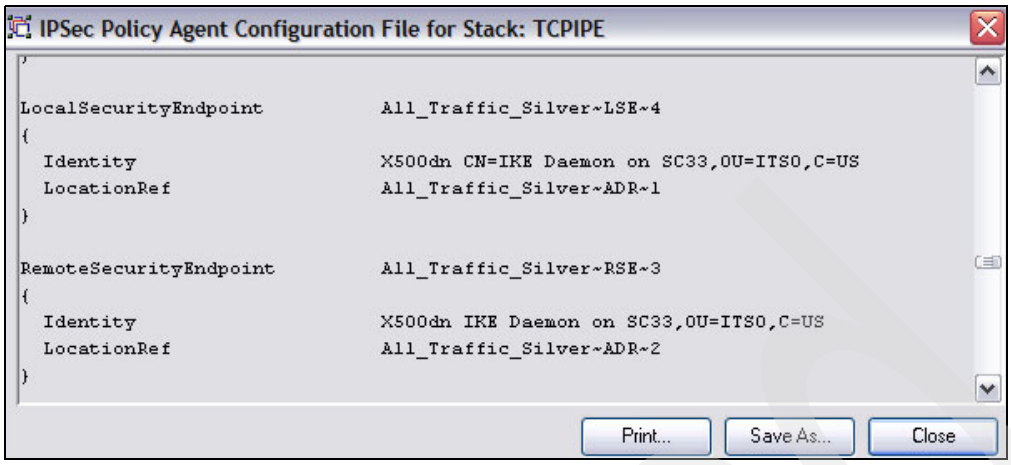


Figure 4-29 SC33's TCPIPE stack: Local and Remote Identity now using x500dn

As with TCPIPA at SC32, the Local and Remote identities are now using the x.500 distinguished name fields to identify the IKE partners. These fields are part of the certificates that each IKE peer will present to the other.

Upon examination we see that the default IPSec filters have not changed since the scenario with pre-shared key mode was executed. Therefore, we FTP only the IPSec policy files to the mainframe, where we point to them from the respective TCPIPA and TCPIPE image files.

At TCPIPA on SC32, we change the image file to look as shown in Example 4-14.

Example 4-14 /etc/pagent32_TCPIPA.conf

```
# ###################################################################
#                  IMAGE FILE FOR Stack TCPIPA on LPAR A25/SC32
# ###################################################################
#
#
IPSecConfig  /u/cs02/TCPIPA.ipsecRSAPolicy
...
```

You see in Example 4-14 that we are now pointing to the IPSec policy that we sent to the mainframe in a previous step.

At TCPIPE on SC33, we also change the policy agent image file to point to the RSA policy file, as shown in Example 4-15.

Example 4-15 /etc/pagent33_TCPIPE.conf

```
# ###################################################################
#                  IMAGE FILE FOR TCPIPE on LPAR A29
#                  TTLS supports the NSS function here
# ###################################################################
#
#
IPSecConfig  /u/cs02/TCPIPE.ipsecRSAPolicy
...
```

IKE Daemon configuration files

Previously we explained two ways to create the IKE Daemon configuration files:

- Manually, by editing at the mainframe (“Definitions for IKE on SC32” on page 101 and “Definitions for IKE on SC33” on page 102)
- Automatically by using the IBM Configuration Assistant (“Building the IKE Daemon files for RSA signature mode” on page 114).

So keep in mind that the IKE Daemon file needs to be modified when you convert to RSA signature mode. Specifically, you must at least point to the correct key ring in which the x.509 certificates for the RSA process are stored. We named our IKE configuration file for IKE with RSA as follows:

- SC32: iked32RSA.conf
- SC33: iked33RSA.conf

4.2.5 Verifying IKE with RSA signature mode

This section shows the different areas that need to be verified to ensure that IPSec is operational when using RSA signature mode.

Initialize TRMD, IKED, and PAGENT at SC32 and SC33

We have already revised our IKEDC, IKEDD, and Pagent policies at SC32 and SC33.

To verify IKE with the new RSA signature mode, we stop both IKE and Policy Agent on each stack and start them up again with the new definitions. Perform the following steps.

1. Start PAGENT, TRMD, and IKEDC at SC32; see the console log in Example 4-16.

Example 4-16 Console log for PAGENT, TRMDA32, and IKEDC at SC32

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT WITH JOBNAME PAGENT IS ASSIGNED TO USER  PAGENT , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IPSEC a
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : TTLS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPC : IPSEC
EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER
S TRMDA32
$HASP100 TRMDA32  ON STCINRDR
IEF695I START TRMDA32  WITH JOBNAME TRMDA32  IS ASSIGNED TO USER TRMD
   , GROUP TCPGRP
$HASP373 TRMDA32  STARTED
S IKEDC
$HASP100 IKEDC    ON STCINRDR
IEF695I START IKEDC  WITH JOBNAME IKEDC  IS ASSIGNED TO USER IKED , GROUP TCPGRP
$HASP373 IKEDC    STARTED
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R9 SERVICE LEVEL CS070402 CREATED ON Apr 3  2007
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/security/iked32RSA.conf b
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPA   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
EZD1058I IKE STATUS FOR STACK TCPIPC   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPC
```

```
EZD1133I IKE STATUS FOR STACK TCPIP  IS ACTIVE WITHOUT IPSECURITY SUPPORT
EZD1046I IKE INITIALIZATION COMPLETE
```

- At **a**, you see that IPSec is implemented for TCPIPA.
- At **b** you see that we are using the new IKED configuration file, that is, the one we built for RSA signature mode.

2. Initialize PAGENT, TRMDE33, and IKEDD at SC33; see the console log in Example 4-17.

Example 4-17 Console log for Pagent, TRMDE33, IKEDD at SC33

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT WITH JOBNAME PAGENT IS ASSIGNED TO USER  PAGENT , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : QOS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : IPSEC   a
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : TTLS
EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER
S TRMDE33
$HASP100 TRMDE33 ON STCINRDR
IEF695I START TRMDE33 WITH JOBNAME TRMDE33 IS ASSIGNED TO USER TRMD
   , GROUP TCPGRP
$HASP373 TRMDE33 STARTED
S IKEDD
$HASP100 IKEDD    ON STCINRDR
IEF695I START IKEDD  WITH JOBNAME IKEDD  IS ASSIGNED TO USER IKED , GROUP TCPGRP
$HASP373 IKEDD    STARTED
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R9 SERVICE LEVEL CS070402 CREATED ON Apr 3  2007
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/security/iked33RSA.conf b
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPE   IS UP
EZD1058I IKE STATUS FOR STACK TCPIPE   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPE
EZD1133I IKE STATUS FOR STACK TCPIP  IS ACTIVE WITHOUT IPSECURITY SUPPORT
EZD1133I IKE STATUS FOR STACK TCPIPD  IS ACTIVE WITHOUT IPSECURITY SUPPORT
EZD1046I IKE INITIALIZATION COMPLETE
```

- At **a**, you see that IPSec is implemented for TCPIPE.
- At **b**, you see that we are using the new IKED configuration file, that is, the one we built for RSA signature mode.

Review SYSLOG daemon logs for information about the initialization

Review the IKE log to determine whether the user (PRIVATE) certificates for SC32 and SC33 have been found during IKE initialization; look at Example 4-18.

Example 4-18 The certificate caches for SC32

```
IKE: Initializing CA Cache
IKE: Begin display of CA Cache
IKE: End display of CA Cache
IKE: Initialization of CA Cache Complete
IKE:
IKE: Initializing Cert Cache
```

```
IKE: IKE CERTINFO : Unable to add certificate to cert cache : No private key My Local
Certificate Authority
IKE: Begin display of Cert Cache    a
IKE: Certificate cache contains certificate with label IKE Daemon on SC32   b
IKE: End display of Cert Cache
IKE: Initialization of Cert Cache Complete
```

At a, you see the beginning of the display of the user certificates that are used to identify local and remote endpoints. At b, you see that the certificate for the IKE Daemon on SC32 is being recognized for IKE processing.

At SC33, investigate the contents of the IKE log, as shown in Example 4-19.

Example 4-19 The certificate caches for SC33

```
IKE: Initializing CA Cache
IKE: Begin display of CA Cache
IKE: End display of CA Cache
IKE: Initialization of CA Cache Complete
IKE:
IKE: Initializing Cert Cache
IKE: IKE CERTINFO : Unable to add certificate to cert cache : No private key My Local
Certificate Authority
IKE: Begin display of Cert Cache
IKE: Certificate cache contains certificate with label IKE Daemon on SC33 c
IKE: End display of Cert Cache
IKE: Initialization of Cert Cache Complete
```

At c, the IKE daemon reveals that it has successfully processed the certificate that identifies the IKE Daemon on SC33.

Note: If SupportedCertAuth was coded in the IKE configuration file, the display of the CA cache for IKE would have listed 'My Local Certificate Authority' in the output shown in Example 4-18 and Example 4-19.

Issue a **pasearch** command to determine if the proper IPSec policy Key Exchange rule was loaded:

```
pasearch -p TCPIPA -v k
```

Example 4-20 IPSec policy Key Exchange rule on SC32

```
CS02 @ SC32:/u/cs02>pasearch -p TCPIPA -v k

TCP/IP pasearch CS V1R9                  Image Name: TCPIPA
  Date:                 03/03/2008       Time:  18:35:39
  IPSec Instance Id:    1204580840

policyRule:             All_Traffic_Silver~5
  Rule Type:            KeyExchange
  Version:              3                Status:            Active
  Weight:               101              ForLoadDist:       False
  Priority:             1                Sequence Actions:  Don't Care
  No. Policy Action:    1
  IpSecType:            policyKeyExchange
  policyAction:         All_Traffic_Silver
   ActionType:          KeyExchange
   Action Sequence:     0
  Time Periods:
```

```
   Day of Month Mask:   00000000000000000000000000000000
   Month of Yr Mask:    000000000000
   Day of Week Mask:    0000000  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:      00:00
   Fr TimeOfDay UTC:    00:00             To TimeOfDay UTC:  00:00
   TimeZone:            Local
  TimeZone:            Local
 IpSec Condition Summary:                NegativeIndicator: Off
  KeyExchange Condition:
   LocalSecurityEndPoint:
    Location:          10.1.1.50     a
    Identity:
     x500dn:
      CN=IKE Daemon on SC32,OU=ITSO,C=US b
   RemoteSecurityEndPoint:
    Location:
     FromAddr:         192.168.1.40 c
     ToAddr:           192.168.1.40
    Identity:
     x500dn:
      CN=IKE Daemon on SC33,OU=ITSO,C=US d

 KeyExchange Action:   All_Traffic_Silver
   Version:            3                 Status:           Active
   HowToInitiate:      Main              HowToRespond:     Either
   AllowNat:           No
   KeyExchangeOffer:   0
    HowToEncrypt:      DES               HowToAuthMsgs:    SHA1
    HowToAuthPeers:    RsaSignature e    DHGroup:          Group1
    RefLifeTmPropose:  480
    RefLifeTmAcptMin:  240               RefLifeTmAcptMax: 1440
    RefLifeSzPropose:  None
    RefLifeSzAccept :  None
```

When you examine the **pasearch** output, you recognize that your IPSec policies accurately reflect the intended environment:

a The local security endpoint address is correct.

b The Identity that was extracted from the certificate is accurately reflected, with commas (not periods) between the individual RDNs.

c The remote security endpoint IP address is also correct.

d The identity that was extracted from the certificate of the remote partner is also accurately reflected, again with commas and not periods in the x500dn string.

e The authentication of the peers is to take place with RsaSignature mode, a change from the previous scenarios, in which pre-shared key was used.

At SC32's OMVS shell, ping the TCPIPE stack from TCPIPA in order to establish a connection across the VPN:

```
ping -p TCPIPA 192.168.1.40
```

Example 4-21 Bringing up a dynamic tunnel between TCPIPA and TCPIPE

```
CSO2 @ SC32:/u/cs02>ping -p TCPIPA 192.168.1.40
```

```
CS V1R9: Pinging host 192.168.1.40
sendto(): EDC5111I Permission denied. (errno2=0x74420291) a
CSO2 @ SC32:/u/cs02>ping -p TCPIPA 192.168.1.40
CS V1R9: Pinging host 192.168.1.40
Ping #1 response took 0.001 seconds.      b
CSO2 @ SC32:/u/cs02>
```

Note how, in Example 4-21 on page 123, the first ping (a) does not succeed.

Why the connection fails at the first ping command: Our IPSec policy has been specified with "OnDemand." ICMP commands, like **ping**, other RAW commands, and UDP protocols, do not have retry logic. Therefore, the first **ping** activates the OnDemand tunnel, making the tunnel ready for the second ping command, which succeeds. TCP flows *do* have retry logic and thus bring up the tunnel so that the subsequent retry succeeds without any error messages.

Other options for activating a dynamic VPN tunnel are manual, with a command, and autoactivate, which initiates the tunnel when TCP/IP and IKE are initialized (and before a user or program may need the tunnel available).

The second **ping** (b) does succeed; it appears that we have established a connection across the VPN. Therefore, we display the dynamic IKE tunnel from SC32:

```
ipsec -p TCPIPA -k display
```

Example 4-22 Display of the IKE connection between SC32 and SC33

```
CSO2 @ SC32:/u/cs02>ipsec -p TCPIPA -k display

CS V1R9 ipsec  Stack Name: TCPIPA  Mon Mar  3 18:14:56 2008
Primary:  IKE tunnel      Function: Display          Format:   Detail
Source:   IKED            Scope:    Current          TotAvail: n/a

TunnelID:                     K1      a
KeyExchangeRuleName:          All_Traffic_Silver~5
KeyExchangeActionName:        All_Traffic_Silver
LocalEndPoint:                10.1.1.50
LocalIDType:                  X500DN    b
LocalID:                      CN=IKE Daemon on SC32,OU=ITSO,C=US
RemoteEndPoint:               192.168.1.40
RemoteIDType:                 X500DN    c
RemoteID:                     CN=IKE Daemon on SC33,OU=ITSO,C=US
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup:           1
AuthenticationMethod:         RSASignature   d
InitiatorCookie:              0X7BA511600C724FA9
ResponderCookie:              0XAD7707FC009BB784
Lifesize:                     0K
CurrentByteCount:             312b
Lifetime:                     480m
LifetimeRefresh:              2008/03/03 23:34:55
LifetimeExpires:              2008/03/04 01:56:06
Role:                         Initiator   e
AssociatedDynamicTunnels:     1
NATTSupportLevel:             None
NATInFrntLclScEndPnt:         No
```

```
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     No
RmtNAPTDetected:              No
RmtUdpEncapPort:              n/a
*******************************************************************************
```

Example 4-22 on page 124 shows several important pieces of information:

a K1 is the identifier for the IKE tunnel.

b The local idtype in use for RSA authentication is the x500dn field.

c The remote idtype in use for RSA authentication is also the x500dn field.

d The authentication method is RSA signature.

e The SC32 side of the IKE tunnel initiated the tunnel.

We display the dynamic tunnel over which the **ping** flowed:

```
ipsec -p TCPIPA -y display
```

Example 4-23 Display the underlying dynamic tunnel between SC32 and SC33

```
CSO2 @ SC32:/u/csO2>ipsec -p TCPIPA -y display

CS V1R9 ipsec  Stack Name: TCPIPA  Mon Mar  3 18:18:51 2008
Primary:  Dynamic tunnel  Function: Display          Format:   Detail
Source:   Stack           Scope:    Current          TotAvail: 1

TunnelID:                     Y2      a
ParentIKETunnelID:            K1      b
VpnActionName:                IPSec__Silver
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport    c
LocalEndPoint:                10.1.1.50
RemoteEndPoint:               192.168.1.40
LocalAddressBase:             10.1.1.50
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
RemoteAddressBase:            192.168.1.40
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP  d
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              3487020753
 AuthOutboundSpi:             1312697222
HowToEncrypt:                 DES  e
 EncryptInboundSpi:           3487020753
 EncryptOutboundSpi:          1312697222
Protocol:                     ALL(0)   f
LocalPort:                    0
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              1
OutboundBytes:                264
InboundPackets:               1
InboundBytes:                 264
Lifesize:                     0K
```

```
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2008/03/03 21:17:17
LifetimeExpires:              2008/03/03 21:56:06
CurrentTime:                  2008/03/03 18:18:51
VPNLifeExpires:               2008/03/04 17:56:06
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
************************************************************************
```

Example 4-23 on page 125 shows multiple important pieces of information:

a Y2 is the identifier for the dynamic tunnel that carried the **ping** command.

b K1 is the IKE tunnel over which was negotiated the phase 2 security association.

c Transport mode is in use for the host-to-host connection.

d Encapsulating Security Protocol (ESP™) is in use, because we are authenticating and encrypting the data.

e The data on this IPSec connection is subject to DES encryption.

f All protocols (TCP, UDP, ICMP, and so on) are supported.

Now check the log at SC33 to determine whether you have IKED and TRMD messages relating to the Permission denied message and then the successful **ping** response.

From the OMVS shell, move to /tmp/syslog and browse syslog.log to view messages like those depicted in Example 4-24.

Example 4-24 IKEDD and TRMD messages for SC33

```
IKEDD a  IKE: IKE CERTINFO : Matched certificate with label IKE Daemon on SC33

IKEDD b  IKE: Message instance 1: EZD1047I IKE phase 1 security association 1 created from
192.168.1.40 to 10.1.1.50 Mode : Main HowToAuthPeers : RsaSignature HowToEncrypt : DES
HowToAuthMsgs : SHA1 DHGroup : Group1 Lifetime : 28800 Lifesize : NONE Remote IKE Port: 500

IKEDD c  IKE: Message instance 2: EZD1016I Phase 2 security association 2 created - LocalIp
: 192.168.1.40 RemoteIp: 10.1.1.50 LocalDataPort : ALL RemoteDataPort : ALL Protocol :
ALL(0) HowToEncap : TRANSPORT HowToEncrypt : DES_CBC_8 HowTwToAuth : HMAC_SHA ( ESP )
RefreshLifetime : 14400 RefreshLifesize : NONE VpnLiife : 86400 PFS : NONE

TRMDE331 d   TRMD.TCPIPE[262378]: EZD0818I Tunnel added: 03/03/2008 22:56:06.30 vpnaction=
IPSec__Silver tunnelID= Y2  AHSPI= 0  ESPSPI= 1312697222

TRMDE331 e   TRMD.TCPIPE[262378]: EZD0814I Packet permitted: 03/03/2008 22:56:09.72 filter
rule= All_Traffic_Silver~7 ext= 2 sipaddr= 10.1.1.50 dipaddr= 192.168.1.40 proto= icmp(1)
type= 8 code= 0 -= Interface= 192.168.2.41 (I) secclass= 255 dest= local len= 284
vpnaction= IPSec__Silver tunnelID= Y2 ifcname= OSA2080L
```

```
TRMDE331 f   TRMD.TCPIPE[262378]: EZD0814I Packet permitted: 03/03/2008 22:56:09.72 filter
rule= All_Traffic_Silver~7 ext= 1 sipaddr= 192.168.1.40 dipaddr= 10.1.1.50 proto= icmp(1)
type= 0 code= 0 -= Interface= 192.168.2.41 (0) secclass= 255 dest= local len= 284
vpnaction= IPSec__Silver tunnelID= Y2 ifcname= OSA2080L
```

Notice, in our case of using OnDemand tunnel activation, there are no `permission denied` messages in the log. This is due to the fact that IPSec did not block the request; as we explained earlier, the first **`ping`** request was unsuccessful because a tunnel was not yet active.

4.2.6 Diagnosing IKE with RSA signature mode

The two most relevant publications for diagnosing problems in an IKE implementation are:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781

Review the information in these two publications, because they describe tests for verifying both manual IPSec environments and dynamic IPSec environments. For example, *z/OS Communications Server: IP Configuration Guide*, SC31-8775 contains the following sections:

- Overview of diagnosing IP security problems
- Steps for diagnosing IP security problems
- Steps for verifying IP security operation
- Tools for diagnosing IP security problems

z/OS Communications Server: IP System Administrator's Commands, SC31-8781, contains the syntax for the **`ipsec`** command. If you are in the UNIX environment and do not recall the syntax, simply issue the following from the OMVS shell environment to see the syntax options:

```
ipsec -?
```

You will also be making use of the **`netstat`** command variations, and of **`pasearch`** to diagnose problems. Be sure to understand how to issue the **`racdcert list`** commands as well. Here is the list of the other relevant commands:

```
netstat
pasearch
racdcert
```

You will find that the syslog daemon processes are extremely important in diagnosing problems you may run into. The most important messages for diagnosing problems are:

- IKE daemon messages
- TRMD messages
- PAGENT messages

Your syslog daemon implementation may place the pertinent messages in separate logs or in one large log. However, many of the messages you may need for advanced diagnosis are only available if you raise the logging levels.

Note: Allow your initial logging levels to default when you configure the files with IBM Configuration Assistant. After you upload the various files to the mainframe, and if you discover that NSS is not working as expected, then you may manually edit the PAGENT, IKED, and NSSD configuration files and any of the policy files to raise the logging levels temporarily.

Reset the logging levels to a lower value after you have diagnosed and remedied the problems.

4.3 Configuring Network Security Services (NSS)

The network topology that we use to configure our NSS scenario is based upon the RSA signature mode scenario that we built in 4.2, “Configuring IKE in RSA signature mode” on page 96. In Figure 4-30 you see again the network diagram that illustrates a successful IPSec and IKE implementation.

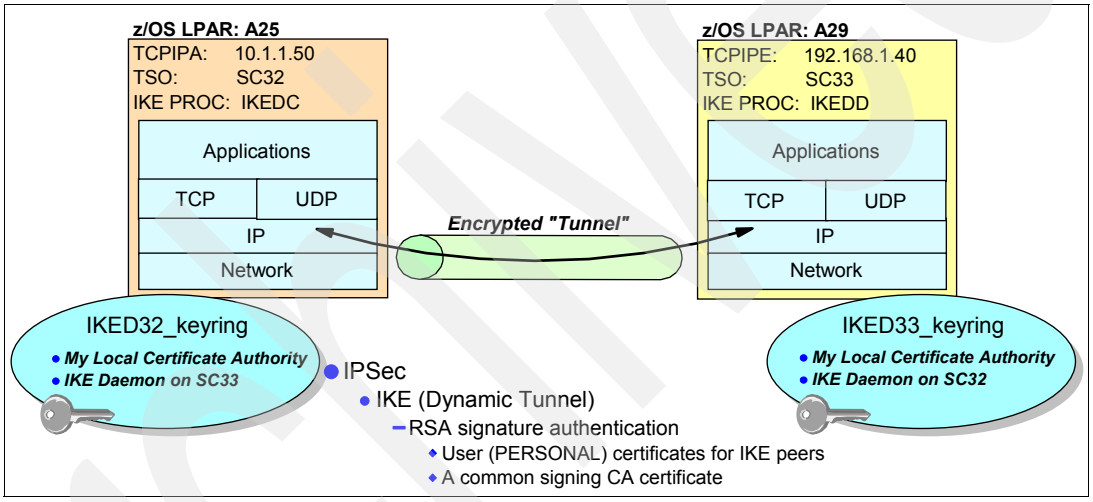


Figure 4-30 VPN tunnel between TCPIPA on SC32 and TCPIPE on SC33

Using IKE with RSA signature mode and with local key rings holding the x.509 certificates that provide authentication, a dynamic VPN tunnel was built between TCPIPA on SC32 and TCPIPE on SC33.

With this as our starting point, we show you how to provide the same encrypted tunnel between SC32’s TCPIPA and SC33’s TCPIPE. However, in this scenario, we use the certificate management services of Network Security Services on behalf of SC32’s TCPIPA and the local certificate services of SC33’s TCPIPE.

Note: With the topology of our test environment, we took some liberties with the rationale for building an NSS. Normally the IKE client in an NSS client server relationship would reside in a Less Secure Zone and take advantage of the services of an NSS server residing in a Secure Zone.

In our scenario, however, we continue to use IKE peers that reside in the same sysplex. In other words, both IKE peers reside in what truly is a Secure Zone. The NSS function can be implemented in such an environment, but its real purpose is to provide secure storage for certificates and identities for IKE peers residing outside the Secure Zone.

4.3.1 Overview of preliminary tasks

1. Select or build an IPSec environment that is using IKE in RSA signature mode so that you can migrate one of the IKE peers to be an NSS client. We have built this environment on 4.2, “Configuring IKE in RSA signature mode” on page 96.
   a. Identify the key rings and certificates that are to be used by the NSS server and by the NSS client.
   b. The assumption is that you have a PAGENT procedure running at the IKE peers for the IPSec filters and policies. See the relevant chapters in this series of publications for information about PAGENT and IPSec.
2. Identify or build the TLS certificate and key ring that the NSS server is to utilize.
   a. The assumption is that you have a PAGENT procedure running at the NSS client and server sites for the TLS policies. See the relevant chapters in this series of publications for information on PAGENT and AT-TLS.
3. SYSLOGD: Ensure that you have SYSLOGD defined at the NSS client and the NSS server nodes so that problem determination is simplified.

   IKED logging

   Verify that a log for IKED exists both at the NSS client node and at the VPN partner node. We added the following lines to the appropriate /etc/syslogd.conf file:

   ```
   *.IKED*.*.* /tmp/iked-sc32.log (at NSS client side)
   *.IKED*.*.* /tmp/iked-sc33.log (at VPN partner side)
   ```

   Note that IKED logging can be accomplished with another variation of the definition:

   ```
   local4.*     /tmp/iked-sc32.log
   ```

   NSSD logging

   At the NSS server node, add a statement like the following to the /etc/syslogd.conf file to capture NSSD messages:

   ```
   *.NSSD*.*.* /tmp/nssd.log
   ```

   However, many NSSD messages are also captured in the local4.* facility name and priority. Therefore, at the server add:

   ```
   local4.*     /tmp/nssd-sc33.log
   ```

   TRMD logging

   At all the IKED nodes on the mainframe, ensure that the TRMD messages flow to a log file. They may flow to the IKED log or elsewhere, as you choose. For example, you might implement the following:

   ```
   *.TRMD*.*.* /tmp/iked-sc32.log (at IKEDC client side)
   *.TRMD*.*.* /tmp/iked-sc33.log (at VPN partner side)
   ```
4. Create a logical network diagram that depicts the MVS System IDs, TCP/IP stack names, IPSec and IKE peers, and IP addresses that play a role in your NSS implementation. Refer to Figure 4-31 on page 130 for an example.
5. Complete the worksheet with the requisite NSS implementation information so that you are prepared to begin the definition process. See the sample worksheet in “Worksheet for NSSD implementation” on page 214. Also consult our worksheet, which has already been filled out for our implementation.

4.3.2 NSS client and NSS server

With this scenario, an NSS client (IKEDC) on SC32 uses the services of NSSD on SC33. The NSS client retrieves certificates and keys from the NSS server in order to establish an IPSec tunnel between TCPIPA on SC32 and TCPIPE on SC33. Refer to Figure 4-31.

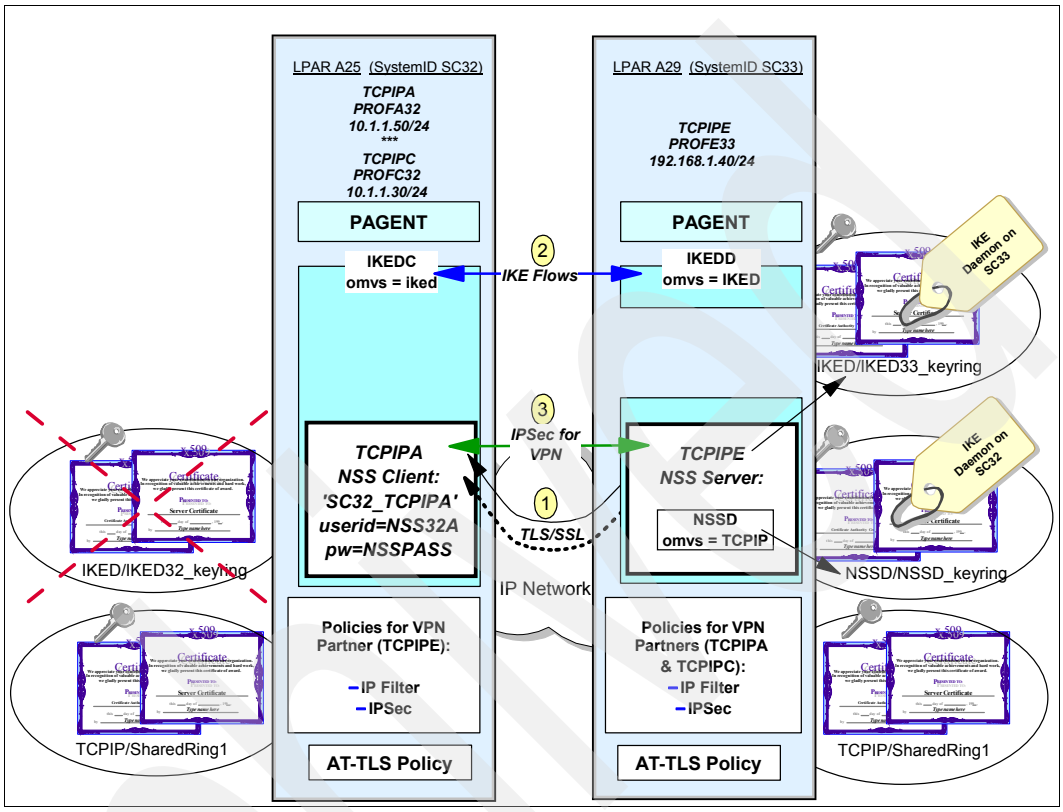


Figure 4-31 TCPIPA is NSS-enabled

TCPIPA on SC32 will build a VPN tunnel with TCPIPE on SC33. In so doing, its IKE procedure is to act as an NSS client for RSA signature verification. The necessary certificates and private keys are stored on the NSS server (TCPIPE) on SC33.

- At (1) in the figure, you see the TLS/SSL connections between NSS client and server. The NSS client and the NSS server are using a shared key ring for their TLS certificates. The NSS server accesses the NSSD/NSSD_keyring for the certificates that will provide RSA signature verification.
- At (2), you see the flows between IKEDC and IKEDD to manage the IPSec session keys. IKEDD natively accesses a key ring named IKED/IKED33_keyring which contains IKE certificates and private keys. The NSS server accesses its key ring, NSSD/NSSD_keyring, when it provides NSS Certificate Management services to IKEDC on behalf of TCPIPA on SC32. IKEDC no longer requires key ring IKED/IKED32_keyring in this scenario because it is using the key ring services of the NSS server.
- At (3), you see the IPSec tunnel that is built between the two VPN partners: TCPIPA and TCPIPE.

As seen, TCPIPA on SC32 in Figure 4-31 is now NSS-enabled. Two new types of identities are assigned to an NSS-enabled stack:

- The symbolic client name, configured by the client in his **`iked.conf`** file
- The client user ID, defined as a RACF user ID at the NSS server node

Therefore, in Figure 4-31 on page 130, you see:

- A symbolic client name of `SC32_TCPIPA`
- A client user ID of `NSS32A`
- A client password of `NSSPASS`

You also see:

- An NSSD server on SC33; this NSSD server has affinity to the TCP/IP stack named TCPIPE.

Next, note that the diagram depicts:

- An IKED key ring available to the NSS server on SC33 (and not required by the NSS client, IKEDC on SC32)
- A shared TLS/SSL key ring available both to SC32 and to SC33

Finally, notice that:

- The IKE certificate for the NSS client, which resides on the NSSD_keyring, has a label of IKE Daemon on SC32.
- The IKE certificate for the IKEDD peer, which resides on the IKED33_keyring, has a label of IKE Daemon on SC33.

This information is critical to the successful implementation of NSSD. This is why we made the recommendation that you already have experience in IPSec, IKE, and AT-TLS prior to working with NSS so that you can focus entirely on NSS itself.

4.3.3 Preparing for configuration

To further simplify the implementation of NSS, we provide a worksheet that can be filled in with all the information required for a successful NSS implementation. Example 4-25 shows a sample worksheet that we filled in for our implementation.

A blank worksheet for you to fill in is provided 4.6, “Worksheet for NSSD implementation” on page 214.

Example 4-25 Worksheet reflecting NSS implementation

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon? Our response for the scenario illustrated in Figure on page 130 is:
   - SC33
2. What is the superuser user ID associated with the OMVS segment under which the NSS Daemon is to run? Our response is:
   - TCPIP
3. With which TCP/IP stack (or stacks) does the NSS daemon establish affinity? Our response is:
   - TCPIPE
4. What is the address or the DNS name that the NSS client will use to connect to the NSS server? Our response is:
   - 192.168.1.40
5. What is the MVS System ID under which each NSS client is to run? What are the TCP/IP stack names that are to use the IKE services of the NSS client? Our responses for our scenario are:

- SC32
  - TCPIPA

6. What “symbolic client name” name do you want to assign to each TCP/IP stack that is to request authentication of the NSS server? In our scenario, the answer is:
   - SC32_TCPIPA
7. What “client ID” do you want to assign to each TCP/IP stack that is to request authentication of the NSS server? The response is:
   - NSS32A
8. What type of authentication should our client user ID present: a password or a passticket? The response is:
   - password of NSSPASS
9. What is the user ID associated with the OMVS Segment of the client’s IKEDC procedure? Our answer is:
   - IKED
10. What is the name of the keyring that NSS should use to manage all of the clients’ private keys and certificates?
    - NSSD/NSSD_keyring
11. What is the certificate label assigned to each IPSec RSA certificate for each TCP/IP stack? Display with **`racdcert listring(IKED_keyring) id(IKED)`**. Our response is:
    - For client TCPIPA: IKE Daemon on SC32 ID(IKED) PERSONAL NO
12. With which user ID must the NSS Client certificate be associated when it is moved to the NSSD key ring? (That is, what is the name of the user ID under which NSSD will run?) Our response is:
    - NSSD
13. What is the name of the key ring on the NSS node that is used for the TLS/SSL secure connection between the NSS server and client? Our response is:
    - TCPIP/SharedRing1
14. What is the name of the keyring on the NSS client node that is used for the TLS/SSL secure connection between the NSS client and server? Our response is:
    - TCPIP/SharedRing1
15. What are the IP addresses of the NSS server and the NSS client for which you may have to establish IP filters on the NSS client node? Our responses are:
    - Client: 10.1.1.50
    - Server: 192.168.1.40
16. If the NSS server has implemented IPSec, what are the IP addresses of the NSS server and the NSS client for which you may have to establish IP filters on the NSS server node? Our responses are:
    - Client: 10.1.1.50
    - Server: 192.168.1.40
17. What are the TSO user IDs of the IPSec administrators who are authorized to manage the VPN? Our response is:
    - CS02, CS03, CS04, CS05, CS06

Role of the IBM Configuration Assistant

The IBM Configuration Assistant program allows you to configure your NSS server (and clients) through a graphical user interface; it then generates the NSSD configuration file as well as the necessary AT-TLS policy and IP filter rules for NSS client-server traffic. Note that the Configuration Assistant also generates the NSS client configuration in the IKED configuration file.

Note: The NSS server itself is not required to have IKED running unless it is also to be the endpoint of an IPSec tunnel. In our scenario, we will use the server node as an IPSec endpoint.

4.3.4 Configuring the NSS environment

As you can see, many tasks should already have been completed prior to defining the NSS server and client environment. You see many of the prerequisite components in Figure 4-32.

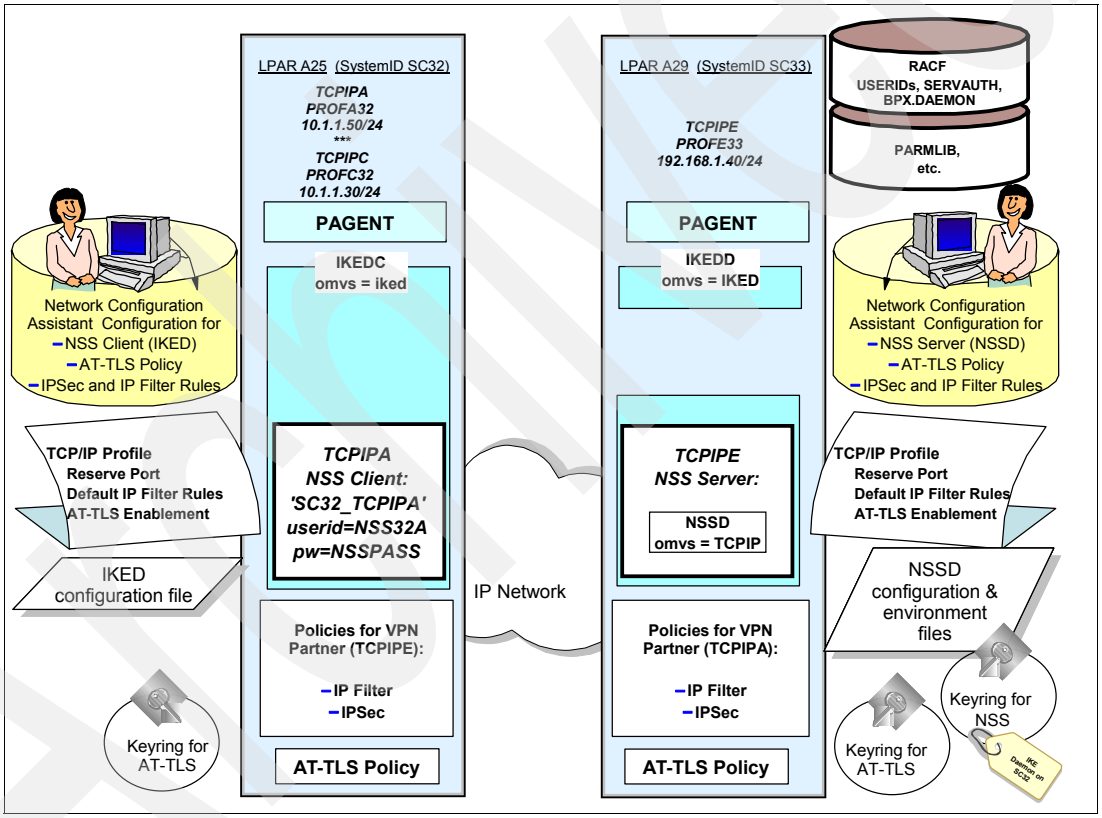


Figure 4-32 Components of an NSS configuration for our scenario

The NSS server or daemon (NSSD) prerequisites

The base environment for NSS has several prerequisites at the server side:

1. A TCP/IP stack that is enabled for AT-TLS.
   - Reserve the NSS server port (the default is 4159).
   - The stack's default IP filtering policy can be updated to allow NSS client/server traffic.
   - Any existing PAGENT IPSec filtering policy must also be updated to permit NSS client/server traffic.
   - AT-TLS enablement in the NSS server IP stack.

2. An existing Policy Agent managing AT-TLS policies.
   - AT-TLS policy to protect the TCP connections between NSS clients and the server.
   - IP traffic descriptors to allow NSS traffic between client systems and the server system.
3. A Network Security Services Daemon (NSSD) with an NSS daemon configuration file that has been generated either manually or with the Network Configuration Assistant.
4. A small set of environment variables that define the location configuration resources.
5. A RACF key ring with certificates for the secure (AT-TLS) communications between NSS server and client.
   - Optionally, a Certificate Authority Certificate that can be used to authenticate the client if the server requests client authentication during the AT-TLS handshake phase.
6. A RACF key ring that contains all of the certificates and private keys that will be used for RSA signature creation and signature verification on behalf of any of the NSS clients during IKE processing.
   - Required: a Certificate Authority Certificate for IPSec and IKE that can be used to authenticate the VPN partners of the NSS client.

   Note: You may use a single key ring for the certificates required for both AT-TLS and IKE. Alternatively, you may use more than one key ring. Consult Chapter 3, “Certificate management in z/OS” on page 31 for more information on this subject. Also see “Key rings for the NSS implementation” on page 92 in this chapter for a discussion of several strategies that can be used to manage key rings.

7. RACF SERVAUTH profiles to control client access to NSS services, including both Remote Management and Certificate services.
8. A RACF environment that authorizes the user ID of the NSS client:
   - Either one or more RACF user IDs with passwords that can be presented for authorization to the server when the client or clients connect.
   - A PassTicket profile for authorization.

The NSS client prerequisites

The base environment at the client side includes the following prerequisites:

1. A TCP/IP stack that is enabled for AT-TLS and IPSec.
   - The stack’s default IP filtering policy can be updated to allow NSS client/server traffic.
   - Any existing PAGENT IPSec filtering policy must also be updated to permit NSS client/server traffic.
   - AT-TLS enablement in the NSS server IP stack.
2. An existing Policy Agent managing AT-TLS policies.
   - An AT-TLS policy to protect the TCP connections between NSS clients and the server.
   - IP traffic descriptors to allow NSS traffic between client systems and the server system.
3. An existing Policy Agent with IPSec filters and policies for the communication between the clients and the IPSec or VPN partners.

   These policies can be managed locally at the client sites, or they can be distributed to the client sites by a Central Policy Server. For information about how to build a set of IPSec filters and policies and, optionally, how to distribute them from a Central Policy Server, see the following chapters:
   - Chapter 8, “IP filtering” on page 319

- Chapter 9, “IP Security” on page 373
- Chapter 6, “Central Policy Server” on page 257

4. An IKE daemon (IKED) with an IKE configuration file specifying the use of an NSSD. The file can be generated either manually or with the Network Configuration Assistant.
   - Note: The RSA certificate for the client’s participation in the Internet Key Exchange Protocol does not reside at the client site with NSS; it resides on the NSS server’s key ring.
5. A small set of environment variables that define the location configuration resources.

4.3.5 Configuring prerequisites for Network Security Services

As already noted, the easiest way to build an NSS environment is to have the prerequisite technologies like TLS/SSL and IPSec already functioning. This means that working certificates are in place and that IPSec is already successfully implemented. With these technologies firmly in place, it is then easy to set up the NSSD environment.

Recommendation: Implement NSSD after you are already comfortable with the concepts of certificates, AT-TLS, IPSec, and the use of the Network Configuration Assistant. Even better, implement NSSD after you have already worked in the se environments. In this way you minimize risk and reduce the learning curve, because you need only implement the NSS daemon and its client.

In our scenario, we assume you are already an expert in creating key rings and certificates, and that you are already familiar with the Network Configuration Assistant. We refer you to previous chapters for review for many of the steps in the following sections.

Certificate creation and management

You must have the appropriate RACF keyring environment with the necessary Certificate Authority Certificates and server certificates available in order to implement AT-TLS and NSS. These key rings should be ready ahead of time. For information about creating the various types of certificates, consult:

- Chapter 3, “Certificate management in z/OS” on page 31.

For working examples of how these certificates were used for AT-TLS with TN3270 and FTP, consult:

- Chapter 11, “Application Transparent Transport Layer Security” on page 443
- Chapter 14, “Telnet security” on page 539
- Chapter 15, “Secure File Transfer Protocol” on page 579

For working examples of how these certificates and keyrings were used for IPSec and with IKED, consult:

- Chapter 9, “IP Security” on page 373.

Note: In the examples provided, we have assumed that all the necessary RACF certificate classes have been defined and activated. If this is not the case, consult the previously mentioned chapters in this book, and consult *z/OS Communications Server: IP Configuration Guide*, SC31-8775. For specific details about RACF Security Server functions and command syntax, consult:

- *z/OS Security Server RACF Security Administrator’s Guide*, SA22-7683
- *z/OS Security Server RACF Command Language Reference*, SA22-7687

Build the TLS/SSL server certificate for the NSS server

In this section we take a look at the certificates that we have already created for a shared Site ring and shared server certificates.

Certificate definition for NSS and AT-TLS

First, look at the CA certificate definition in Example 4-26. This is the self-signed CA certificate with which we sign our SITE certificate for the NSS daemon.

Example 4-26 Certificate Authority certificate

```
//CERTAUTH  JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     Create Certificate Authority Certificate for ITSO           *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert certauth gencert                          -
    subjectsdn( o('IBM Corporation')                 -
                ou('ITSO Certificate Authority')     -
                 C('US'))                            -
    NOTBEFORE(DATE(2007-09-11))                      -
    NOTAFTER(DATE(2008-09-11))                       -
    keyusage(certsign)                               -
    withlabel('CS19 ITSO CA1')
  setropts raclist(DIGTCERT) refresh
  racdcert certauth list(label('CS19 ITSO CA1'))
/*
```

Next, examine the shared SITE certificate that we have used for several servers, like FTP and TN3270, on behalf of SSL/TLS. We have decided to use the same SITE certificate to represent our NSS server during the SSL/TLS flows. See Example 4-27 for the JCL we used to create the NSS server certificate.

Example 4-27 SITE certificate for TLS/SSL communication between NSS server and client

```
//CERTSITE  JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTSITE EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//*     CREATE SITE AUTHORITY CERTIFICATE FOR ALL SERVERS (SHARED)
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert site gencert subjectsdn(cn('ITSO.IBM.COM')   -
     o('IBM Corporation')                              -
     ou('ITSO CS19 Shared SITE')                       -
     C('US'))                                          -
     NOTBEFORE(DATE(2007-09-11))                        -
     NOTAFTER(DATE(2008-09-11))                         -
     withlabel('CS19 ITSO SharedSite1')                -
     signwith(certauth label('CS19 ITSO CA1')
     racdcert site list(label('CS19 ITSO SharedSite1'))
/*
```

Example 4-28 on page 137 shows the JCL we used to create the key ring for AT-TLS and to attach the certificates to that key ring.

Example 4-28 Create key ring and connect certificates for NSS

```
//KEYRINGS  JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRINGS EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*******************************************************************
//* Add a new keyring to the various clients' RACF ID , then ...      *
//* Add the SITE certifiate to the servers' keyring.  The             *
//* keyring name in the server configuraton must be changed to        *
//* point to the new ring name as in "KEYRING SAF <userid>/SharedRing *
//* This job assumes that keyrings are associated with userid 'TCPIP' *
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert ID(TCPIP) ADDRING(SharedRing1)
  racdcert ID(TCPIP) CONNECT(CERTAUTH                  -
                     LABEL('CS19 ITSO CA1')            -
                     RING(SharedRing1)                 -
                     USAGE(CERTAUTH)
  racdcert ID(TCPIP) CONNECT(SITE                      -
                     LABEL('CS19 ITSO SharedSite1')    -
                     RING(SharedRing1)                 -
                     DEFAULT                           -
                     USAGE(PERSONAL)
   setropts raclist(DIGTRING) refresh
   setropts raclist(DIGTCERT) refresh
   racdcert listring(SharedRing1) id(TCPIP)
 /*
```

TCP/IP stack enablement for AT-TLS

Consult the following chapter for information about enabling AT-TLS in the TCP/IP profile for both the NSS client (IKEDC) on SC32 and the NSS server (NSSD) on SC33:

- Chapter 11, “Application Transparent Transport Layer Security” on page 443

We perform this step later in “Update the TCP/IP profile” on page 150.

Build the AT-TLS policy for NSS server and client

We use the Network Configuration Assistant in a later step (“Update the policy files at the NSS server and client nodes” on page 151) to build the AT-TLS policy for NSS server and client.

Build the IKED certificates for the NSS client

IKED certificate definition for NSS client

We already built the IKED certificates for the NSS client when we created the IKE scenario with RSA signature authentication in 4.2.1, “Generating certificates for IKE RSA signature mode” on page 96.

However, those certificates reside on a key ring named IKED/IKED32_keyring. This is a key ring we no longer need for TCPIPA, because TCPIPA on SC32 is to become an NSS client. The certificates that resided on IKED32_keyring now need to reside on the NSSD_keyring. Therefore, we need to create the key ring for the NSS server and connect the required certificates to NSSD’s key ring.

Avoid this mistake: It sounds easy enough to create a key ring for the NSS server and then connect the NSS client's certificate to it. However, there is a critical set of steps that must be executed to ensure that the NSS server can properly process the NSS client certificates. In fact, the user (PERSONAL) certificates on the NSSD keyring must be owned by the NSSD server—they *cannot* be associated with the old IKED owner.

You cannot simply reconnect the existing certificate to the new ring. Instead, you must execute these steps to move certificates from the old IKE key ring to the new NSS key ring:

1. Export the existing certificate to a file in PKCS12DER format. This format includes the private key in the operation.
2. Delete the existing certificate from the RACF database, thus deleting its association with the old owner IKED.
3. Import the certificate under the user ID of NSSD to the RACF database.
4. Add the certificate to the NSSD key ring.

Note that CERTAUTH or SITE certificates do not have individual owners, and can simply be reconnected to the NSSD key ring.

1. Review the JCL in Example 4-29, where you see how we export the NSS client's certificate to a dataset. Remember to name a password for this operation; it is used again later when we import the certificate.

Example 4-29 Export a certificate to a dataset

```
//EXPORT32  JOB MSGCLASS=X,NOTIFY=&SYSUID
//EXPORT32 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     To migrate an individual PERS certificate onto NSSD Keyring   *
//*     Export the individual PERSONAL certificate with its key       *
//*     from the RACF database in base-64 encoded format.  Then import*
//*     the certificate with a userid of NSSD; and attach to the      *
//*     NSSD_keyring.                                                 *
//*     Related Jobs:  EXPORT32, CERTDE32, IMPORT32, KEYRAD32         *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT ID(IKED) EXPORT(LABEL('IKE Daemon on SC32')) -
     DSN('CSO2.CERT.EXPORT32') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
/*
```

2. Examine the JCL we used to delete the certificate from the RACF database; see Example 4-30.

Example 4-30 Delete the certificate under its previous owner from the RACF database

```
//CERTDE32  JOB MSGCLASS=X,NOTIFY=CSO2
//CERTDE32 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Related jobs:  EXPORT32, CERTDE32, IMPORT32, KEYRAD32         *
//*     Delete a certificate in order to import it again with         *
//*        a different USERID.                                        *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert ID(IKED) delete                          -
    (label('IKE Daemon on SC32'))
```

```
  setropts raclist(facility) refresh
  racdcert ID(IKED) list(Label('IKE Daemon on SC32'))
```

3. Import the certificate with its private key back into the RACF database, but associate it with the user ID of NSSD; see Example 4-31.

Example 4-31 Import the certificate and its key; associate with new user ID

```
//IMPORT32  JOB MSGCLASS=X,NOTIFY=CSO2
//IMPORT32 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     To migrate an individual PERS certificate onto NSSD Keyring   *
//*     Export the individual PERSONAL certificate with its key       *
//*     from the RACF database in base-64 encoded format.  Then import*
//*     the certificate with a userid of NSSD; and attach to the      *
//*     NSSD_keyring.                                                 *
//*     Related Jobs:  EXPORT32, CERTDE32, IMPORT32, KEYRAD32         *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT ID(NSSD) ADD('CSO2.CERT.EXPORT32')        -
     WITHLABEL('IKE Daemon on SC32') TRUST PASSWORD('security')
  setropts raclist(facility)    refresh
/*
```

4. Create the NSSD key ring and add the NSS Client certificate and private key to the NSSD key ring, as shown in Example 4-32.

Example 4-32 Create new key ring for NSSD and add the NSS Client certificate(s) to it

```
//KEYRAD32  JOB MSGCLASS=X,NOTIFY=CSO2
//KEYRAD32 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     To migrate an individual PERS certificate onto NSSD Keyring   *
//*     Export the individual PERSONAL certificate with its key       *
//*     from the RACF database in base-64 encoded format.  Then import*
//*     the certificate with a userid of NSSD; and attach to the      *
//*     NSSD_keyring.                                                 *
//*     Related Jobs:  EXPORT32, CERTDE32, IMPORT32, KEYRAD32         *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
racdcert ID(NSSD) ADDRING(NSSD_keyring)
racdcert ID(NSSD) CONNECT(                          -
                  LABEL('IKE Daemon on SC32')       -
                  RING(NSSD_keyring)                -
                  USAGE(PERSONAL)
  setropts raclist(DIGTRING) refresh
  setropts raclist(DIGTCERT) refresh
  racdcert listring(NSSD_keyring) id(NSSD)
/*
```

5. Finally, you must add the CA Certificates that have signed the certificates of the NSS Client's VPN partners to the NSSD key ring; see Example 4-33.

Example 4-33 Adding the CA certificates to the NSSD key ring

```
//KEYNSSD2  JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRING2 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Create NSSD keyring and associate with userid of NSSD         *
```

```
//*     Add NSS(IKE) individual or shared SITE     *
//*                  Certificates for NSS Clients                    *
//*     Add NSS(IKE) CA Certificates of remote IKE peers             *
//* racdcert ID(NSSD) addring(NSSD_keyring)
//*******************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert ID(NSSD) CONNECT(CERTAUTH                  -
                    LABEL('My Local Certificate Authority')       -
                    RING(NSSD_keyring)                 -
                    USAGE(CERTAUTH))
  setropts raclist(DIGTRING) refresh
  setropts raclist(DIGTCERT) refresh
  racdcert listring(*) id(nssd)
/*
```

TCP/IP stack enablement for IP Security (IPSec)

Ensure that the NSS client stacks are already implemented with IPSec and IKED. All the VPN peers to the NSS client stacks must also be enabled for IPSec. Consult Chapter 9, “IP Security” on page 373 for more information about these topics.

4.3.6 Configuring authorizations for NSS

We used RACF as our external security manager for Network Security Services. This section explains the steps to authorize resources to NSS.

1. Create a user ID for the NSSD started task and associate it with a required UID of 0. Example 4-34 shows the RACF commands we executed from TSO to create this user ID. *z/OS Communications Server: IP Configuration Guide*, SC31-8775, suggests a user ID of NSSD.

   Example 4-34 RACF commands to add a user ID for the NSS daemon

   ```
   ADDUSER TCPIP DFLTGRP(TCPGRP) OMVS(UID(0) HOME('/'))
   RDEFINE STARTED NSSD.* STDATA(USER(NSSD))
   SETROPTS RACLIST(STARTED) REFRESH
   ```

2. We permitted the NSSD user ID to SYS1.PARMLIB.

   ```
   PERMIT SYS1.PARMLIB ID(NSSD) ACCESS(READ)
   ```

3. We defined the key ring controls for the NSS client certificates.

   ```
   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.GENREQ uacc(NONE)
   RDEFINE FACILITY IRR.DIGTCERT.GENCERT uacc(NONE)
   ```

4. We permitted both the NSSD user ID and the user ID that will administer key ring access to the facility classes. The user CS02 requires this access.

   ```
   PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(NSSD,CS02) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(NSSD,CS02) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(NSSD,CS02) ACC(UPDATE)
   ```

5. We permitted the NSSD user ID of TCPIP to the BPX.DAEMON facility class.

   ```
   PERMIT BPX.DAEMON CLASS(FACILITY) ID(NSSD) ACCESS(READ)
   setropts raclist(facility) refresh
   ```

6. In your case, if you have chosen to implement secured signon with PassTickets, follow the instructions for doing so in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. We elected to use a simple password of NSSPASS for the NSS function and skipped this step.
7. The next implementation step requires that you authorize SERVAUTH access to the NSS client and to the user who will execute remote IPSec management with Network Security Services.
   a. Create a user ID for the NSS client. Recall that in our diagram we decided to name our client NSS32A. The client's password is to be NSSPASS. Example 4-35 shows the JCL we used to create this user ID.

Example 4-35 NSS Client user ID and password generation

```
//ADDNSSU  JOB (999,POK),'ADD A USER',CLASS=A,REGION=0M,
//     MSGCLASS=T,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM SYSAFF=*
//*
//* Use this job to create a user ID that does not use TSO and
//* require no particular RACF authority.
//*
//IKJEFT   EXEC PGM=IKJEFT01
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   ADDUSER NSS32A PASSWORD(NEW2DAY) +
    NAME('ID for Comm Server') +
    OWNER(HAIMO) UACC(READ) DFLTGRP(SYS1) +
    AUTHORITY(JOIN) GRPACC +
    OMVS(AUTOUID HOME(/u/nss32a) PROGRAM(/bin/sh))
  CONNECT   NSS32A GROUP(SYS1) OWNER(HAIMO) AUTHORITY(JOIN) +
            UACC(ALTER)
  ALU NSS32A PASSWORD(NSSPASS) NOEXPIRED
  PASSWORD USER(NSS32A) NOINTERVAL
  ADDSD     'NSS32A.**' UACC(NONE) OWNER(NSS32A)
  SETROPTS REFRESH RACLIST(TSOPROC ACCTNUM TSOAUTH)
  DEFINE ALIAS   (NAME('NSS32A') RELATE('UCAT.COMCAT'))
/*
//DEFINEZ  EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(NSS32A.HFS) -
            LINEAR CYLINDERS(1 1) SHAREOPTIONS(3)    -
            VOLUMES(COMST4))
/*
//CREATEZ  EXEC   PGM=IOEAGFMT,
//           PARM=('-aggregate NSS32A.HFS -compat
//             -owner NSS32A -group SYS1 -perms o755')
//SYSPRINT DD     SYSOUT=*
//STDOUT   DD     SYSOUT=*
//STDERR   DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//CEEDUMP  DD     SYSOUT=*
//*
//COPYPRO EXEC PGM=IKJEFT01
//PDS1   DD DISP=SHR,
//       DSN=WTSCPLX5.USS.PROFILE(NEWPROF)
```

```
//HFS1   DD PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//       PATH='/u/nss32a/.profile'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  PROF MSGID WTPMSG
  OCOPY INDD(PDS1) OUTDD(HFS1) TEXT
/*
//CHOWN1     EXEC PGM=BPXBATCH,
//   PARM='SH chown nss32a /u/nss32a/.profile'
//STDOUT     DD  PATH='/tmp/stdout',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//STDERR     DD  PATH='/tmp/stderr',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//*
```

b. Then we permitted this user ID to a new class that we created with the commands in Example 4-36. This class is used for the NSS Certificate Services.

Example 4-36 Authorizing user IDs to new SERVAUTH class for NSS

```
RDEFINE SERVAUTH EZB.NSS.SC33.SC32_TCPIPA.IPSEC.CERT UACC(NONE)

PERMIT EZB.NSS.SC33.SC32_TCPIPA.IPSEC.CERT CLASS(SERVAUTH) ID(NSS32A) ACCESS(READ)

SETROPTS RACLIST(SERVAUTH) REFRESH

SETROPTS GENERIC(SERVAUTH) REFRESH
```

Note the syntax of this class:

```
EZB.NSS.<NSSD MVS System ID>.<NSS Client Symbolic Name>.CERT
```

Now you understand why we recommended that you carefully design and diagram this scenario and assign meaningful naming conventions to it.

You will need a class like this for each client. Alternatively, you can wildcards, but you lose some granularity and control when you do so.

c. NSS has a second capability beyond Certificate Services: IPSec management. Therefore, we next permitted the NSS client user ID and the user ID of the NSS administrator to a new class that we created with the commands in Example 4-37. This class is used for the NSS Network Management Services.

Example 4-37 Authorizing user IDs to new SERVAUTH facility class for NSS

```
RDEFINE SERVAUTH EZB.NSS.SC33.SC32_TCPIPA.IPSEC.NETMGMT UACC(NONE)

PERMIT EZB.NSS.SC33.SC32_TCPIPA.IPSEC.NETMGMT CLASS(SERVAUTH) ID(NSS32A)
ACCESS(READ)

SETROPTS RACLIST(SERVAUTH) REFRESH

SETROPTS GENERIC(SERVAUTH) REFRESH
```

Note that the syntax of this second class is similar to the one you saw for Certificate Services. However, there is an extra IPSEC segment. Note the syntax of this SERVAUTH class:

```
EZB.NSS.<NSSD MVS System ID>.<NSS Client Symbolic Name>.IPSEC.NETMGMT
```

d. We must define an unusual SERVAUTH class resource profile for each NSS client. One of the fields in it is comprised of the *label name* of the NSS client's IKE certificate. This resource grants access by the client to its own IKE certificate, even though that IKE certificate resides at the NSS server site.

Needing to verify the label on that certificate, we first used a RACF command to list the contents of the IKE key ring containing the certificate assigned to our TCPIPA NSS client; see Figure 4-38.

Example 4-38 RACDCERT LISTRING(NSSD_keyring) ID(NSSD)

```
Digital ring information for user NSSD:

  Ring:
       >NSSD_keyring<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  ------------------------------     ------------   --------   -------
  IKE Daemon on SC32                  ID(NSSD)       PERSONAL     NO
  My Local Certificate Authority      CERTAUTH       CERTAUTH     NO
```

With this display we verify that the label assigned to the client certificate is *IKE Daemon on SC32*.

Now we are ready to define the new class resource with this format:

```
EZB.NSSCERT.<NSSD MVS System ID>.<NSS Client IKE Certificate Label>.HOST
```

or

```
EZB.NSSCERT.<NSSD MVS System ID>.<mappedlabelname>.HOST
```

You see our definition in Example 4-39.

Example 4-39 SERVAUTH Resource profile for NSS client's Certificate Authority certificate

```
RDEFINE SERVAUTH EZB.NSSCERT.SC33.IKE$DAEMON$ON$SC32.HOST UACC(NONE)
PERMIT EZB.NSSCERT.SC33.IKE$DAEMON$ON$SC32.HOST CLASS(SERVAUTH) ID(NSS32A)
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS GENERIC(SERVAUTH) REFRESH
```

Notice the unusual syntax of the certificate label field in the SERVAUTH and PERMIT definitions. Because our label contains lowercase characters, we need to ensure that our SERVAUTH definitions are created with uppercase characters. Furthermore, because our label contains blanks, we need to represent those blanks with dollar signs ($). Such a representation of a label is called a "mapped label name." A mapped label name imposes two very strict requirements:

- All lowercase alphabetics in a label name must be converted to uppercase.
- Certain characters in a certificate label must be mapped to the dollar sign ($).

  The characters affected by this rule are: asterisk (*), percent sign (%), ampersand (&), and blank.

> **Note:** We executed the RDEFINE and the PERMIT statements from the TSO ISPF environment. As a result, even if we had entered the label in lowercase, our definitions would have been successful because TSO would have automatically converted the alphabetics to uppercase.

For more information about mapped label names, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

e. You are now familiar with mapped label names. We must build another profile for the Certificate Authority certificate that has signed the NSS client's certificate. We already know from a previous display that its label is My Local Certificate Authority.

Example 4-40 SERVAUTH Resource profile for NSS client's Certificate Authority certificate

```
RDEFINE SERVAUTH EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.CERTAUTH UACC(NONE)

PERMIT EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.CERTAUTH CLASS(SERVAUTH)
ID(NSS32A) ACC(READ)

SETROPTS RACLIST(SERVAUTH) REFRESH

SETROPTS GENERIC(SERVAUTH) REFRESH
```

Note again the dollar signs ($) in the mapped label name; they represent the blanks in the certificate label of the CA certificate. The syntax is similar to what you have seen before:

```
EZB.NSSCERT.<NSSD MVS System ID>.<NSS Client CA Certificate Label>.CERTAUTH
```

or

```
EZB.NSSCERT.<NSSD MVS System ID>.<mappedlabelname>.CERTAUTH
```

Hint: To avoid having to insert dollar signs ($) into mapped label names, avoid the use of the following characters when creating RACF certificates: *, %, &, and the blank character. To avoid the uppercase conversion requirement imposed by SERVAUTH definitions, always create RACF certificates in uppercase.

f. Example 4-41 shows additional SERVAUTH resource that you must define allow remote users to monitor (DISPLAY) or manage (CONTROL) NSS Clients.

Example 4-41 Resource to monitor and manage NSS clients remotely

```
RDEFINE SERVAUTH EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.CONTROL

RDEFINE SERVAUTH EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.CONTROL CLASS(SERVAUTH) ID(CS01,
CS02,CS03,CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

PERMIT EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.DISPLAY CLASS(SERVAUTH) ID(CS01,
CS02,CS03,CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Note the syntax of this SERVAUTH resource:

```
EZB.NETMGMT.<NSSD MVS System ID>.<NSS Client Symbolic Name>.IPSEC.CONTROL
```

and

```
EZB.NETMGMT.<NSSD MVS System ID>.<NSS Client Symbolic Name>.IPSEC.DISPLAY
```

g. In our case, we have a few remaining SERVAUTH resources to define. If your user, CS02, wishes to execute the **ipsec** command with the **-x** option, the user can display information about the NSS server. Therefore, on the server system, you need the resource profile and permissions for CS02 shown in Example 4-42 on page 145.

Example 4-42 SERVAUTH profile to display NSS information

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.NSS.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.NSS.DISPLAY CLASS(SERVAUTH) ID(CSO1,CSO2,CSO3,
CSO4,CSO5,CSO6,CSO7,CSO9,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

This syntax in Example 4-42 is quite different from what we have seen before:

```
EZB.NETMGMT.<NSS Server SYSID>.<NSS Server SYSID>.NSS.DISPLAY
```

Note that the NSS MVS System ID represents *two* of the fields in the resource profile.

If user CS02 wants to display information about the IKE daemon at the NSS server node with the **ipsec -w display** command, then the resource profile shown in Example 4-43 must be defined.

Example 4-43 Displaying IKE at server

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.IKED.DISPLAY CLASS(SERVAUTH) ID(CSO1,CSO2,CSO3,
CSO4,CSO5,CSO6,CSO7,CSO9,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Note: Up until now all the SERVAUTH resource profiles have been defined on the NSS server node, SC33. This pattern changes with the next SERVAUTH resource profile that we define on the NSS client node.

h. If user CS02 wants to display information about the IKE daemon's NSS clients, then the resource profile shown in Example 4-44 must be defined at the NSS client node. The command for this type of display is again **ipsec -w**.

Example 4-44 Displaying IKE at NSS client

```
rdefine SERVAUTH EZB.NETMGMT.SC32.SC32.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC32.SC32.IKED.DISPLAY CLASS(SERVAUTH) ID(CSO1,CSO2,CSO3,
CSO4,CSO5,CSO6,CSO7,CSO9,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Note: You may have found the numerous authorizations quite tedious. We show them in detail in order to emphasize several critical points:

- The numerous individual SERVAUTH profiles and authorizations provide a great deal of control and security. In order to understand them well, it is necessary to show you this detail. However, after you understand the concepts, you can very well use wildcards in many of the fields to reduce the amount of definition required.
- The definitions require detailed planning. As mentioned in “Overview of preliminary tasks” on page 129, two of the planning steps are to prepare a logical network diagram and to fill out a worksheet containing the information that you will need for these definitions. This preliminary work is very useful to expedite the steps covered in “Configuring authorizations for NSS” on page 140.
- If you have already worked with IPSec and IP Filtering, the task of setting up NSSD becomes quite simple.

4.3.7 Configuring the NSS server

The NSS server requires a configuration file, environment variables, a job or procedure with which to start the file, changes to the TCP/IP profile, an AT-TLS policy, TLS and IPSec certificates.

The NSS configuration file

The NSS server requires a configuration file that can be created using either of these methods:

- Copy nssd.conf from /usr/lpp/tcpip/samples into /etc/security and modify the copied file.
- Or, create the nssd.conf file with the Network Configuration Assistant.

Initially we chose to create our file manually with the sample file stored in the HFS. Later, we allowed the IBM Configuration Assistant to create the file so that we could compare the two methods.

Example 4-45 shows the pertinent parts of our NSS configuration file, which we stored in /etc/security as nssd33.conf.

Example 4-45 /etc/security/nssd33.conf: created from nssd.conf in /usr/lpp/tcpip/samples

```
#
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZANSCFG
#

NssConfig
{
# Port portNumber                      (dynamically modifiable)
# This is the TCP port that the Network Security Server will bind to.
# Default: 4159
Port 4159   a
# Sys 0-255                   (dynamically modifiable)
# Specifies the level of logging to obtain from the Network Security Server.
# To specify a combination of log levels, add the level numbers.
# The supported levels are:
# 0 - NSS_SYSLOG_LEVEL_NONE      - Disable Network Security Server syslog
#                                  messages
# 1 - NSS_SYSLOG_LEVEL_MINIMUM   - Minimal Network Security Server syslog
#                                  messages
```

```
# 2 - NSS_SYSLOG_LEVEL_VERBOSE   - Include cascaded internal error messages
#                                  (for IBM service)
# 4 - NSS_SYSLOG_LEVEL_CERTINFO  - Include info about certificate cache
# 8 - NSS_SYSLOG_LEVEL_CLIENTLIFECYCLE - Include info about client lifecycle
# 16 - reserved
# 32 - reserved
# 64 - reserved
# 128 - reserved
# Default: 1
# SyslogLevel 4
# SyslogLevel 15
  SyslogLevel 255 b

# This is the keyring holding the IKE certificates for IKED Server/Clients
KeyRing NSSD/NSSD_keyring c

# KeyRing userid/ringname             (dynamically modifiable)
# The owning userid and ringname used by the Network Security Server when
# performing RSA Signature Mode of authentication. The userid must be
# the userid of the process under which NSSD will run.
# There is no default value.  If KeyRing is not specified, then the Network
# Security Server cannot provide certificate services.
# KeyRing nssd/keyring
}
```

As you can see from the example, the configuration file is quite simple.

We specify the port that the NSS server is to bind to and listen on when it is initialized at a. Port 4159 is the default port.

We specify a SYSLOG level of 255 at b, which will help us with problem determination. The default SYSLOG level is 4; we set the level quite high to ensure that we trap all possible messages that could help us sort out any complexities we might run into.

We identify the NSSD key ring that holds the client certificates that are to be used for IKE RSA signature processing: NSSD/NSSD_keyring at c. Although unnecessary in our example, we specified the owning user ID of NSSD in front of the name of the key ring (NSSD/NSSD_keyring). The specification of the owning user ID is not necessary unless the user ID is different from that of the NSS procedure. Note that the instructions in the sample file and in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, indicate that the owner of the key ring must be the same as the owner of the NSS procedure.

Important: An NSSD procedure associated with an OMVS segment defined with UID=0 is generally insufficient for proper authority. To access the private key of server certificates, the owner of the key ring must be the same as the USERID under which NSSD is running—with one exception: If the key ring has been set up to be a shared keyring and it is populated with SITE certificates for the servers, then the owner of the key ring may be a USERID different from that under which NSSD or IKED is running.

Setting BPX_JOBNAME and BPX_USERID

We decided to start the NSS server with JCL, although you may choose to start it from /etc/rc. Therefore, in our case we omitted the following steps, which are documented in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Note: If you are starting the NSS server from the UNIX shell or from within /etc/rc, you will want to set some environment variables as follows:

```
# Start the NSS daemon
export NSSD_FILE=/etc/nssd33.conf
export _BPX_JOBNAME='NSSD' /usr/lpp/tcpip/sbin/nssd -f /etc/security/nssd33.conf &
export _BPX_USERID='NSSD'
unset _BPX_USERID
unset _BPX_JOBNAME
```

If you fail to do so, NSSD will not run with the proper authority. Furthermore, any administrator or user who sets _BPX_USERID must have access to the FACILITY class profile BPX.DAEMON.

Recommendation: We recommend starting NSSD as a procedure during or after TCP/IP startup, because its prerequisite processes are not usually initialized prior to this time frame.

NSS server authorization to RACF

We performed the various NSS server authorization tasks earlier in 4.3.6, “Configuring authorizations for NSS” on page 140.

SYSLOGD isolation for the NSS server

This step is detailed in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. However, we performed this step earlier as one of the initial tasks prior to setting up the NSS server. For information about isolating NSS server messages, see 4.3.1, “Overview of preliminary tasks” on page 129.

NSS server environment variables

This is an optional step. The following variables may be specified in the environment variables:

- NSSD_CTRACE_MEMBER
- NSSD_FILE
- NSSD_PIDFILE

Consult *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for more information about this topic.

We built a member called NSSD33 in our standard environment variable dataset. The member is depicted in Example 4-46.

Example 4-46 NSS standard environment variables

```
NSSD_FILE=/etc/security/nssd33.conf
NSSD_CTRACE_MEMBER=CTINSS00
```

Establishing the NSS server’s IKE key ring: NSSD_keyring

We created this key ring earlier and populated it with the IKE client certificates; refer to “Build the IKED certificates for the NSS client” on page 137.

Note: The key ring can be the same as the actual IKED key ring if it is populated with SITE certificates, or it can be one used solely by NSSD. Our IKE certificate for the new NSS client was an individual, Personal certificate. We therefore chose to build a new NSSD_keyring owned by NSSD.

Update the NSS catalogued procedure

There are several ways to initialize the NSS daemon, as described here.

- From an MVS procedure
  - This is the modified sample that we copied from hlq.SEZAINST(NSSD) into our MVS procedure library:

Example 4-47 NSSD procedure running on MVS image SC33

```
//NSSD PROC
//*
///NSSD EXEC PGM=NSSD,REGION=OK,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//        'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPE"', a
//        '"_CEE_ENVFILE=DD:STDENV")/')
//* Provide environment variables to run with the desired
//* configuration. As an example, the data set or file specified by
//* STDENV could contain:
//*
//* NSSD_FILE=/etc/security/nssd.conf
//* NSSD_CTRACE_MEMBER=CTINSS01
//* Sample MVS data set containing environment variables:
//STDENV DD DSN=TCPIP.SC33.STDENV(NSSD33),DISP=SHR   b
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
```

The NSS procedure runs as a generic server from an APF-authorized library. As a generic server, NSSD binds to all available TCP/IP stacks. However, in our case we wanted the NSS procedure to be associated only with one stack: TCPIPE. Therefore, we established stack affinity to TCPIPE with the variable _BPXK_SETIBMOPT_TRANSPORT=TCPIPE as you see at a in Example 4-47. Our standard environment file at b is depicted in Example 4-48.

Example 4-48 Standard environment file for Network Security Services server

```
NSSD_FILE=/etc/security/nssd33.conf
NSSD_CTRACE_MEMBER=CTINSS00
```

Language Environment® variables: We executed our NSS server with PGM=NSSD. There are restrictions regarding Language Environment variables that may require you to run the NSS server under BPXBATCH. Read about these restrictions in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

  - Remember to copy member CTINS00 from SYS1.IBM.PARMLIB and into the production PARMLIB if it is not already present.
- From the UNIX Shell or from within /etc/rc
  - Remember to specify _BPX_JOBNAME as previously recommended if you use this method.

- From the COMMNDxx member in hlq.PARMLIB

Note: Do not use AUTOLOG to start the NSS server. Although AUTOLOG will work, you will lose NSS cached information if the NSS procedure has already been initialized and then the AUTOLOG process causes it to cancel and restart.

Update the TCP/IP profile

There are potentially three tasks to complete in this section:

1. You should reserve the NSS port in the TCP/IP profile. We used the default of port 4159 as shown in Example 4-49 at a.

Example 4-49 Reserving the NSS port

```
PORT
....
   4159 TCP NSSD a;
   16310 TCP PAGENT            ;
....
```

2. The TCP/IP stack for NSS must be enabled for AT-TLS with the TCPCONFIG statement and TTLS parameter. Follow the guidelines for AT-TLS enablement as described for TN3270 in “Telnet security” on page 539 or for FTP in “Secure File Transfer Protocol” on page 579. You see an example of the appropriate TCPCONFIG parameter for AT-TLS in Example 4-50.

Example 4-50 AT-TLS stack enablement

```
TCPCONFIG TTLS
```

Note: You may need to perform this task for more than one stack, depending on whether you are allowing NSSD to establish affinity with more than one TCP/IP stack in the MVS image.

3. If the TCP/IP stack under which the NSS server is to run is also enabled for IPSec, then you may also want to update the IPSec default rules in the TCP/IP profile; see Example 4-51. Consult “Updating the TCP/IP stack to activate IPSec” on page 378 for information about how to accomplish this.

Example 4-51 Updating the TCP/IP profile for IP Security on behalf of NSS

```
IPCONFIG ... IPSECURITY ... b
;
NETMONITOR SMFSERVICE
;
; Added IPSEC statement
;
IPSEC  LOGENABLE
;  ;;  OSPF protocol used by Omproute
  IPSECRULE *  *  NOLOG  PROTOCOL OSPF
;  ;;  IGMP protocol used by Omproute
  IPSECRULE *  *  NOLOG  PROTOCOL 2
;  ;;  DNS queries to UDP port 53
 IPSECRULE *  *  NOLOG  PROTOCOL UDP SRCPORT * DESTPORT 53  SECCLASS 100
;  ;;  Administrative access
;IPSECRULE *  9.1.1.1 LOG  PROTOCOL *
 IPSECRULE *  10.1.100.221 LOG  PROTOCOL *
 IPSECRULE *  10.1.100.222 LOG  PROTOCOL *
```

```
 IPSECRULE *  10.1.100.223 LOG  PROTOCOL *
 IPSECRULE *  10.1.100.224 LOG  PROTOCOL *
;  ;;  Network security services (NSS) server access to the NSS client
;  ;;  Network security services (NSS) server access to the NSS client
 IPSECRULE  * * LOG  PROTOCOL TCP SRCPORT 4159 DESTPORT * c ; Network Security
;IPSEC6RULE * * LOG  PROTOCOL TCP SRCPORT 4159 DESTPORT * d ; Network Security
;
ENDIPSEC
```

At **b** in Example 4-51 on page 150, you see that we have enabled IPSECURITY on the IPCONFIG statement. At **c** you see that we have included default IPSEC rules on behalf of the NSS server for IPv4. At **d**, we commented out the IPSEC6RULE because we had not enabled IPv6 security in this stack.

Note: You may need to perform this task for more than one stack, depending on whether you are allowing NSSD to establish affinity with more than one TCP/IP stack in the MVS image.

Update the policy files at the NSS server and client nodes

You may need to consider up to two types of policy for NSS communications:

- You must have an AT-TLS policy for the NSS connections between server and client.
- You may need an IP Security policy. If IPSec has been enabled at the IP stack under which NSS runs, you will want to verify that the IPSec policy will allow communication between the NSS server and the NSS client. If it does not, you will need to build one.

We show you how to update the policy files required in 4.3.9, “Creating NSS files with IBM Configuration Assistant” on page 156.

4.3.8 Enabling an NSS client to use Network Security Services

The NSS client on SC32 is IKEDC. It is to interoperate with NSS on SC33, on behalf of TCP/IP stack TCPIPA on SC32; see Figure 4-33 on page 152.

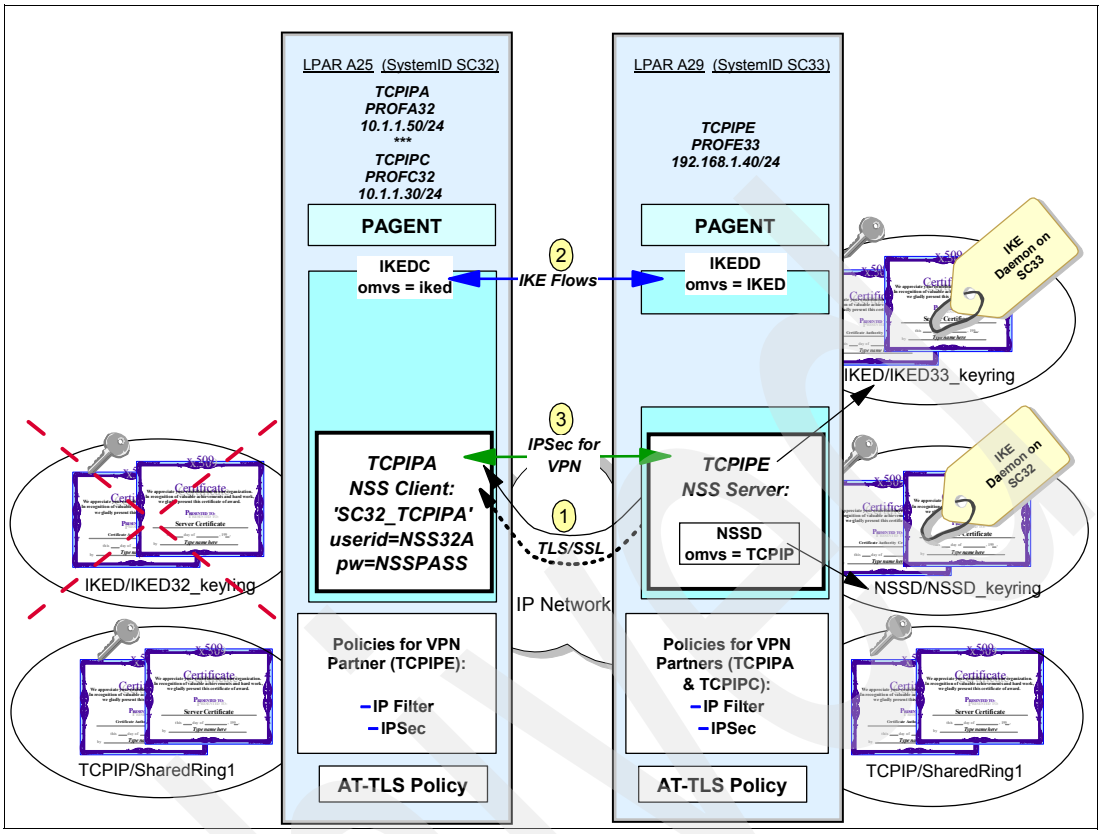


Figure 4-33 TCPIPA is NSS-enabled at system SC32

There are two ways to create the IKEDC definition file to reflect its usage of NSS for stack TCPIPA:

- You may use the IBM Configuration Assistant.
- You may directly edit the IKED configuration file.

We chose to build this file with the IBM Configuration Assistant. We show you how to use the IBM Configuration Assistant to build all the pertinent configuration files and policy files for both the NSS client and the NSS server.

However, for the purposes of describing the fields in the configuration file, we show you the text version of the configuration file in Example 4-52.

Example 4-52 ikedNSSClient.conf: IKED configuration file for an NSS client

```
##
## IKE Daemon Configuration file for:
##    Image: SC32
##
## Created by the IBM Configuration Assistant for z/OS Communications Server a
## Version 1 Release 9
## Backing Store = C:\Program
Files\IBM\zCSConfigAssist\V1R9\files\NSS_IPSEC_ATTLS.backingstore
...
IkeConfig
{
# IkeSyslogLevel    0-255             (dynamically modifiable)
# Specifies the level of logging to obtain from the IKE daemon.
# IkeSyslogLevel 1
```

```
IkeSyslogLevel 255 b

# PagentSyslogLevel 0-255             (dynamically modifiable)
# Specifies the level of logging to obtain from pagent through the PAPI
# PagentSyslogLevel 0
# PagentSyslogLevel 128
PagentSyslogLevel 255 c

# Keyring           userid/ringname   (not dynamically modifiable)
KeyRing IKED/IKED32_keyring d

# KeyRetries        1-10              (dynamically modifiable)
# Specifies the number of times that an unanswered key negotiation (Phase 1)
# message will be retransmitted before the negotiation is aborted.
KeyRetries 10

# KeyWait           1-300             (dynamically modifiable)
# Specifies the number of seconds between retransmissions
# of key negotiation (Phase 1) messages.
KeyWait 30

# DataRetries       1-10              (dynamically modifiable)
# Specifies the number of times that an unanswered data negotiation (Phase 2)
# message will be retransmitted before the negotiation is aborted.
DataRetries 10

# DataWait          1-300             (dynamically modifiable)
# Specifies the number of seconds between retransmissions
# of data negotiation (Phase 2) messages.
DataWait 15

# Echo              yes,no            (dynamically modifiable)
# Echoes all IKE daemon log messages to the job output file,
# specified by the IKEDOUT DD (JCL) statement.
Echo No

# PagentWait        0-9999            (not dynamically modifiable)
# The time limit in seconds to wait for connection to the policy agent.
# A value of 0 means retry forever.
PagentWait 0

# SupportedCertAuth label             (dynamically modifiable)
# SupportedCertAuth My Local Certificate Authority e
# Specifies the label of a Certificate Authority(CA) certificate on the
# IKE server's keyring.  Use multiple instances of this keyword to specify
# multiple CA certificates.
#######
NetworkSecurityServer 192.168.1.40 Port 4159 Identity X500dn "CN=ITSO.IBM.COM,OU=ITSO CS19
Shared SITE,O=IBM Corporation,C=US" f
#######
#NetworkSecurityServer 192.168.1.40 Port 4159 Identity X500dn "CN=ITSO.IBM.COM,O=IBM
#Corporation,OU=ITSO CS19 Shared SITE,C=US" g
#######
NssWaitLimit   60
NssWaitRetries 3
}
NssStackConfig TCPIPA h
{
   ClientName  SC32_TCPIPA i
   ServiceType Cert j
```

```
   ServiceType RemoteMgmt k
   UserId      NSS32A l
   AuthBy      Password NSSPASS m
}
```

Example 4-52 on page 152 shows that we will provide IKED services for all TCP/IP stacks on this system (MVS SC32) that have been IPSec-enabled through the IPCONFIG parameter IPSecurity. However, for stack TCPIPA at h, we will use Network Security Services. Examine each of these statements in Example 4-52 on page 152.

a This IKED configuration file has been built with the IBM Configuration Assistant.

b The initial level of IKED logging is to be very high. After testing is complete, we will revert to a lower level of IKED logging.

c The initial level of PAGENT logging is to be very high. After testing is complete, we will revert to a lower level of PAGENT logging.

d The key ring that contains the Certificate Authority certificate and the IKE certificates for RSA signature authentication is named IKED_keyring, and is owned by the superuser IKED. This key ring will be used by all stacks except for stack TCPIPA. TCPIPA, the NSS client, is to obtain key ring services from the NSS server.

e We have commented out the supported CA certificate named `My Local Certificate Authority` because we will accept any IKED certificates from partners that are signed by any CA certificate on the key ring. With SupportedCertAuth, the peer is “requested” to use an acceptable CA, but is not required to do so; as long as the CA exists on the key ring, any CA may be used, regardless of what is in the SupportedCertAuth list. The certificates with the named CAs are processed first.

f Up to this point, all the definitions apply to IKED in general. With the statement `NetworkSecurityServer`, we begin the definitions that apply to Network Security Services.

`NetworkSecurityServer` specifies the IP connection address and port of the NSS server that will be used for the NSS clients identified below the statement. It also provides the identity of the NSS server’s TLS certificate. (Note that this is *not* the identity of the NSS server’s IKED certificate!)

The x.500 Distinguished Name sequence at f is that under which the certificate has been stored in RACF. x500dn is also known as the Subject’s Name in a **racdcert** output display. You see the `Subject's Name` at 1 in Example 4-53 with the command **racdcert site list**.

Example 4-53 Output of racdcert site list for TLS Server Certificate of NSS Server

```
Label: CS19 ITSO SharedSite1
Certificate ID: 2QiJmZmiiaOFg8Pi8flAyePi1kDiiIGZhYTiiaOF8UBA
Status: TRUST
Start Date: 2007/09/11 00:00:00
End Date:   2008/09/11 23:59:59
Serial Number:
     >01<
Issuer's Name:
     >OU=ITSO Certificate Authority.O=IBM Corporation.C=US<
Subject's Name:
     >CN=ITSO.IBM.COM.OU=ITSO CS19 Shared SITE.O=IBM Corporation.C=US< 1
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
```

```
Ring Owner: TCPIP
Ring:
   >SharedRing1<
```

Important: The racdcert display separates the individual fields (also known as Relative Distinguished Names or RDN) of the Distinguished Name (DN) with periods as delimiters. The display pattern is:

RDN<period>RDN<period>RDN<period>RDN

However, note that the syntax required in the iked.conf file depicted in Example 4-53 requires a comma as a delimiter. The coding pattern in this case is:

RDN<comma>RDN<comma>RDN<comma>RDN

Therefore, the display output of CN=ITSO.IBM.COM.OU=ITSO CS19 Shared SITE.O=IBM Corporation.C=US must be converted to CN=ITSO.IBM.COM,OU=ITSO CS19 Shared SITE,O=IBM Corporation,C=US when it is defined after the Identity parameter of the iked.conf file.

We have provided an x500dn identity in the IKED client configuration file, shown in Example 4-52 on page 152, that is comprised of three RDNs. However, there are other RDNs that can comprise an x500dn identity. Table 4-1 lists other RDN attributed that are valid entries in a certificate definition.

Table 4-1 Table of attributes recognized by System SSL for certificate processing

Abbreviation	Meaning
C	Country
CN	Common name
DC	Domain component
E	E-mail address
EMAIL	E-mail address (preferred)
EMAILADDRESS	E-mail address
L	Locality
O	Organization name
OU	Organizational unit name
PC	Postal code
S	State or province
SN	Surname
SP	State or province
ST	State or province (preferred)
STREET	Street
T	Title

g Note that we have commented out the second `NetworkSecurityServer` definition in Example 4-52 on page 152. Although it appears to define the same TLS certificate that is

defined in f, it does not; the difference lies in the sequence of the fields in the x500dn parameter.

The sequence at g is the one that was used when defining the certificate, as you see in Example 4-54.

Example 4-54 Definition sequence for TLS server certificate of the NSS server

```
racdcert site gencert subjectsdn(cn('ITSO.IBM.COM')   -
   o('IBM Corporation')                              -
   ou('ITSO CS19 Shared SITE')                       -
   C('US'))                                          -
   NOTBEFORE(DATE(2007-09-11))                        -
   NOTAFTER(DATE(2008-09-11))                         -
   withlabel('CS19 ITSO SharedSite1')                -
   signwith(certauth label('CS19 ITSO CA1')
```

The defined sequence that you see in Example 4-54 does not always provide valid authentication; the *stored* sequence does. Therefore, if you use x500dn parameter for NSS server authentication on the Identity parameter of the NetworkSecurityServer statement, verify the stored sequence with the **racdcert site list** command output as we did in Example 4-53 on page 154.

h The **NssStackConfig** statement identifies the name of the TCP/IP stack or procedure that should request services of the NSS server. In our scenario, this is TCPIPA.

i The symbolic client name of the TCP/IP stack is SC32_TCPIPA. This symbolic name will be used by the server for SERVAUTH processing. Recall that you defined numerous SERVAUTH definitions and permissions in 4.3.6, “Configuring authorizations for NSS” on page 140.

j and k Two service types are valid for this NSS client: certificate management, and remote management via ipsec or an NMI application. Note that we have commented out the second `NetworkSecurityServer` definition in Example 4-52 on page 152.

l The user ID that will be presented to the NSS server for either password or pass token authentication is NSS32A. Recall that you defined this user ID in 4.3.6, “Configuring authorizations for NSS” on page 140.

m The password that was assigned to this user ID is NSSPASS.

4.3.9 Creating NSS files with IBM Configuration Assistant

In this section we show you how to use the IBM Configuration Assistant to build:

1. For the NSS server:
   a. The NSSD configuration file
   b. The AT-TLS policy for secure communication between the NSS server its clients
   c. Optionally, IPSec policy to permit traffic to and from the NSS server if the NSS server has been enabled for IPSec
2. For the NSS client:
   a. The IKED configuration file
   b. The AT-TLS policy for secure communication between the NSS client and the server
   c. The IPSec policy that permits communication with the NSS server

After initializing IBM Configuration Assistant on the workstation, you can open the Help panels for NSS by selecting **Help** --> **NSS Overview** from the Main Perspectives panels of the GUI. These operations take you to the panel shown in Figure 4-34.

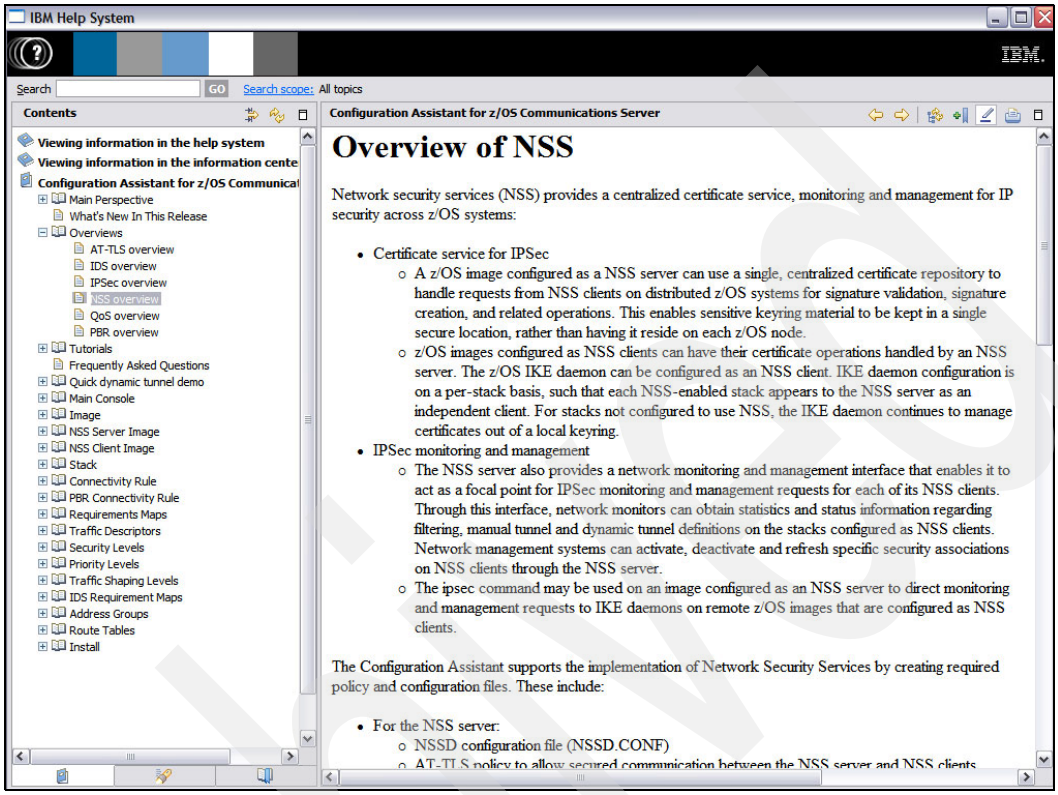


Figure 4-34 Help panels for NSS Overview in IBM Configuration Assistant

From this panel you can read through the overview or proceed to other selections in the left-hand navigation panel to learn how to code for NSS server and client images; see Figure 4-35 on page 158.

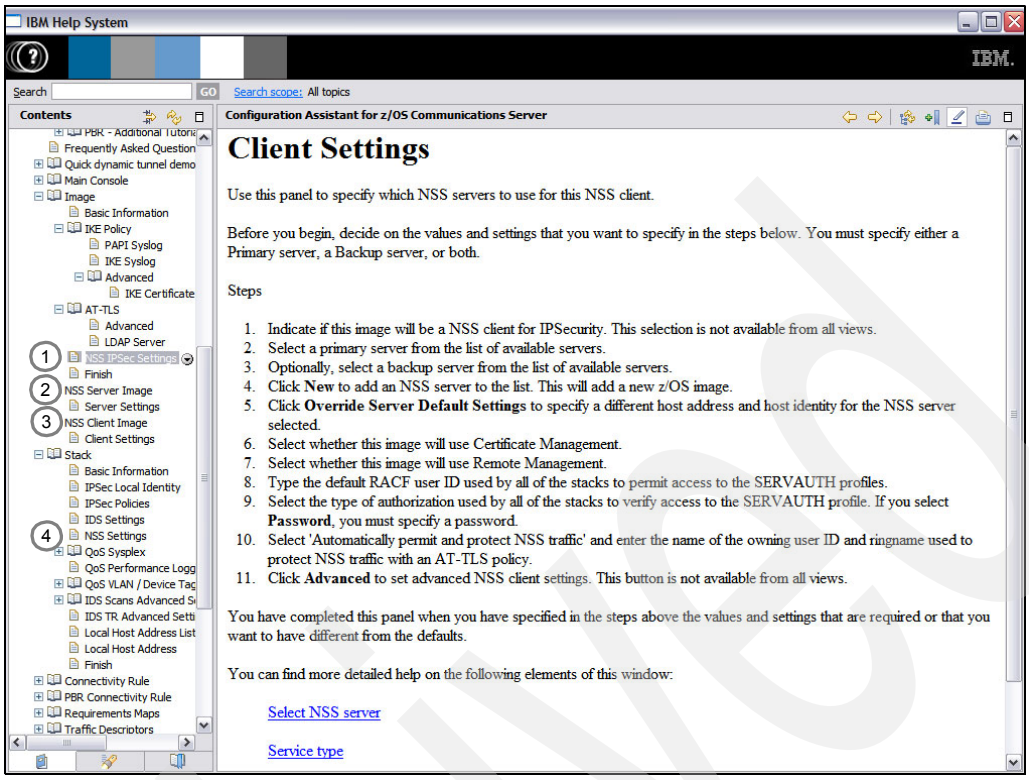


Figure 4-35 Additional Help panels for NSS in IBM Configuration Assistant

The line items indicated with 1, 2, 3, and 4 in Figure 4-35 represent other areas that can be defined for the NSS client and server. After perusing these panels, we closed them to return to the Main Perspective panel of the GUI.

Creating a new backing store file and merging old IPSec file into it

We already have Configuration Assistant backing store files for an IPSec implementation between SC32_TCPIPA and for SC33_TCPIPE. The name of this backing store file is Between-2zOS-RSA.backingstore, and it is the backing store file that we created in 4.2.4, “Modifying existing policies to use RSA signature mode” on page 104. Our plan is to keep that file as our backup and to create a new backing store file that includes Network Security Services.

Therefore, we create a new backing store in the IBM Configuration Assistant. From the Main Perspective we execute the following steps from the File pull-down:

```
File --> Open --> Create New Backing Store
```

We provide the name `Between-2zOS-RSA-NSS.backingstore`, as shown in Figure 4-36 on page 159.

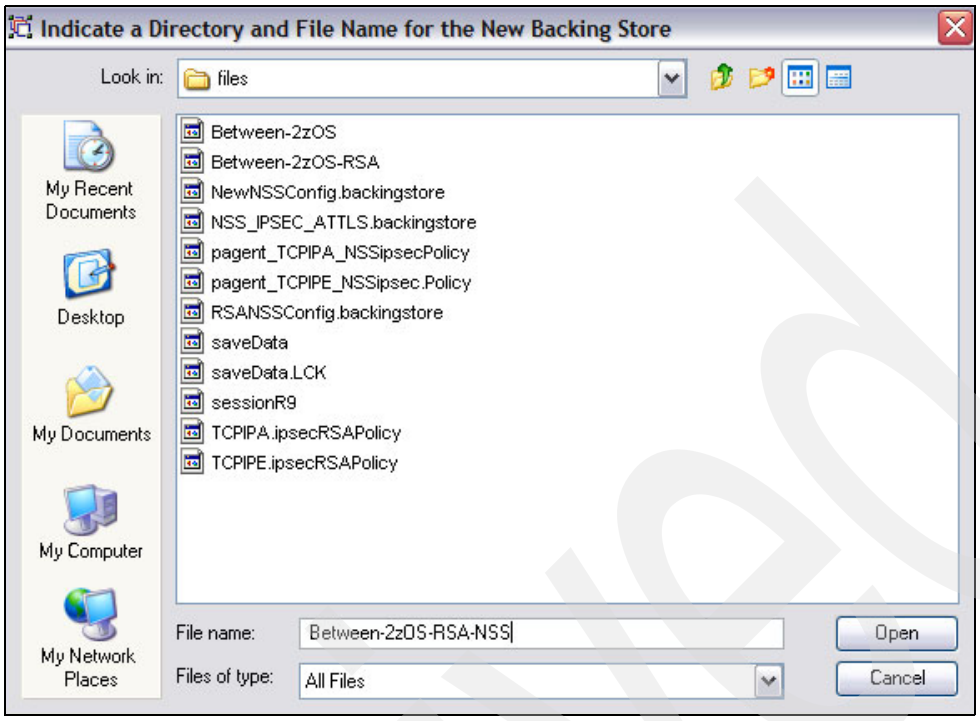


Figure 4-36 Creating a new backing store file for the NSS configuration

We press **Open**, which takes us back to the Main Perspective for the new backing store file.

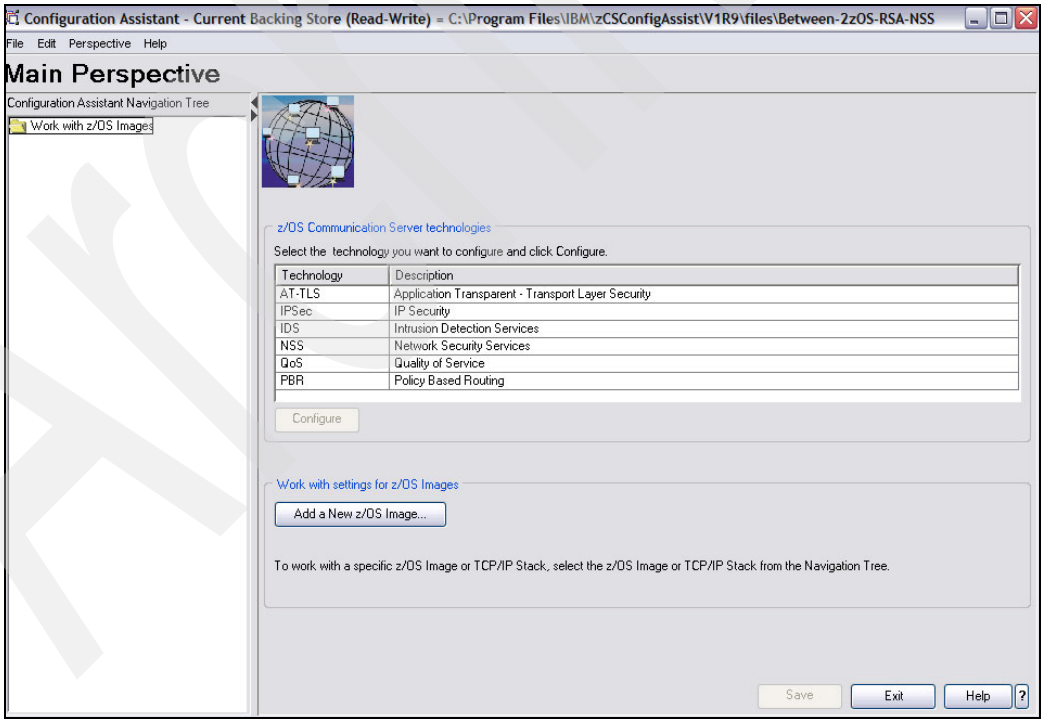


Figure 4-37 Beginning with a new backing store file

In Figure 4-37, notice that the top line of the panel shows that we are working on the new backing store we just created: `Between-2zOS-RSA-NSS.backingstore`.

We are now ready to import the working IPSec backing store that we previously used in 4.2.4, “Modifying existing policies to use RSA signature mode” on page 104. From the File pull-down, we execute **`File --> Import --> Local or shared file`** and import the backup of the existing file `Between-2zOS-RSA`, as shown in Figure 4-38.

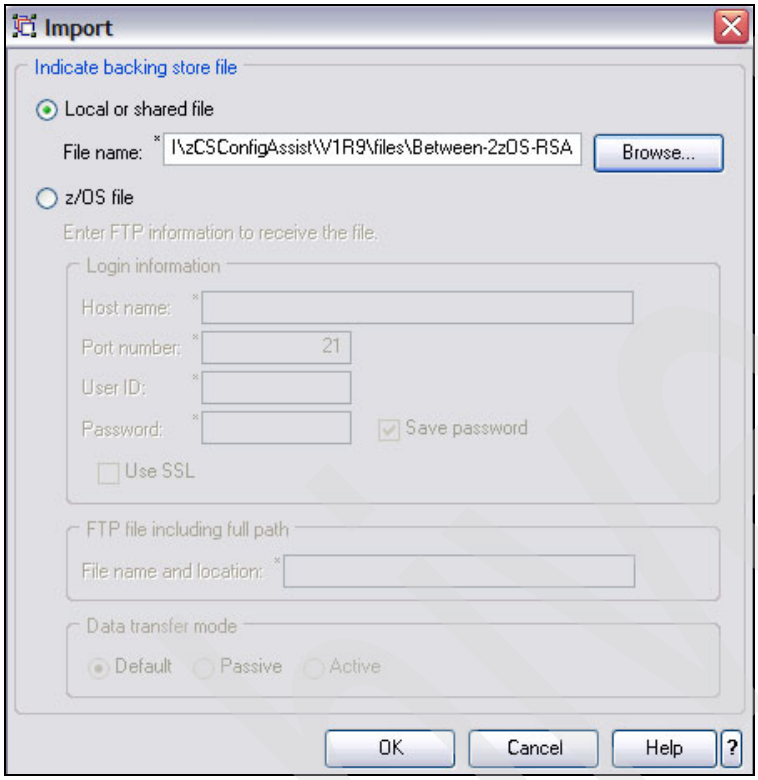


Figure 4-38 Importing an existing backingstore file into a new one

When we press **OK**, a message appears indicating the import was successful; see Figure 4-39.

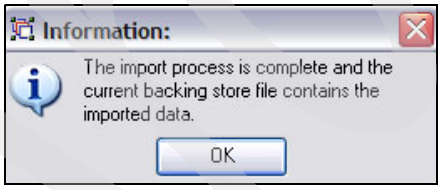


Figure 4-39 Successful import of a backingstore

Pressing **OK** from this panel presents a summary of the import activity, as depicted in Figure 4-40 on page 161.

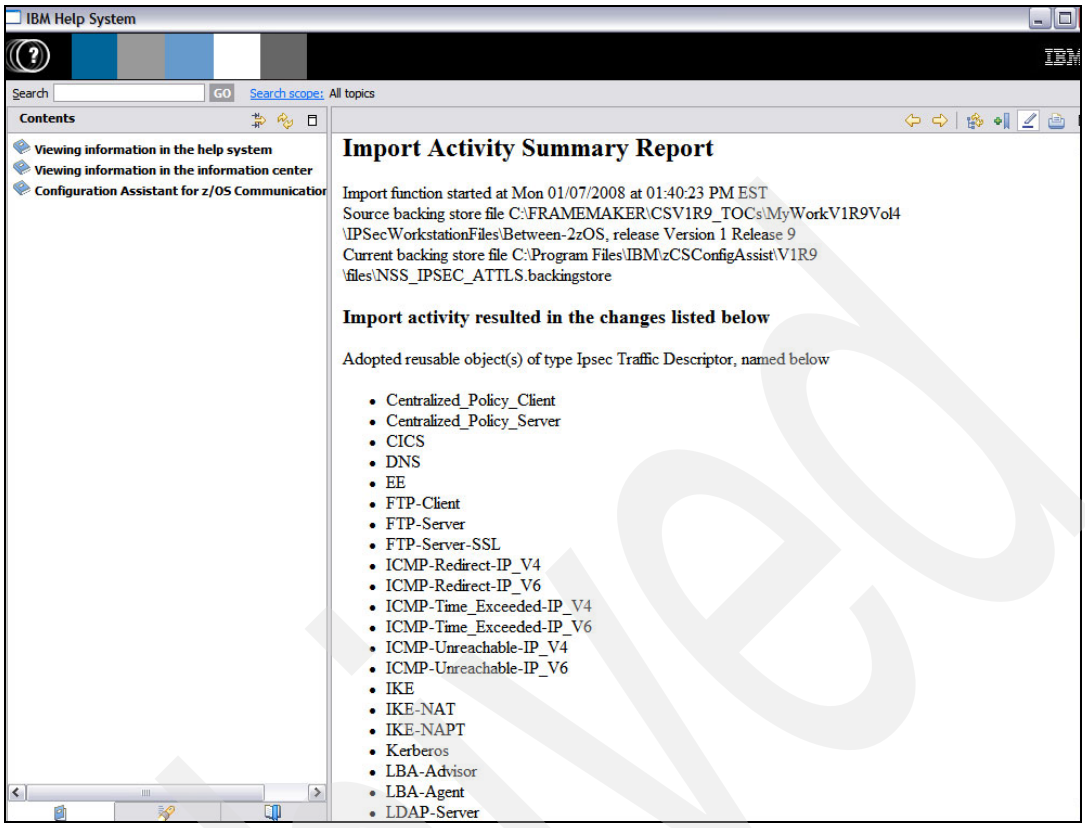


Figure 4-40 Import Activity Summary Report

Adding NSS to new backingstore

We close this Help panel and are now ready to add NSS to our new backingstore; see Figure 4-41 on page 162.

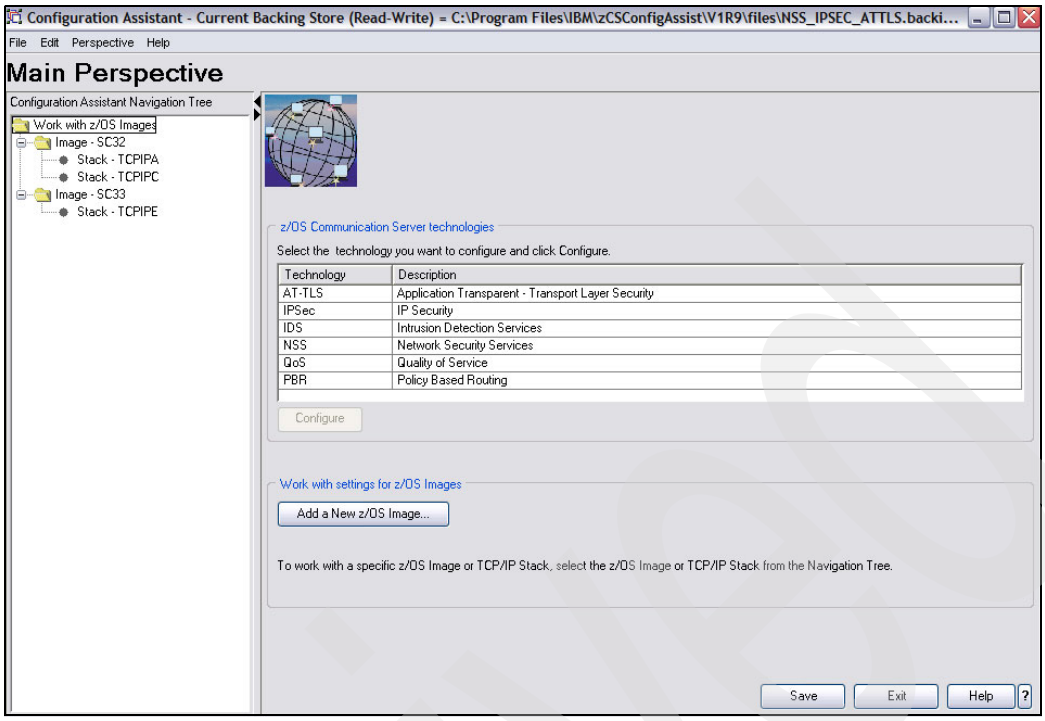


Figure 4-41 Starting point for NSS: successful backing store for IPSec

We highlight `Image SC33` and are moved to another panel where we select **NSS** and press **Enable**. The NSS configuration is showing as `Incomplete`; see Figure 4-42.

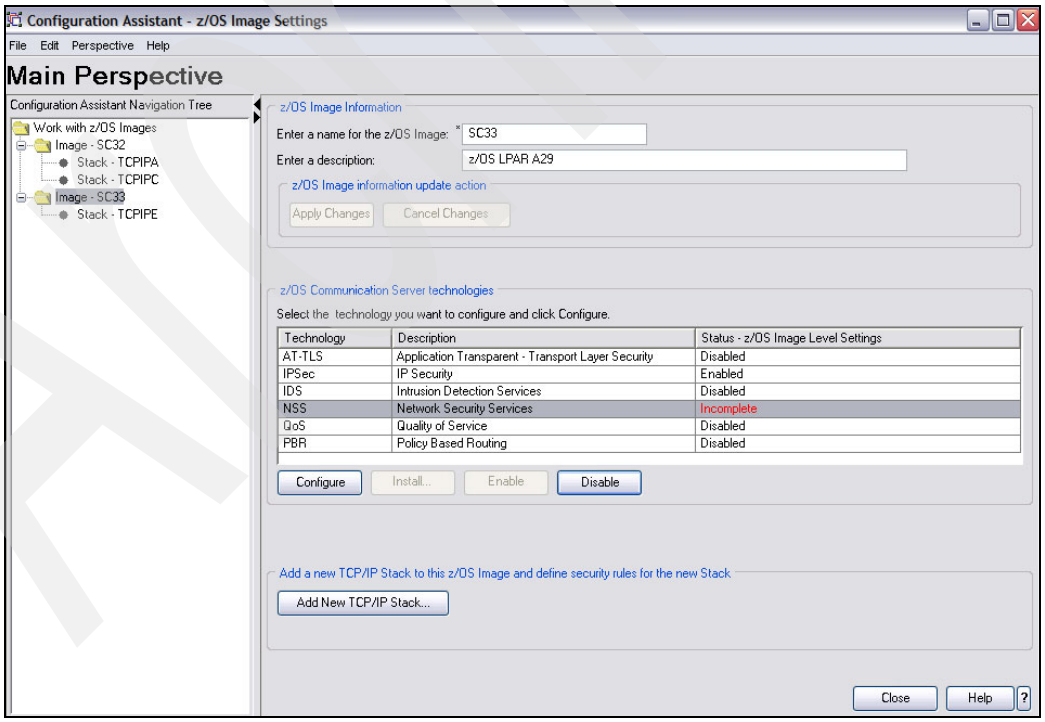


Figure 4-42 Begin to work on NSS for the NSS server at SC33

We therefore press **Configure** and are presented with the panel in Figure 4-43 on page 163.

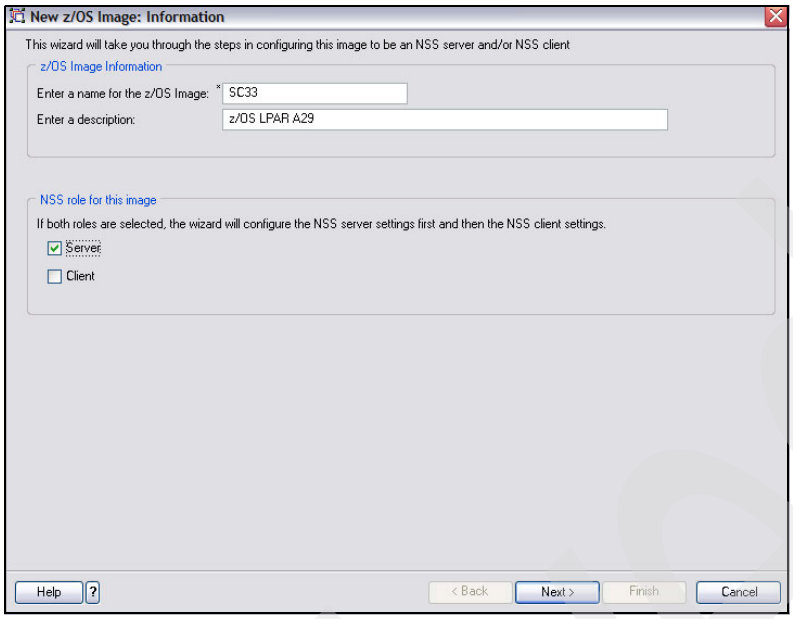


Figure 4-43 NSS role or roles for this MVS image

We select the **Server** role for SC33 and press **Next**. We fill out the values in the subsequent panel in Figure 4-44 on page 164.

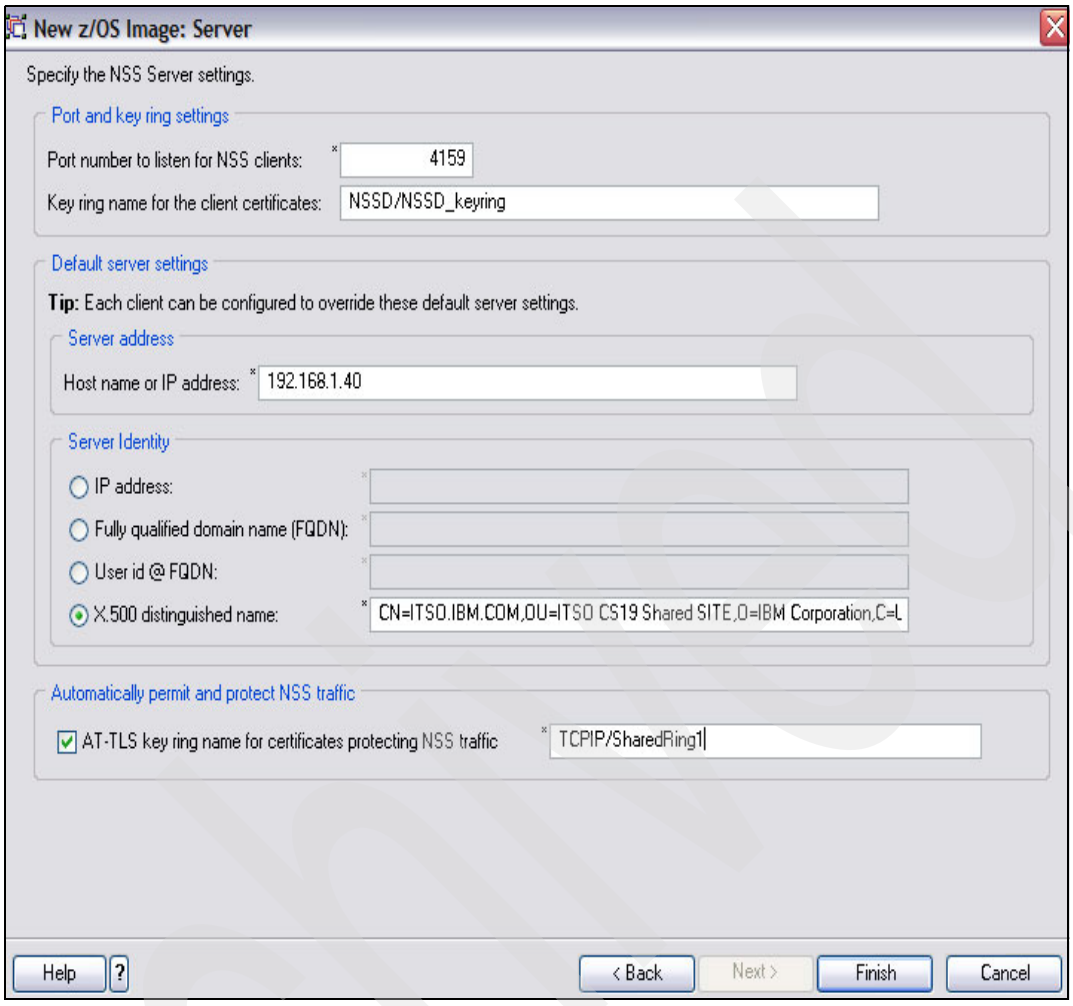


Figure 4-44 NSS definitions of the server

We filled in the name of the NSSD_keyring at the top of the panel and gave an IP address of the NSS server at TCPIPE on SC33 (`192.168.1.40`). We also copied and pasted the CN value of the NSS server's TLS server certificate into the X500dn field.

Important: This is the CN value from the TLS certificate, and *not* the CN value from the IKE certificate. Our TLS certificate was created as a SITE certificate.

We therefore used the **`racdcert site list`** command to obtain the x500dn value; we copied it from the command output and pasted it into the GUI field. Remember that the **`racdcer`**t output, shown in Example 4-53 on page 154, delimits the individual fields of the x500dn with periods. Notice in Figure 4-44 on page 164 that we changed the periods to commas in the IBM Configuration GUI.

Figure 4-44 also shows that we included the owner and name of the TLS key ring that the NSS server will use during secure communications with the client.

Common mistake: The panel in Figure 4-44 on page 164 calls for two key ring names for the NSS Server node: the NSS or IKE key ring that the server uses, and the TLS keyring that the server uses. Do not confuse the two key rings.

We press **Finish** on this panel and are returned to the NSS Perspective; see Figure 4-45.

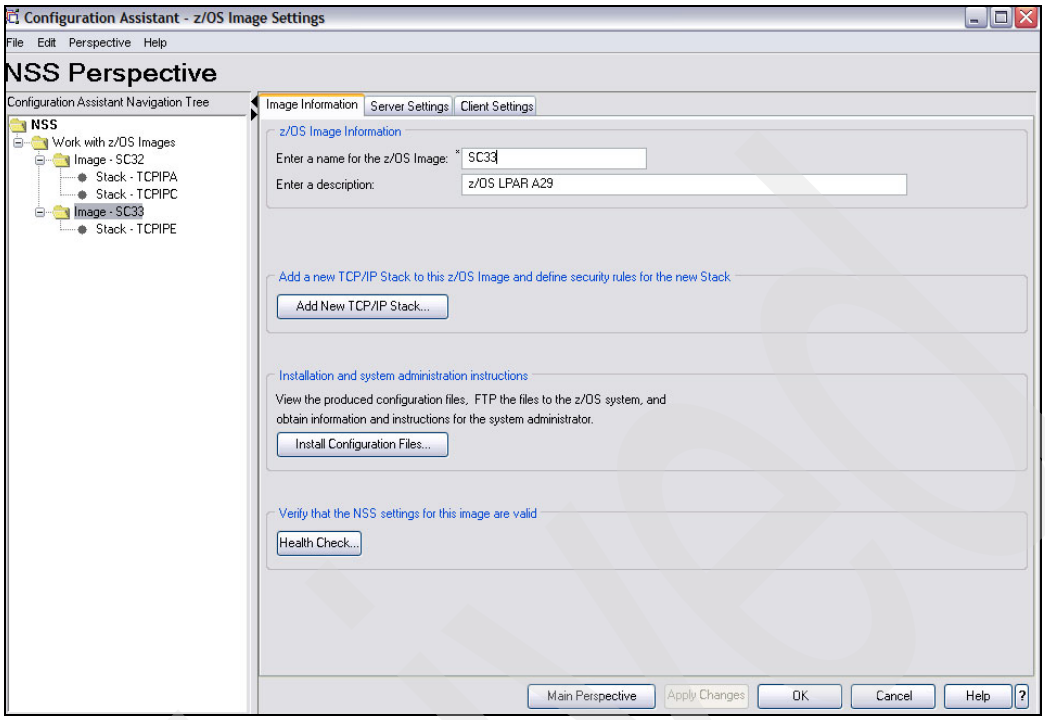


Figure 4-45 Completed NSS perspective for the SC33 image

From this panel we run **Health Check...**, which tells us in informational messages that we do not yet have any NSS clients. We close the Health Check panel and return to the Main Perspective, shown in Figure 4-46.

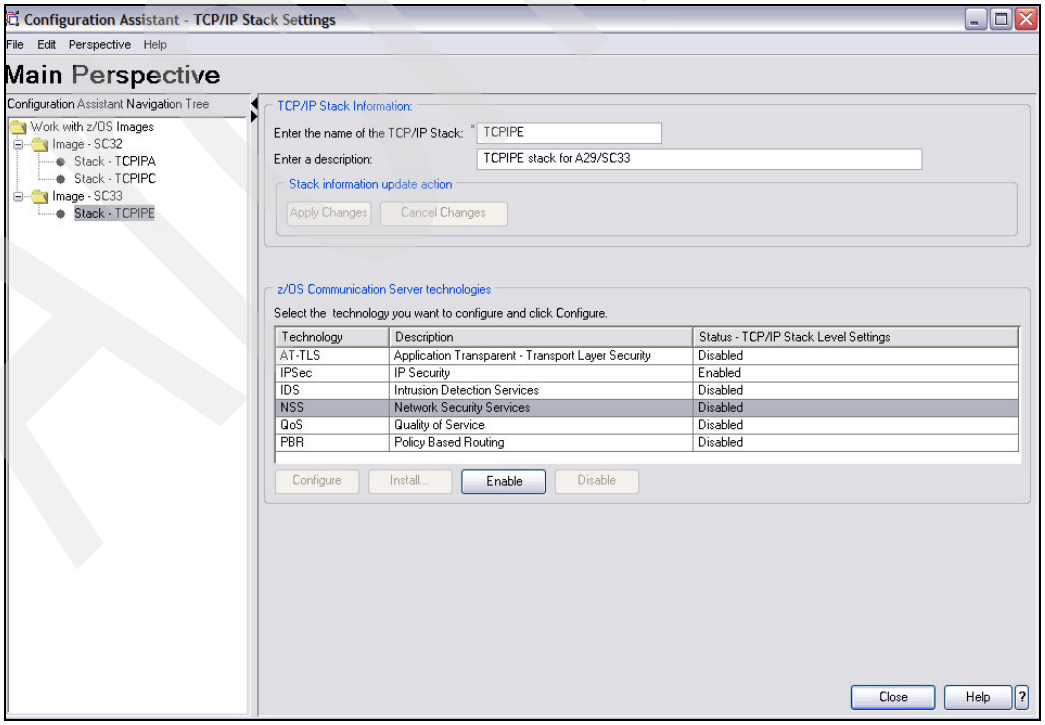


Figure 4-46 TCPIPE is disabled for NSS: enable it

After selecting the TCPIPE stack, we see that it is disabled for NSS. Therefore, we select **NSS** from the panel and press **Enable**.

Next, we select **Image - SC32**, the client image, where we enable NSS at the client as shown in Figure 4-47.

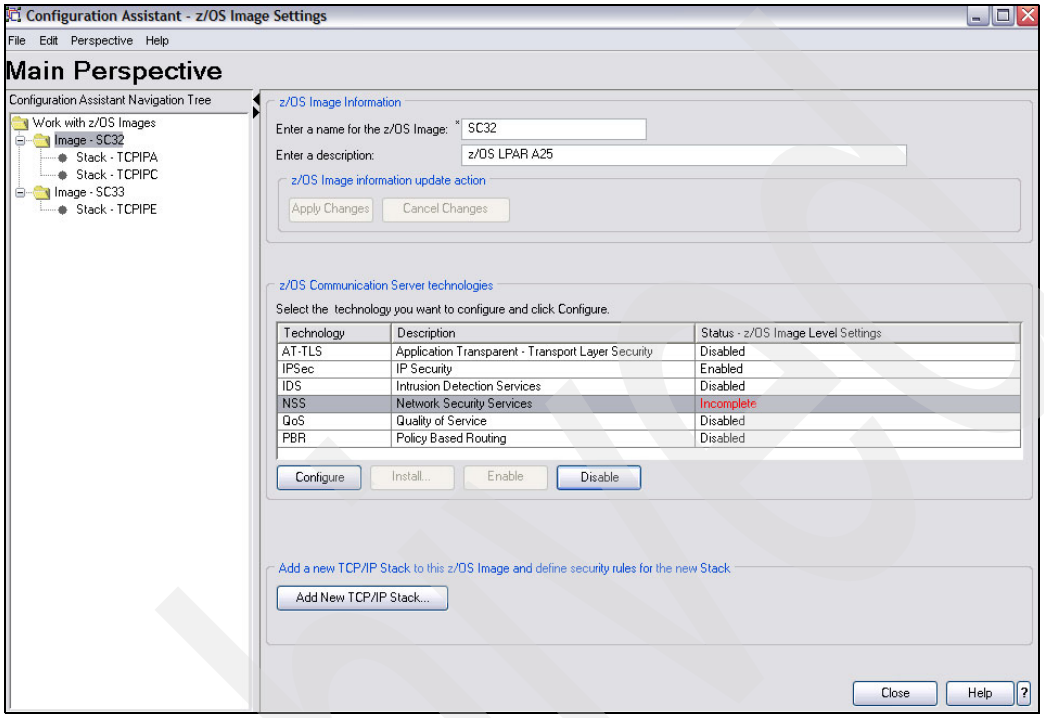


Figure 4-47 Enable Client image for NSS

Selecting **Configure** takes us to the subsequent panel, where we identify image `SC32` as the client; see Figure 4-48.

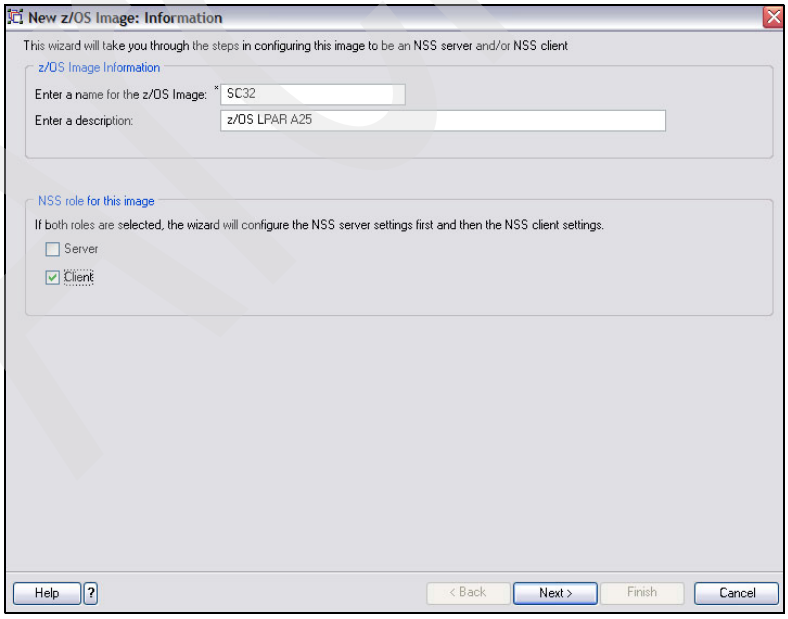


Figure 4-48 Identifying the NSS client role of SC32

We press **Next**, which moves us to Figure 4-49.

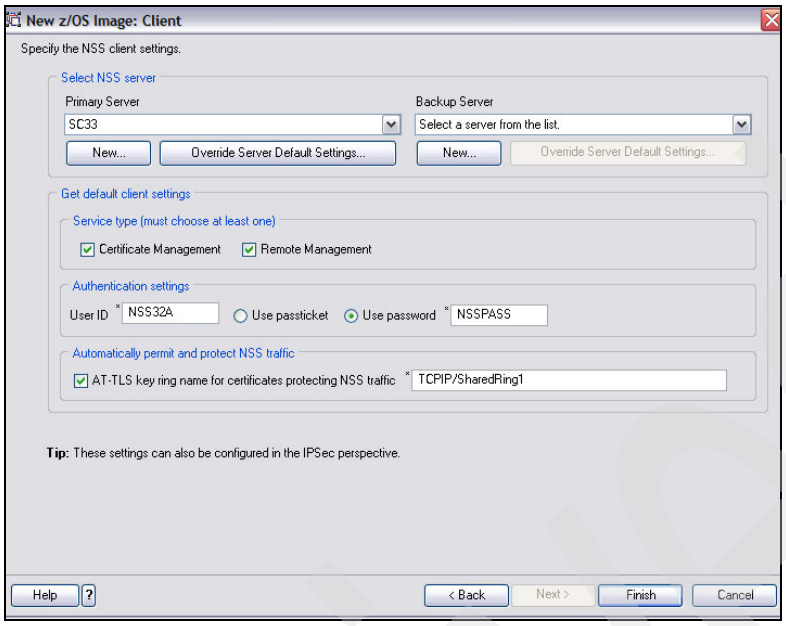


Figure 4-49 Identifying the NSS server, the NSS client user ID, and the client AT-TLS key ring

From the Primary Server pull-down we select **SC33** as our NSS server. With a checkmark we request both types of Network Security Services: Certificate Management and Remote Management. We fill in the user ID that we want to assign to the NSS client (`NSS32A`), and we request password authentication with a password of `NSSPASS`. Finally, we fill in the name of the client's AT-TLS certificate key ring (`SharedRing1`) and press **Finish**.

Note: The selections we made in Figure 4-49 on page 167 were planned by us when we created the prerequisite environment for NSS and its clients. Refer to 4.3.6, "Configuring authorizations for NSS" on page 140 for more detailed information about this topic.

We run Health Check and discover from the resulting messages that we have not enabled any client TCP/IP stack yet; see Figure 4-50 on page 168.

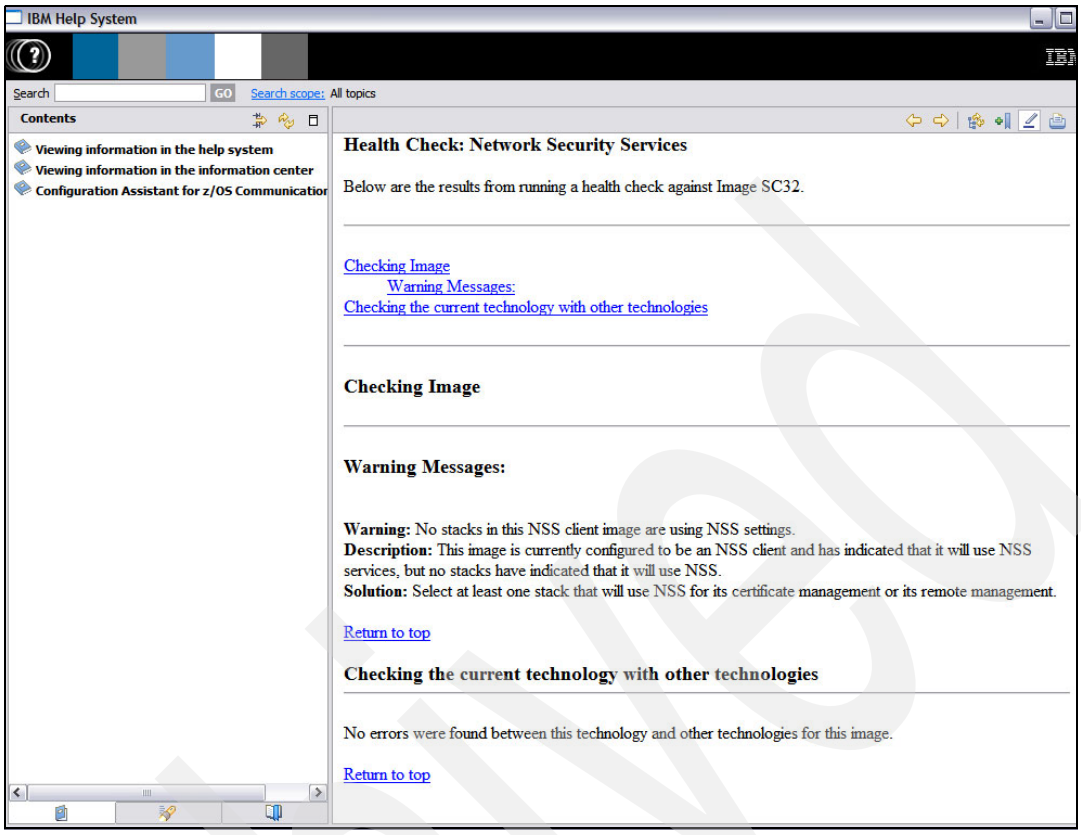


Figure 4-50 Warning messages: no NSS client TCP/IP stacks yet

So we close the Help panel and return to the Main Perspective. Here we select the TCPIPA stack, but it is still `Disabled` for NSS; see Figure 4-51 on page 169.

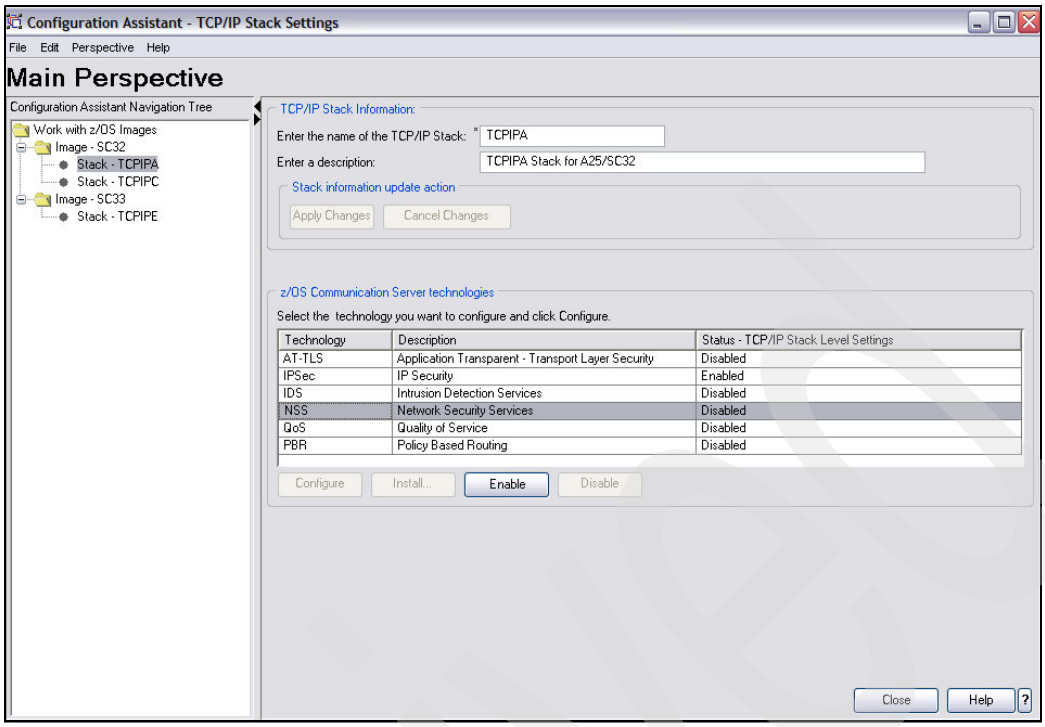


Figure 4-51 TCPIPA is not yet enabled as an NSS client

We highlight **NSS** in the middle of the panel and press **Enable**, which takes us back to the NSS Perspective. We select the Client IPSec settings tab, check the box for IPSec, and then fill in the fields.

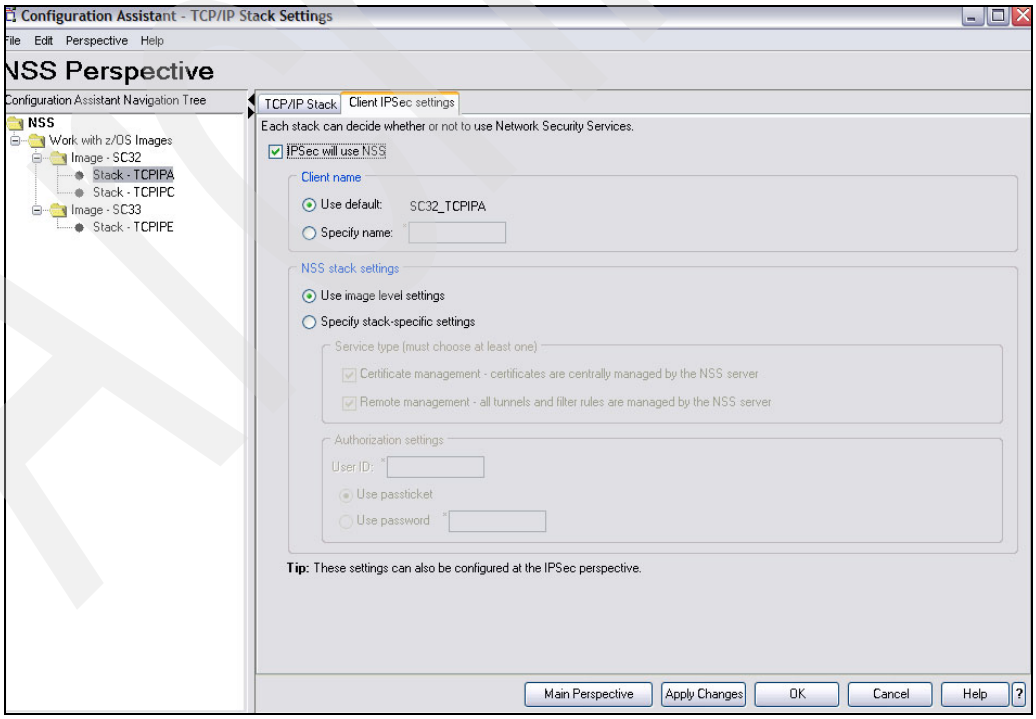


Figure 4-52 NSS client identity: symbolic client name

We leave the default in place for the NSS symbolic client name of `SC32_TCPIPA` and press **Apply Changes** and then **OK**. This takes us to a panel from which we can select an **NSS Summary**, as displayed in Figure 4-53.

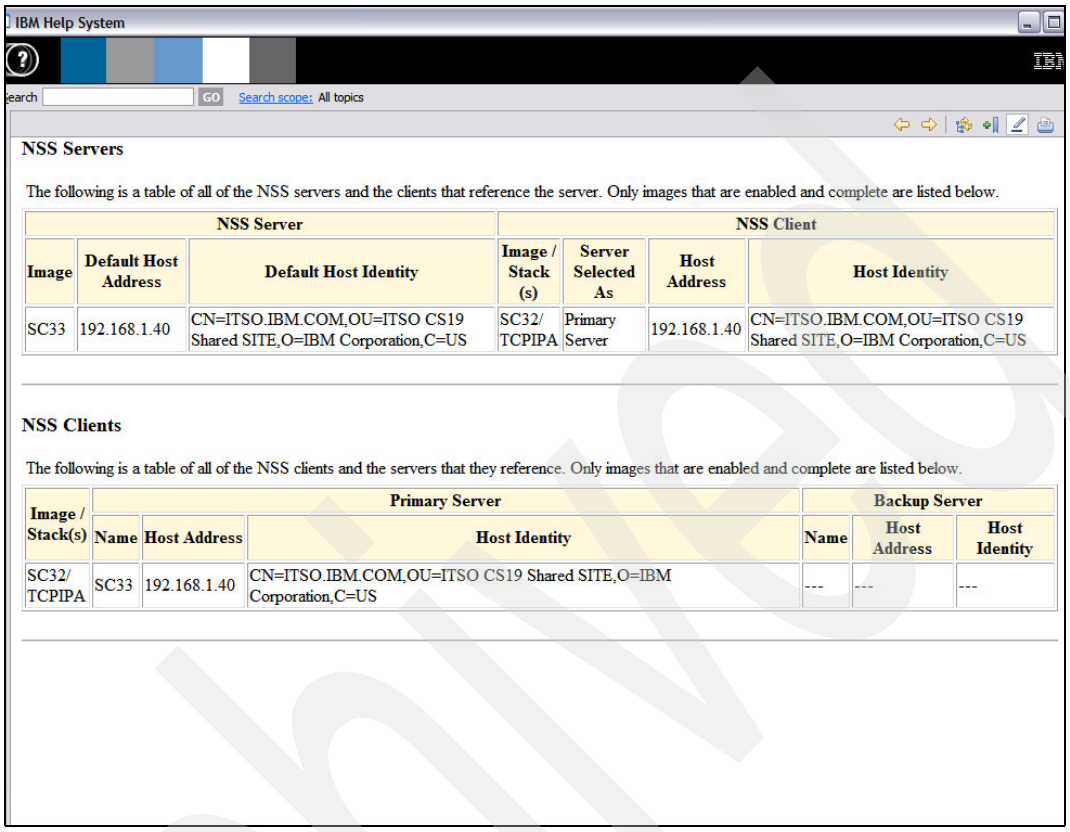


Figure 4-53 Summary of NSS definitions in this backing store file

We close this Help window and highlight **TCPIPC** in Image - SC32; see Figure 4-53.

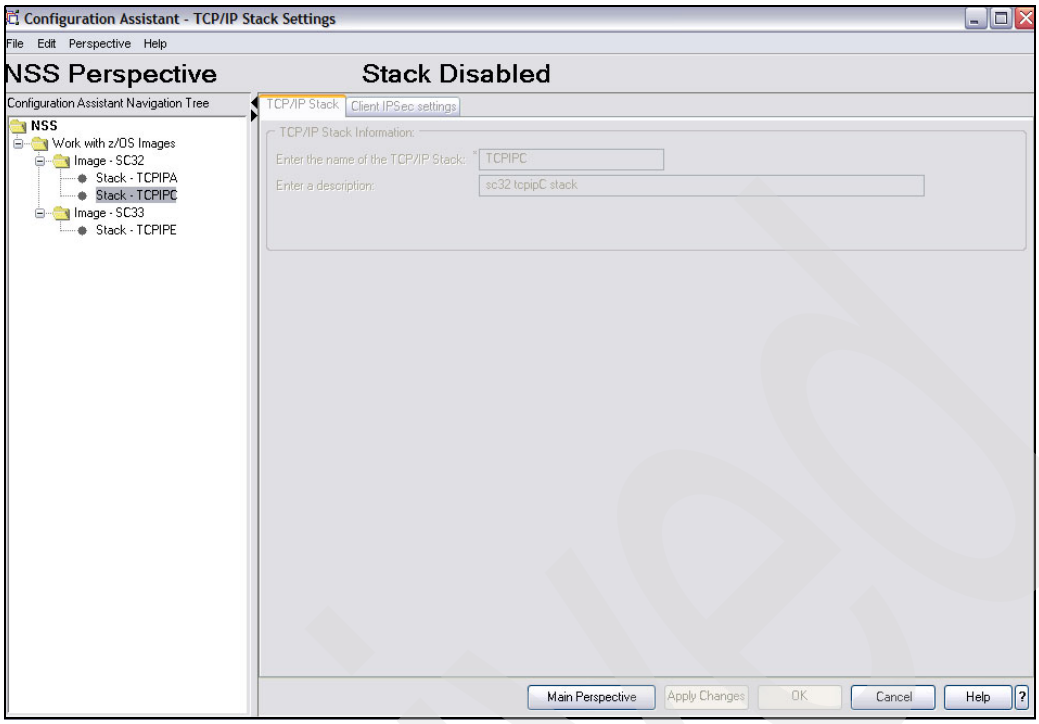


Figure 4-54 NSS Perspective: TCPIPC stack is not enabled for NSS

So far only the TCPIPA stack is enabled for NSS. Although there is a second stack in image SC32, we are going to allow that stack to use native IKED services and will leave it disabled for NSS. We return to the Main Perspective and highlight SC33 (`TCPIPE`); see Figure 4-55.

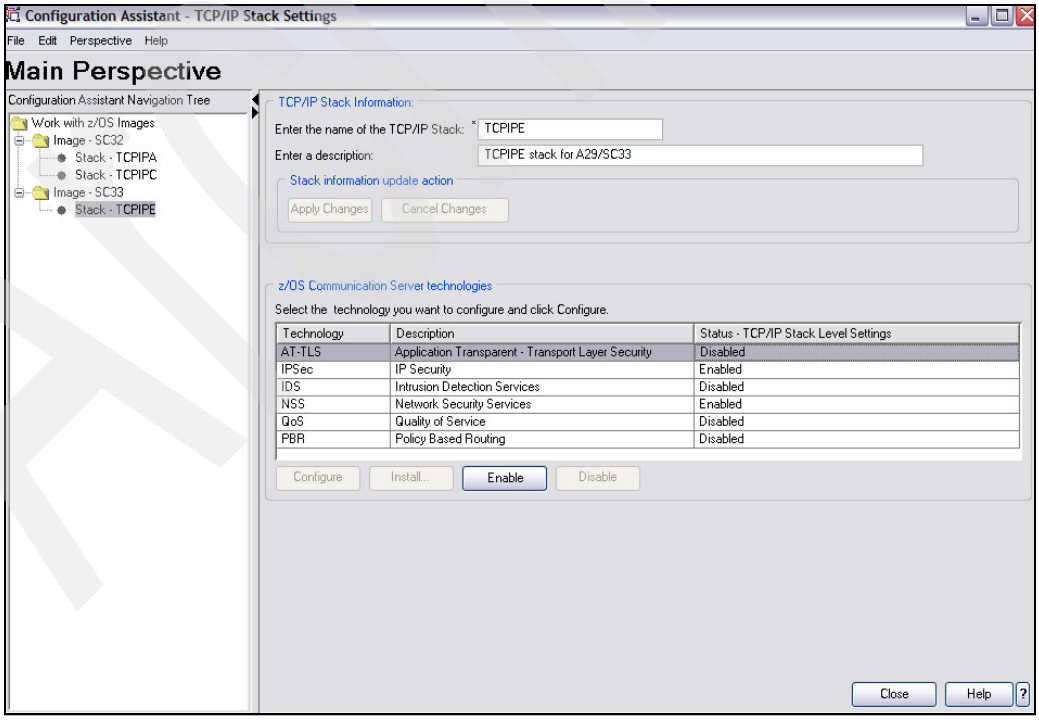


Figure 4-55 AT-TLS is not yet enabled at TCPIPE, the NSS server stack

Note in Figure 4-55 on page 171 that TCPIPE is enabled for both IPSec and for the NSS server function. We know from previous exercises that the IPSec definitions between TCPIPE and TCPIPA are functional; therefore, we only need to verify that the IPSec connectivity rules that are already defined are valid for communication between the NSS server at 192.168.1.40 and the NSS client at 10.1.1.50. In our next steps we verify the IPSec connectivity rules that exist for the NSS server and client.

Verifying and augmenting existing IPSec policy for NSS

From the Main Perspective, we switch to the IPSec Perspective. We highlight the `TCPIPE` (NSS server) stack and look for a connectivity rule entry that could apply to the relationship between the NSS server and the NSS client, as shown in Figure 4-56.

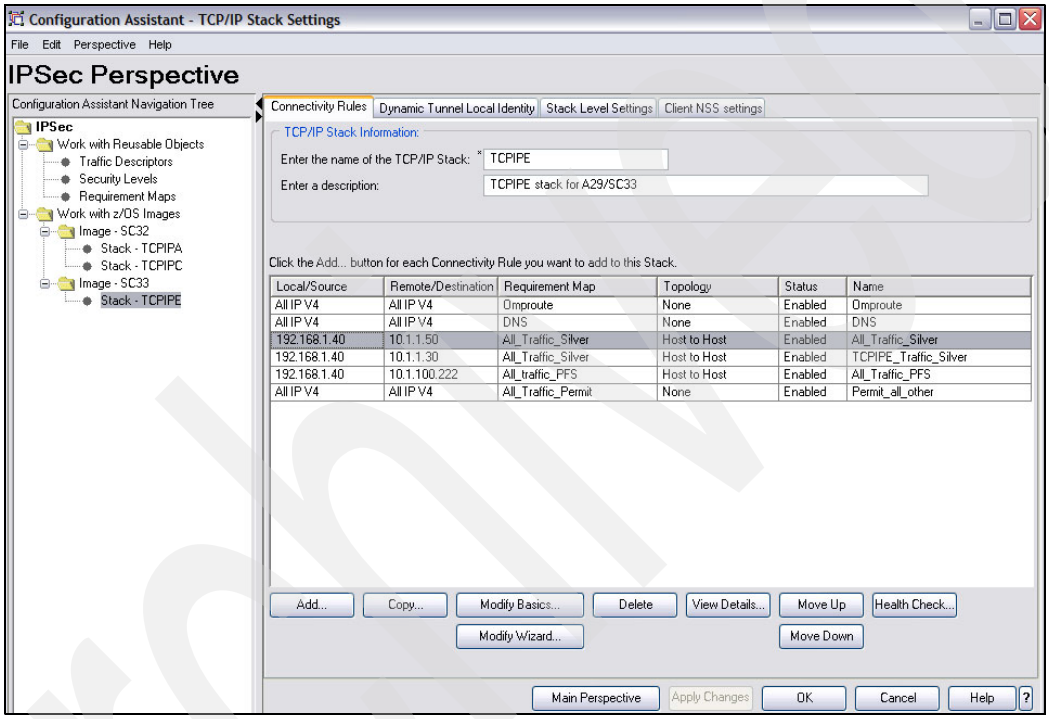


Figure 4-56 IPSec Perspective: Connectivity rules for NSS Server and Client

We click **View Details** of the highlighted connectivity rule to determine whether it is appropriate for NSS traffic.

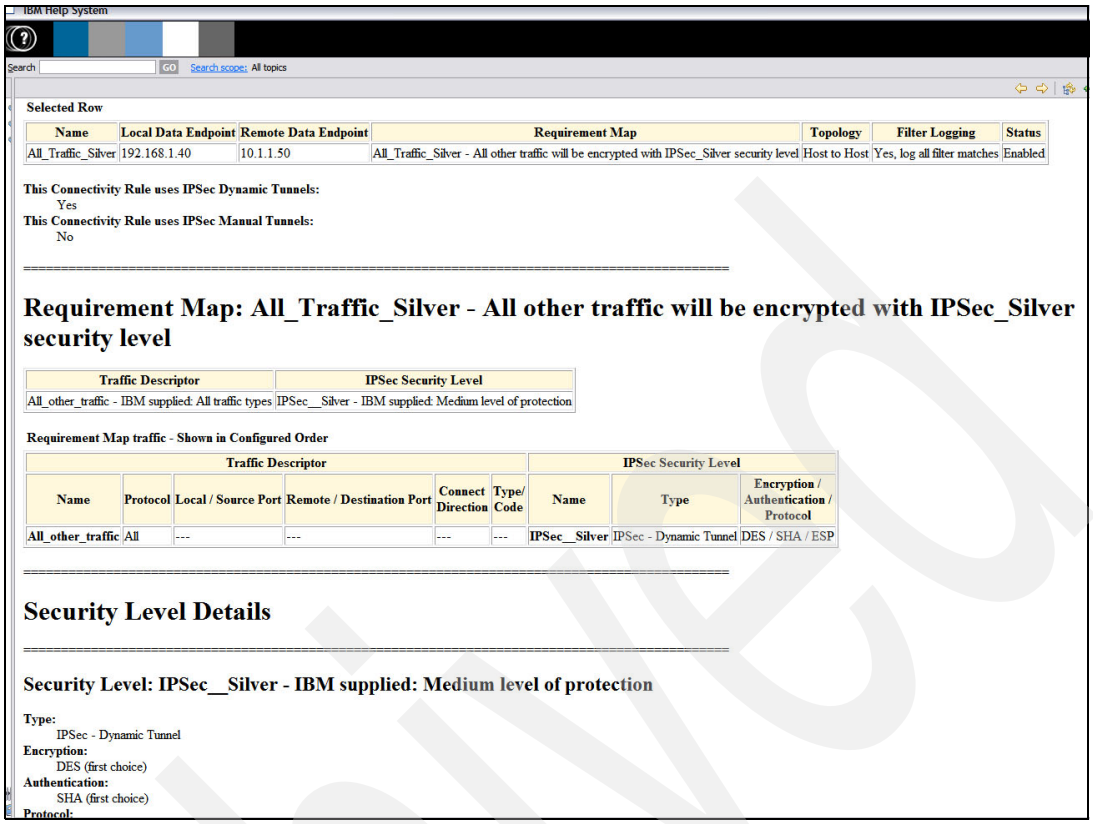


Figure 4-57 IPSec Connectivity: all traffic is encrypted

Note on the Requirement Map that all the traffic between the two addresses is encrypted. However, NSS uses TLS for encryption, which may mean that you do not want to encrypt both for the TLS connections as well as with IPSec, because there could be a performance impact.

Therefore, we need to add a new Connectivity Rule to our policies; these IPSec rules should specify no encryption of data to and from Port 4159 (the NSS server port).

We begin by closing the Help panel, which returns us to the Connectivity Rules panel of IPSec; see Figure 4-58 on page 174.

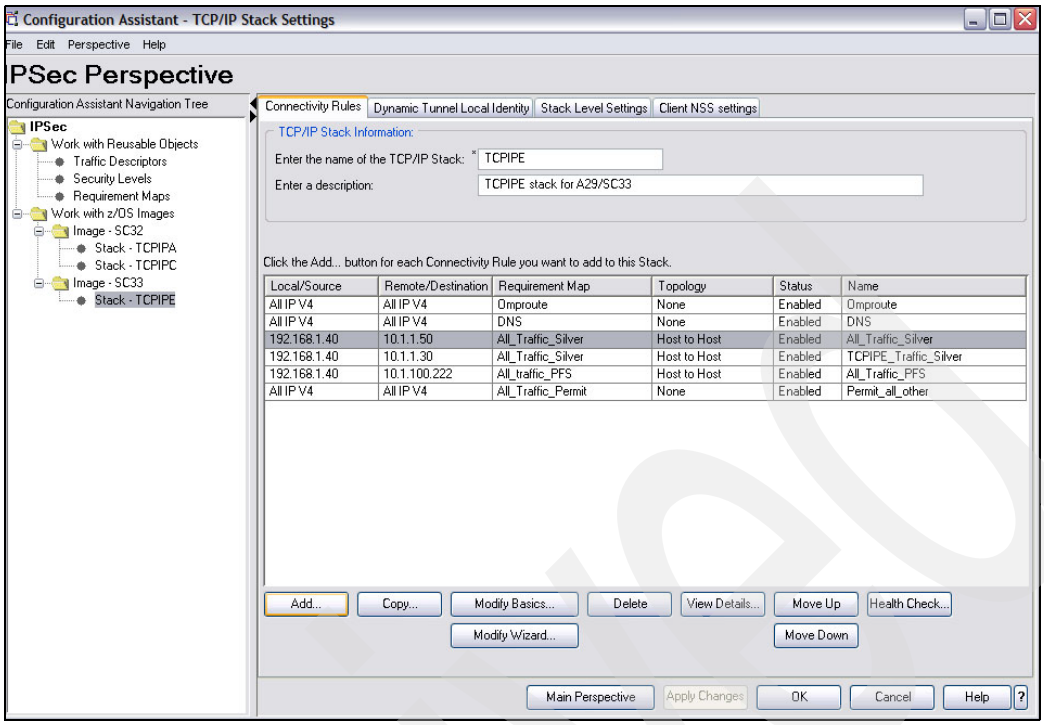


Figure 4-58 Connectivity Rules for IPSec; add a new rule to prevent double encryption for NSS

We select **Add** from the bottom of this panel and select **Next** from the subsequent Welcome panel. Now look at Figure 4-59.

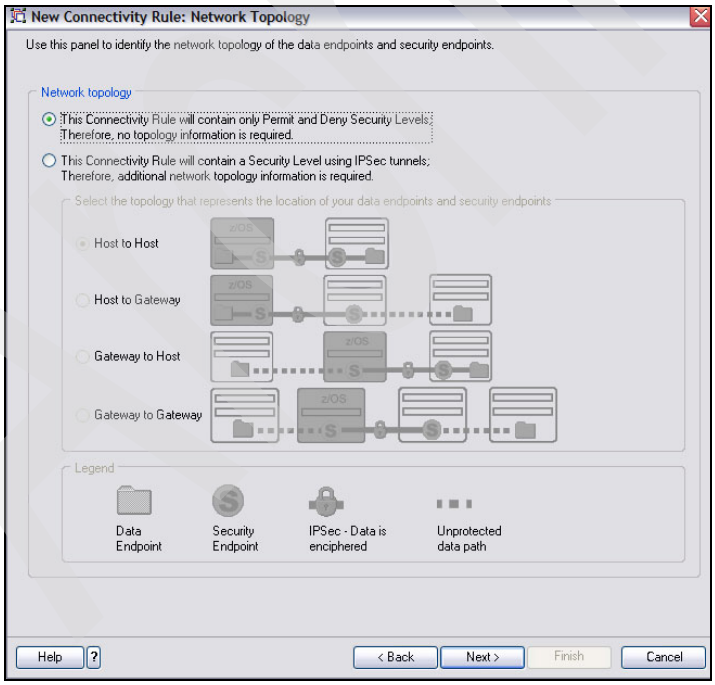


Figure 4-59 NSS Connectivity Rule: select radio button for no encryption

After selecting **Next** from Figure 4-59, we are presented with Figure 4-60 on page 175.

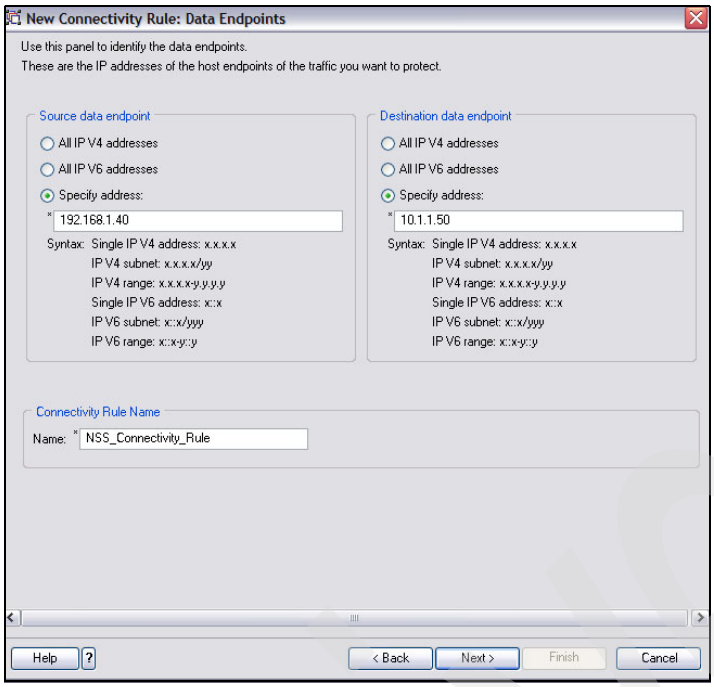


Figure 4-60 NSS Connectivity Rule

We fill out the data endpoints and a new rule name of `NSSConnectivityRule` and press **Next**, which takes us to a panel asking us to select or add a Requirement Map.

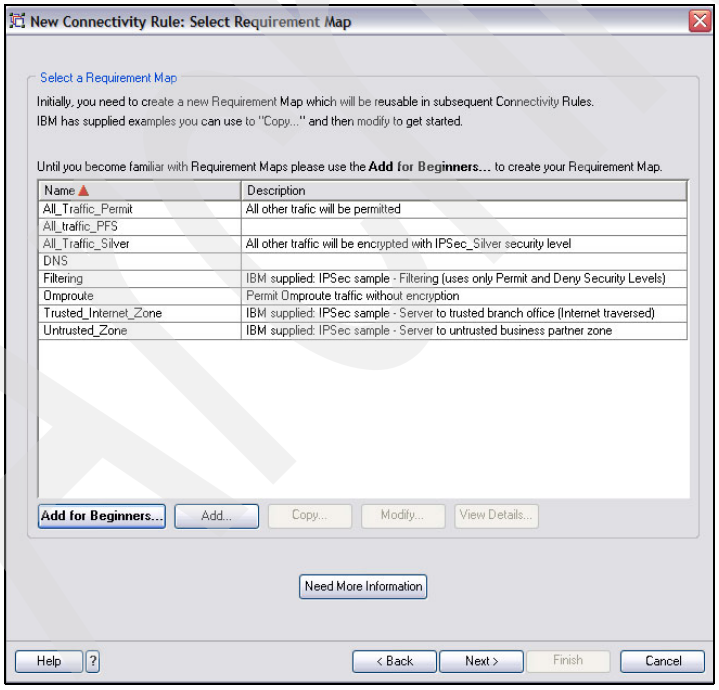


Figure 4-61 Add a Requirement Map for NSS

We press **Add for Beginners...** and **Next**, which presents us with the following panel.

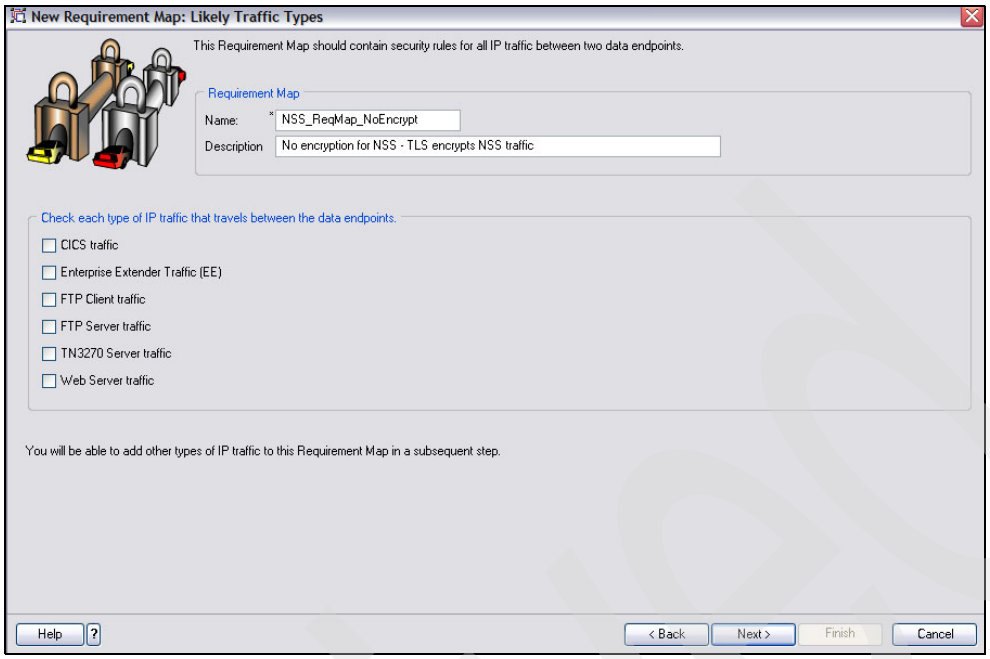


Figure 4-62 Naming the new requirement map for NSS traffic

In our case, we decided to build a single Requirement Map for both the NSS client and server. If your security requires more granularity, you may choose to build two separate Requirement Maps: one for the client and one for the server.

We fill in the name of the Requirement Map and press **Next**; see Figure 4-62.

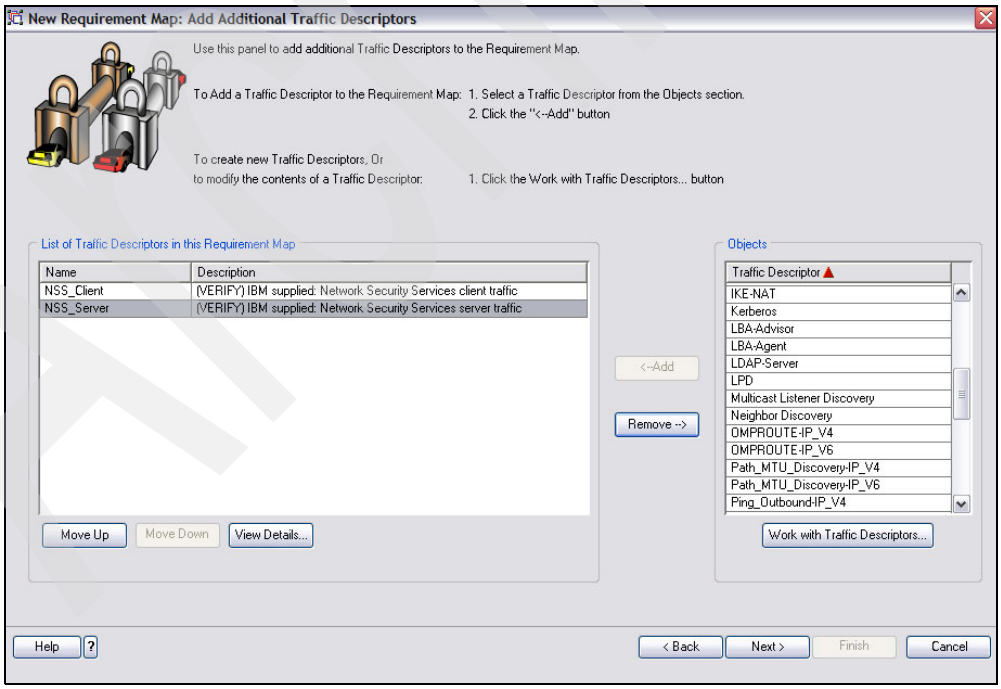


Figure 4-63 Adding Traffic Descriptors to NSS requirement map

In Figure 4-63 you see that we have already selected the NSS Server and NSS Client traffic from the right panel of traffic descriptors and added them to the Requirement Map in the

center of the panel. (We also removed the default traffic descriptor All other traffic.) We then pressed **Next**.

We selected a Security Level of Permit for the two types of NSS traffic from the pull-down in the center of the page.

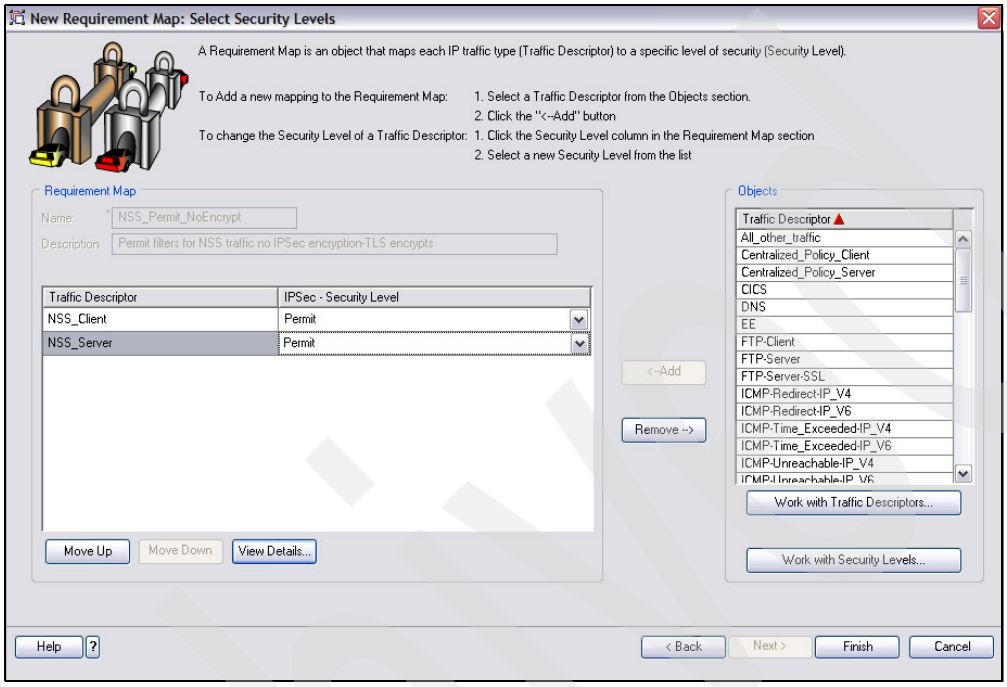


Figure 4-64 Permit but do not encrypt NSS traffic (TLS performs encryption)

On Figure 4-64 we pressed **Finish**, taking us to the panel shown in Figure 4-65.

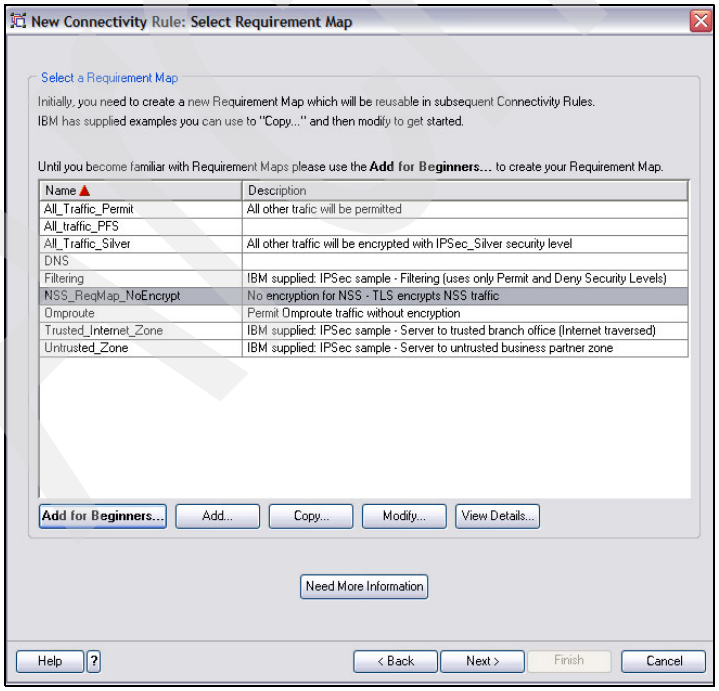


Figure 4-65 Completed Requirement Map for NSS traffic: both client and server

When we press **View Details** we see that we now permit the NSS traffic to and from port 4159 without enforcing encryption, because TLS will be encrypting the NSS flows; see Figure 4-66.

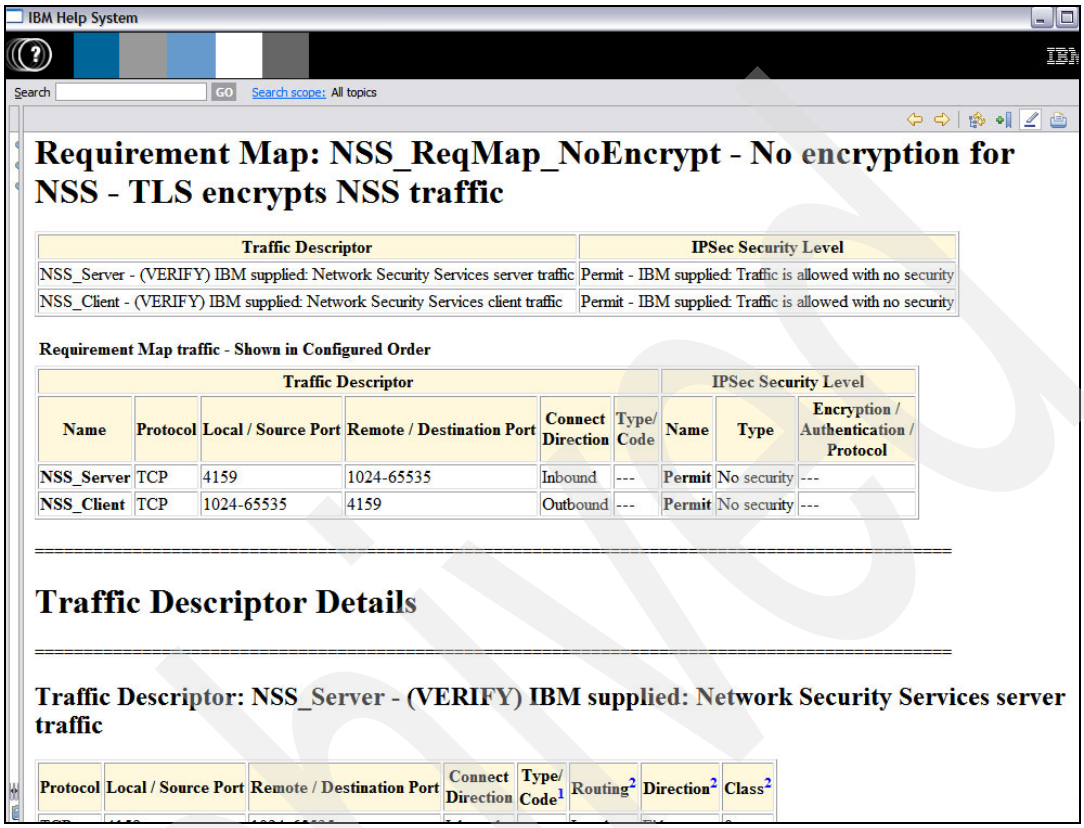


Figure 4-66 Details of NSS requirement map

We close this help panel and press **Next** and **Finish**, which brings us to the IPSec Perspective shown in Figure 4-67 on page 179.

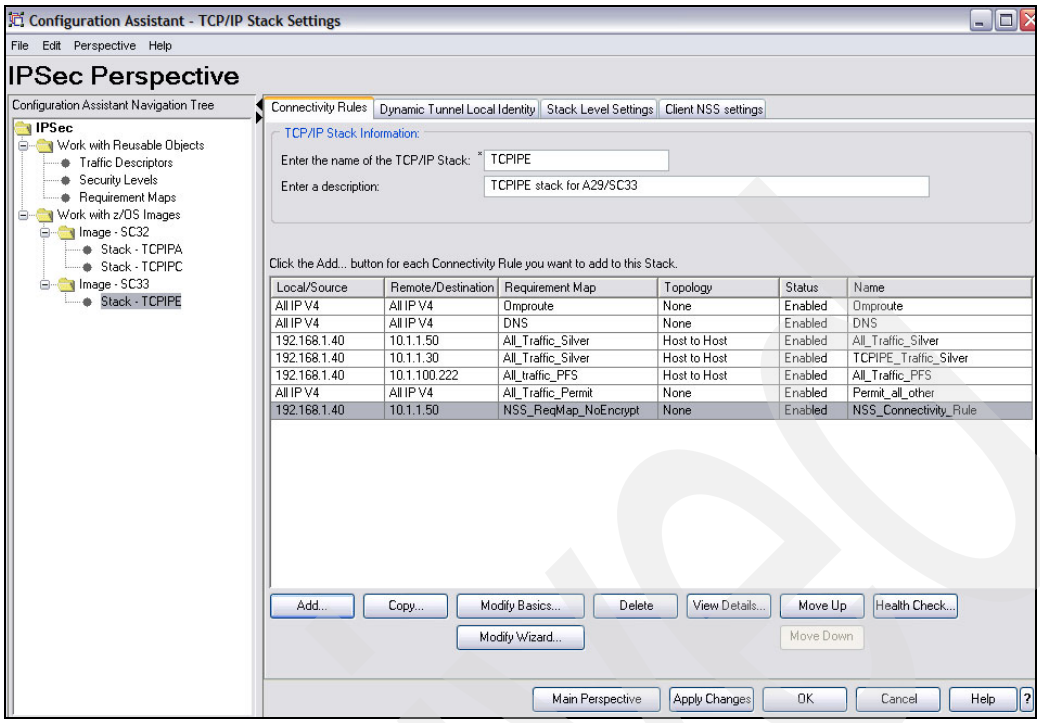


Figure 4-67 IPSec perspective with new Connectivity Rule for NSS

However, notice where the rule has been placed: at the lowest priority of the list. We need to move this rule above the encryption rules so that it is found at a higher priority. So, we highlight the rule and press **Move Up** until the Permit rule resides above the encryption rule with the Requirement Map of All_Traffic_Silver; see Figure 4-68.

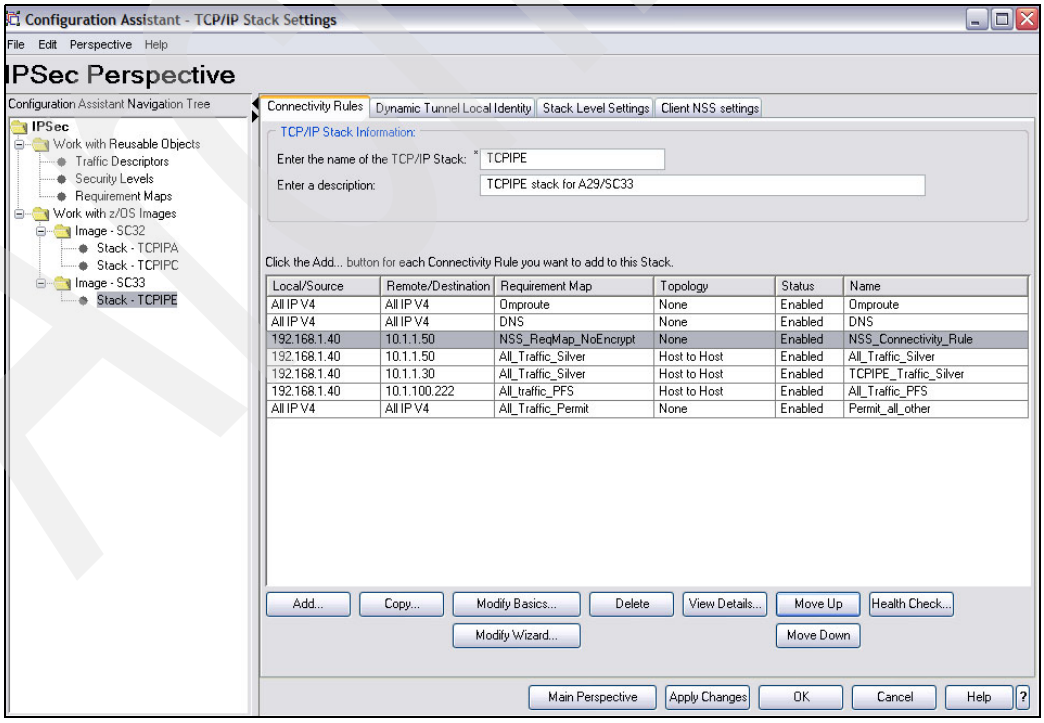


Figure 4-68 TCPIPA: IPSec connectivity rule for NSS

We press **Apply Changes** for the TCPIPE stack, and then select the **SC32 TCPIPA** stack in Figure 4-69, which also needs these new IP connectivity and Requirement Maps.

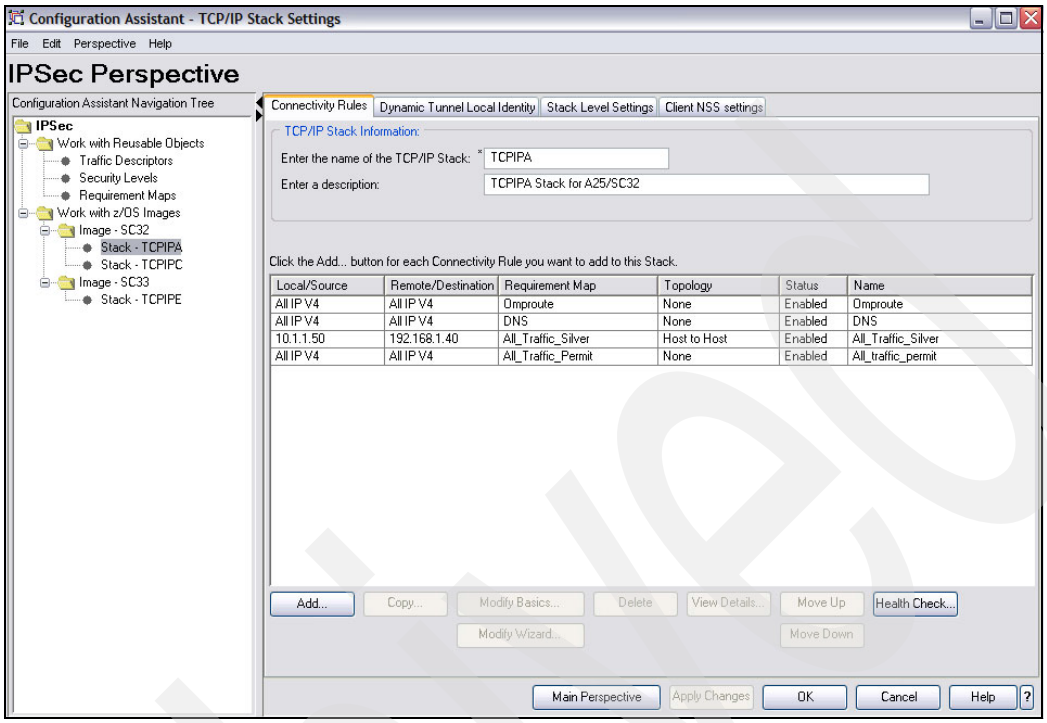


Figure 4-69 New connectivity rule needed for NSS at TCPIPA stack

We press **Add** from Figure 4-69 and, as was done for the TCPIPE stack, choose **Typical** and **No network topology** on the subsequent two panels.

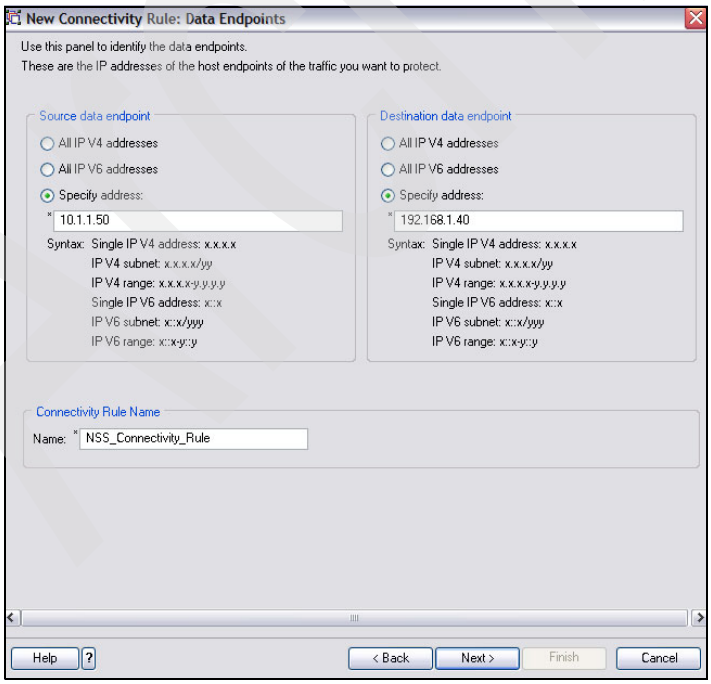


Figure 4-70 Connectivity Rule for Client to Server

We fill out the address and rule name information on Figure 4-70 on page 180 and press **Next**, which presents Figure 4-71.

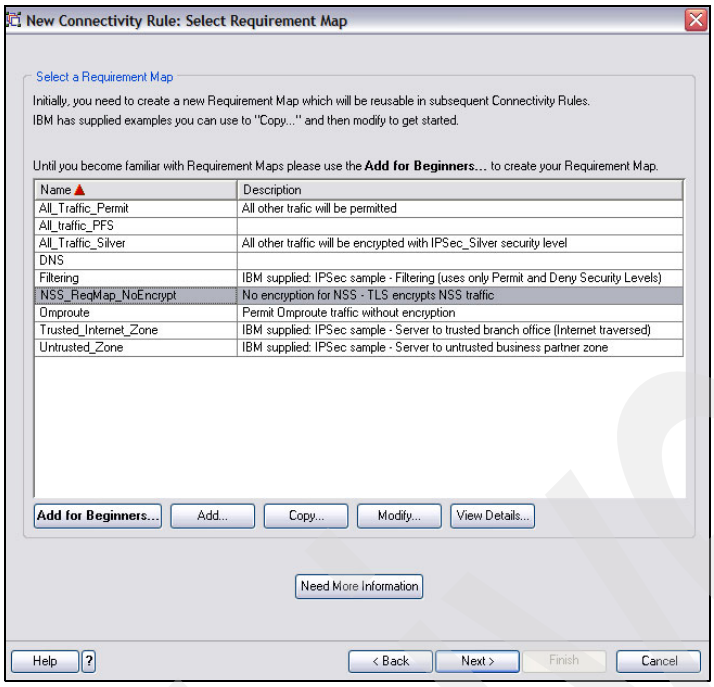


Figure 4-71 Available requirement maps for the NSS connectivity rule

Because we had already created a Requirement Map for NSS, we can now simply select it from this panel and choose **Next**. We press **Finish** at the connectivity rule and see that TCPIPA now has the appropriate rules for NSS, as shown in Figure 4-72.

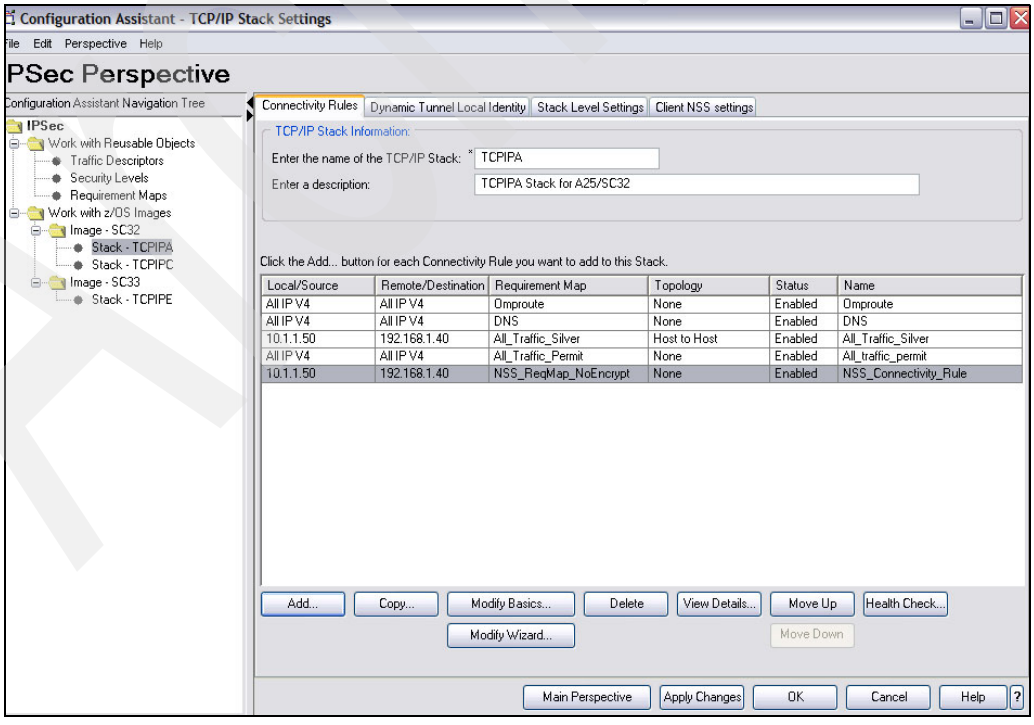


Figure 4-72 TCPIPA: IPSec connectivity rule for NSS

However, notice where the rule has been placed: at the lowest priority of the list. We need to move this rule above the encryption rules so that it is found at a higher priority. So, we highlight the rule and press **Move Up** until the Permit rule resides above the encryption rule with the requirement map of All_Traffic_Silver, as shown in Figure 4-73.

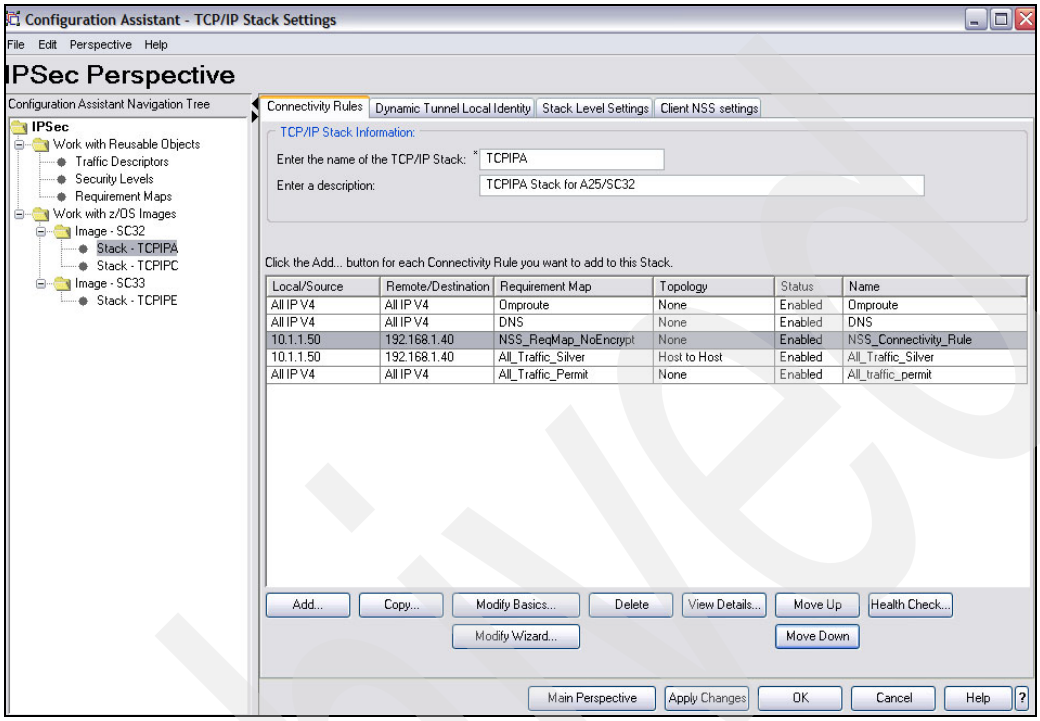


Figure 4-73 Move permit rule

We click **Apply Changes** and then choose **Main Perspective** from Figure 4-73.

With TCPIPA already highlighted in the Main Perspective panel, we see that AT-TLS is `Disabled` for TCPIPA; see Figure 4-74 on page 183.

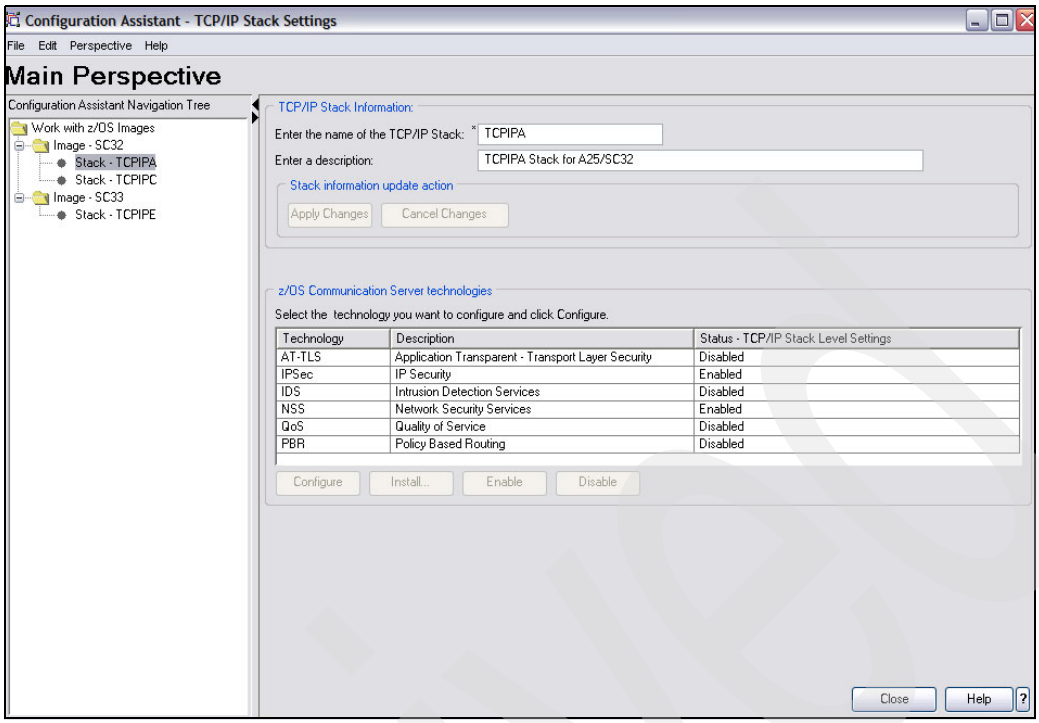


Figure 4-74 No AT-TLS enablement for TCPIPA

In fact, when we selected TCPIPE, we also saw that AT-TLS is not enabled there, either. However, we must have AT-TLS in order for this NSS configuration to work. Therefore, although we have defined the NSS portions of the solution, we are still missing the necessary AT-TLS definitions on both TCPIPA and TCPIPE, so we build those next.

Enabling AT-TLS for NSS

To enable AT-TLS for NSS, select TCPIPE and highlight it, **Enable** AT-TLS, and then press **Configure**. This takes you to a panel where you specify the connectivity rules; see Figure 4-75 on page 184.

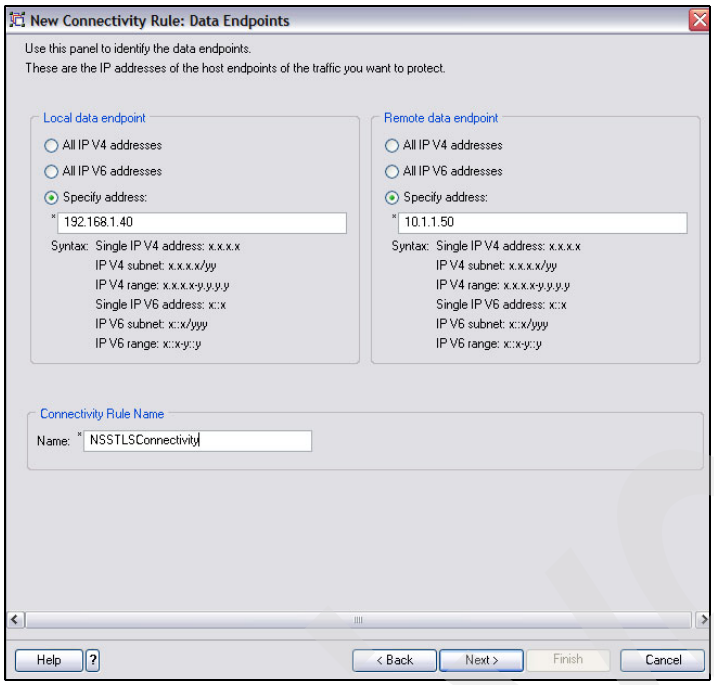


Figure 4-75 Connectivity Rule for AT-TLS and NSS

As you have learned in other chapters in this book, fill in the fields with the addresses of the NSS server (at TCPIPE) and TCPIPA as the NSS client. Then press **Next**, which takes you to the Requirement Map panel, where you select **Add for Beginners...**. This step takes you to Figure 4-76.

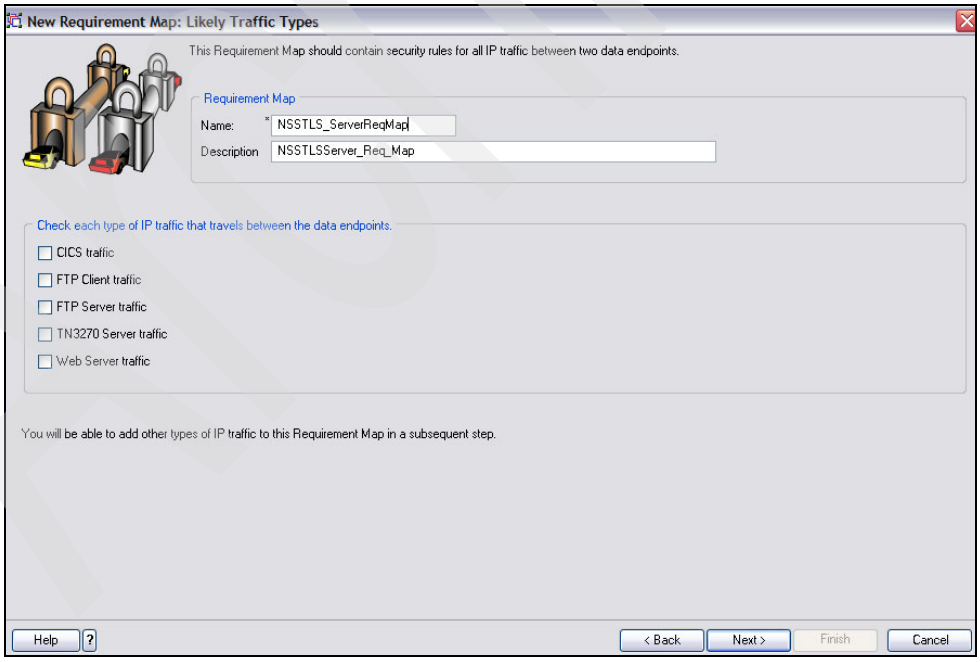


Figure 4-76 New Requirement Map: Traffic Types

Press **Next** and on the subsequent panel choose the NSS server traffic descriptor, as shown in Figure 4-77 on page 185.

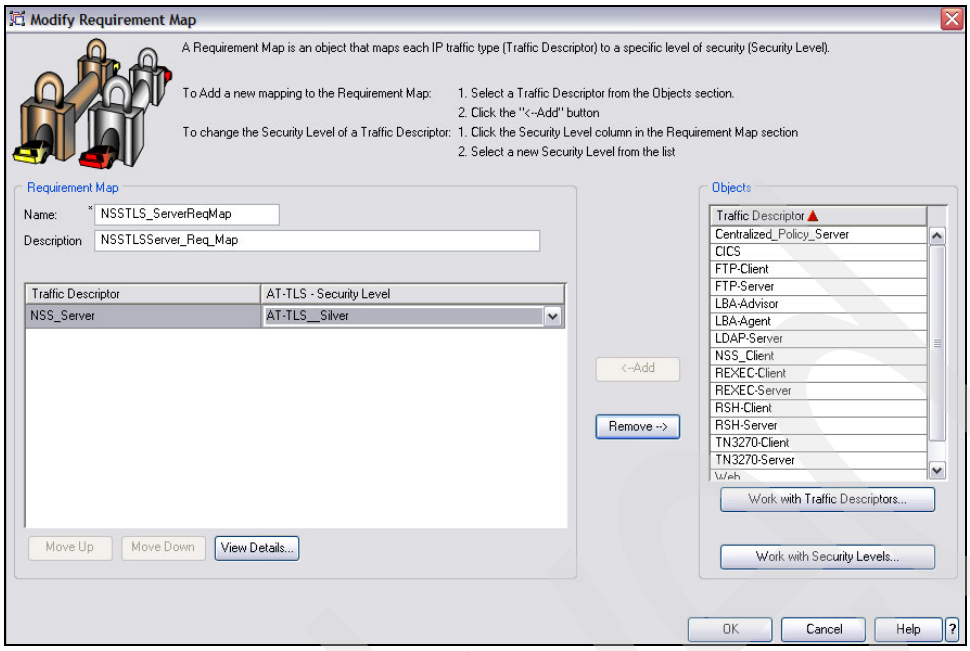


Figure 4-77 NSS Client and Server Traffic Descriptors

Click **Next**. You are given the opportunity to select the Security Levels; in our case, we selected AT-TLS Silver. Continue pressing **Next** and **Finish** until you arrive back at the AT-TLS Perspective panel, which you see in Figure 4-78.

Note: Do not be concerned if you see messages about having only one traffic descriptor in a Requirement Map. You may press **Proceed** when you see such messages. You can always go back and add other traffic descriptors later.

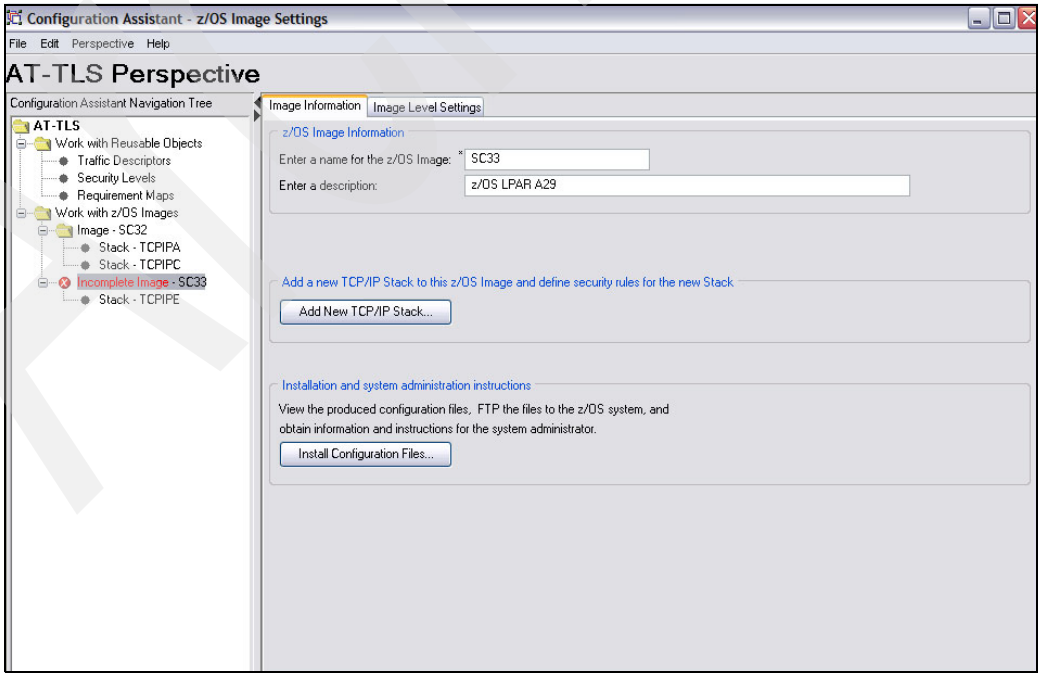


Figure 4-78 AT-TLS Perspective for NSS server at TCPIPE

Notice in Figure 4-78 on page 185 that the SC33 image is still `Incomplete`, from an AT-TLS perspective. Highlight SC33 and select the **Image Level Settings** tab.

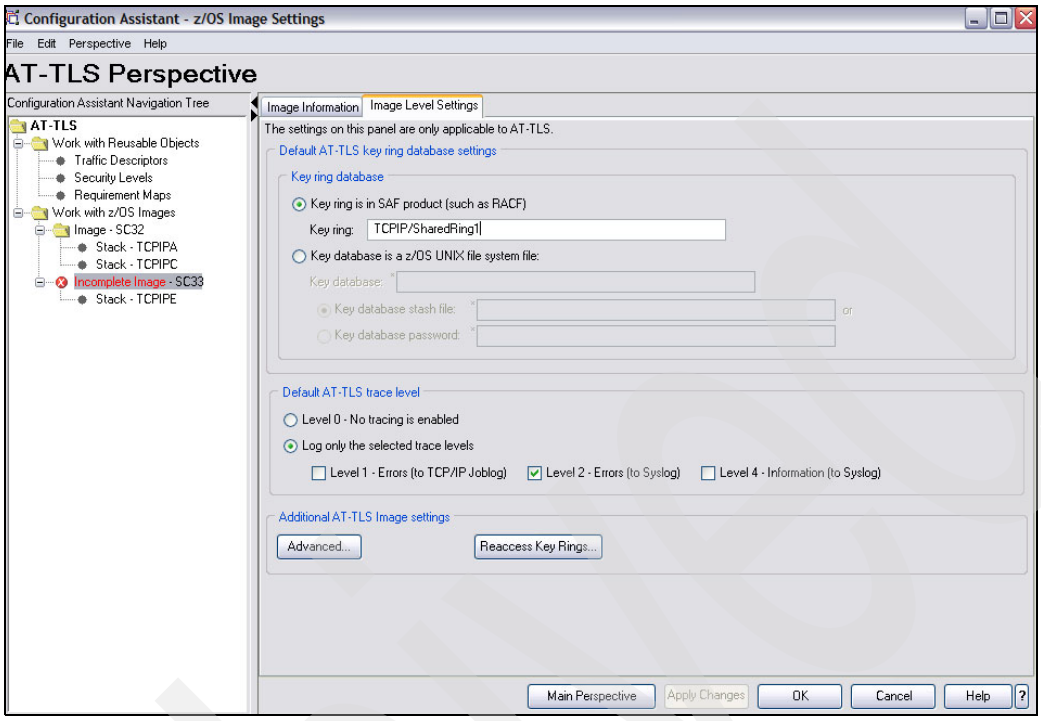


Figure 4-79 The TLS key ring for NSS

On Figure 4-79 you need to specify the correct TLS keyring for the secure communication between the NSS server and client. In our scenario, our server and client share the same TLS keyring. Note that your configuration may have separate TLS key rings: one for the client and one for the server.

You must fill in the appropriate value for the image with which you are working. An informational message appears for us. (Depending on the path you have taken through the GUI, you might receive a different message.)

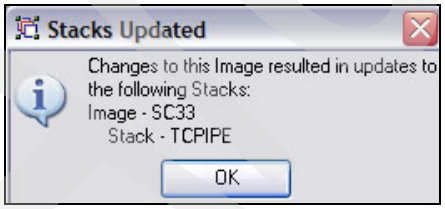


Figure 4-80 Completed AT-TLS configuration for NSS

The SC33 image now shows as complete on the AT-TLS perspective panel in Figure 4-81 on page 187.

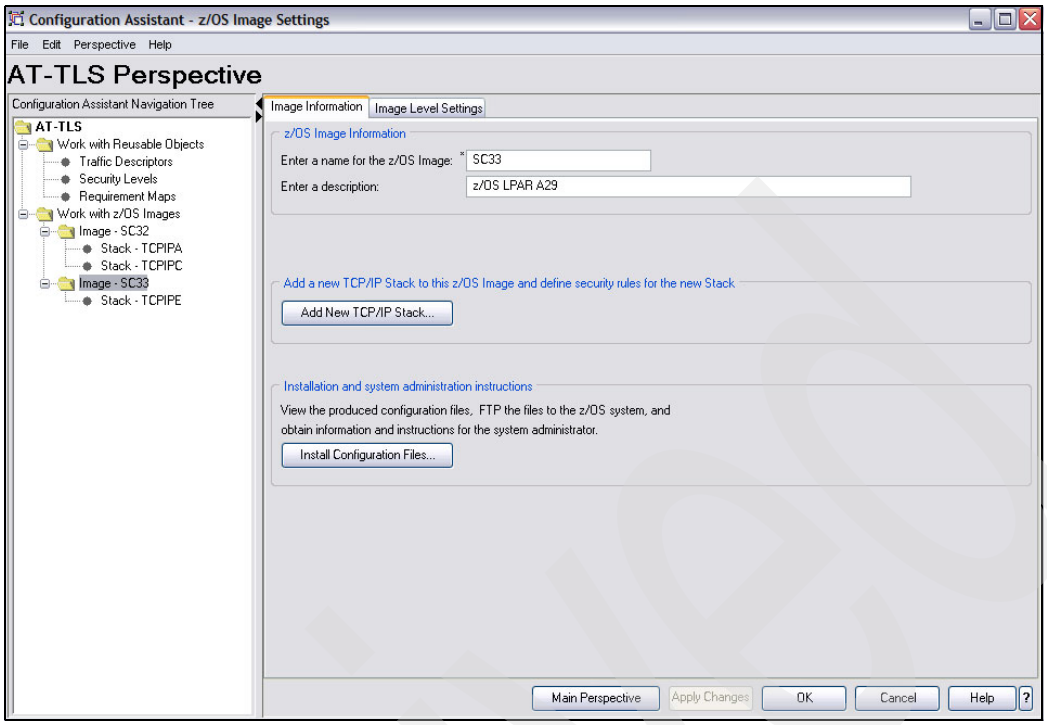


Figure 4-81 Completed AT-TLS perspective for SC33 and TCPIPE

When we select **TCPIPA** from the AT-TLS Perspective, we see that this stack is disabled, as shown in Figure 4-82.

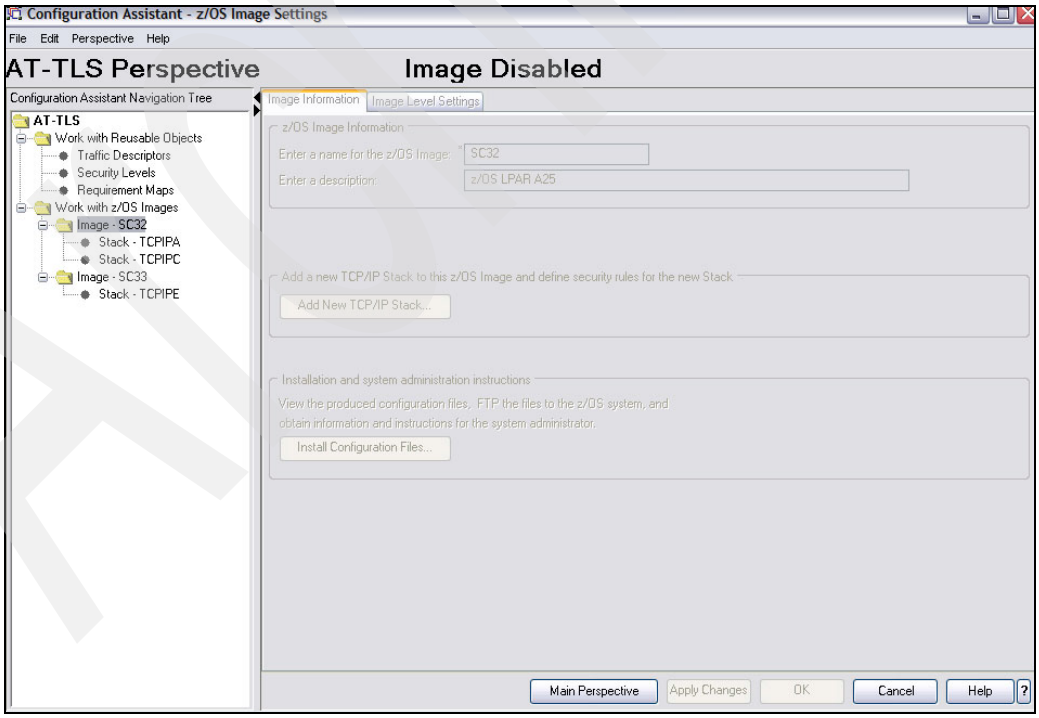


Figure 4-82 SC32 is disabled for AT-TLS

From the Main Perspective, press **Enable** to enable SC32 for AT-TLS. Then select **TCPIPA**, and press **Enable** to enable it for AT-TLS. Where necessary, press **Apply Changes**.

Next, press **Configure** to configure NSS client stack, TCPIPA, for AT-TLS just as you did for the NSS server.

Note: We do not repeat all the steps here to configure AT-TLS for the client. Review the steps you executed for the server and do the same for the NSS client.

An excerpt from the resulting AT-TLS policy file for the NSS server at TCPIPE is shown in “AT-TLS policy for SC33_TCPIPE (NSS Server)” on page 188.

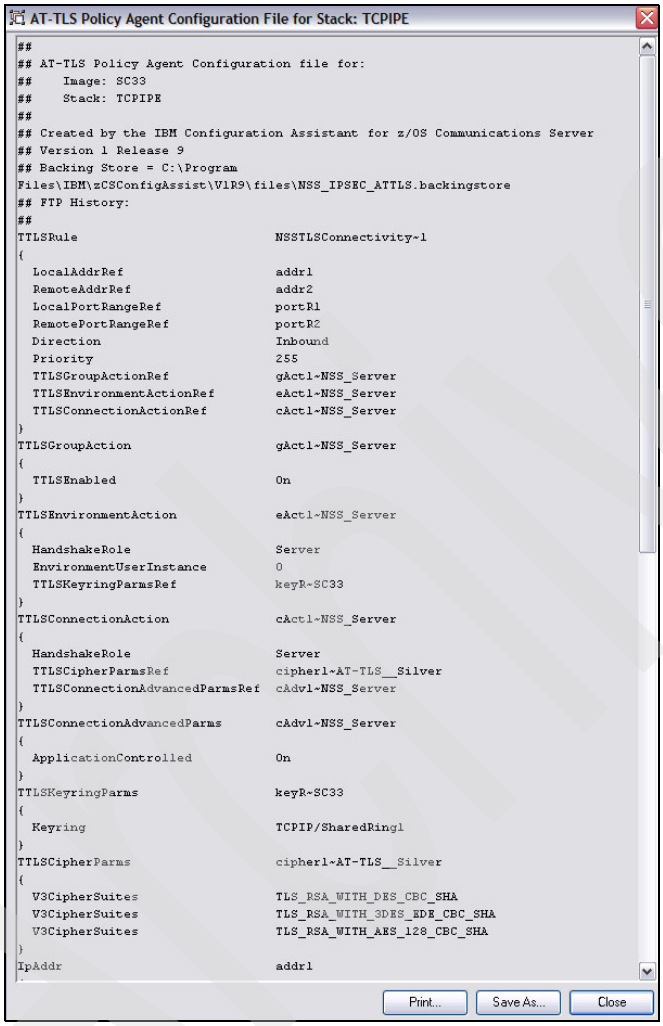


Figure 4-83 AT-TLS policy for SC33_TCPIPE (NSS Server)

The resulting AT-TLS policy for the NSS client is shown in Figure 4-84 on page 189.

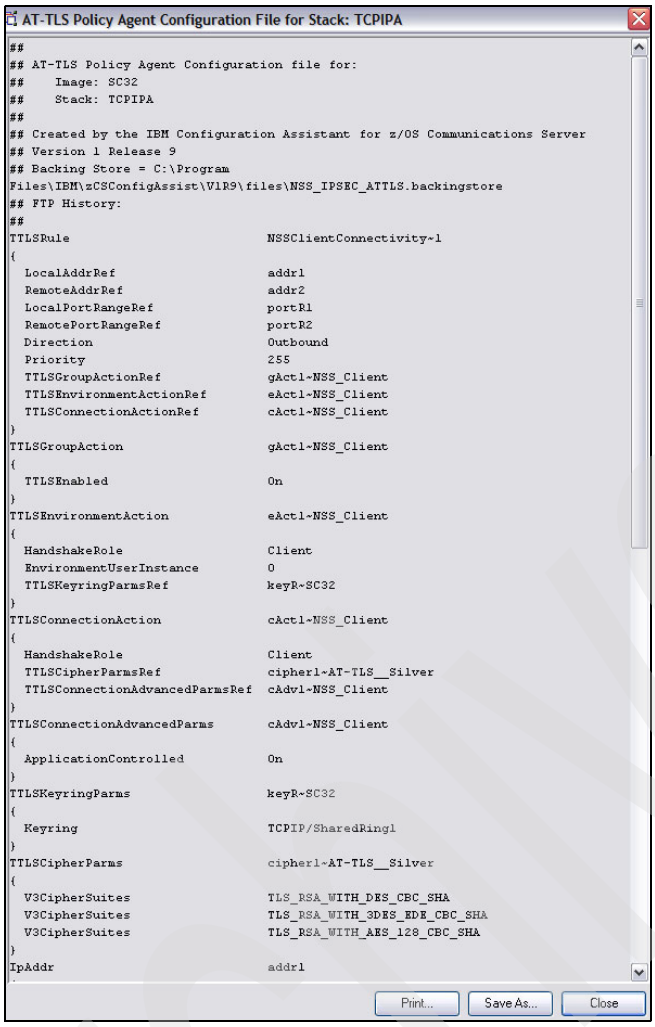


Figure 4-84 AT-TLS policy for NSS client SC32_TCPIPA

Next, we need to see the other files that IBM Configuration Assistant has generated for us.

NSS and IKED configuration files

We move to the Main Perspective and select **Server Image SC33**. Then we select the NSS view and ask to install the files generated. Figure 4-85 on page 190 shows the panel with those filenames for an NSS procedure and for an NSSD configuration file.

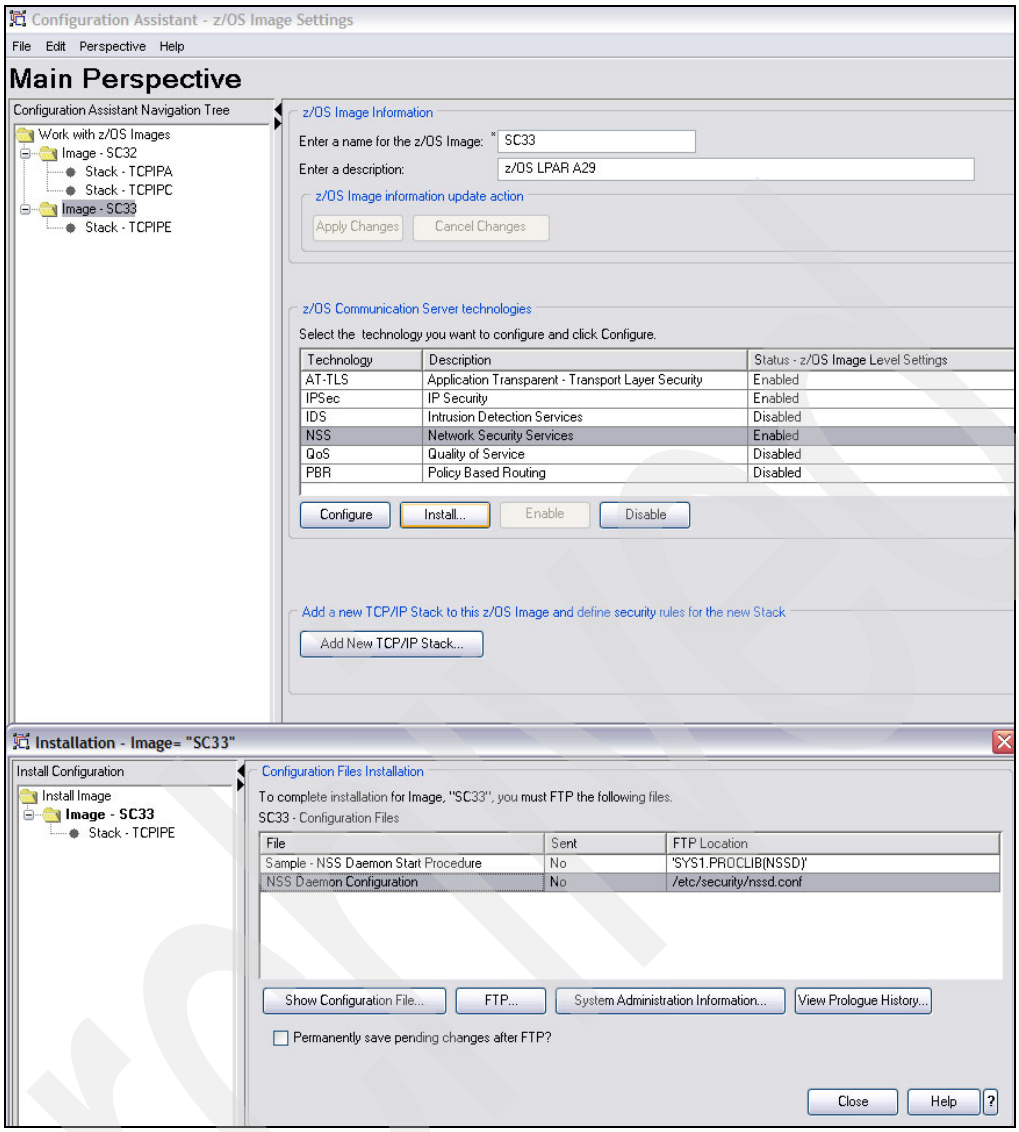


Figure 4-85 NSS server configuration files generated by IBM Configuration Assistant GUI

We press **Show Configuration File** and view the NSS configuration file that contains all the fields already explained in Example 4-45 on page 146. The generated file is shown in Figure 4-86 on page 191.

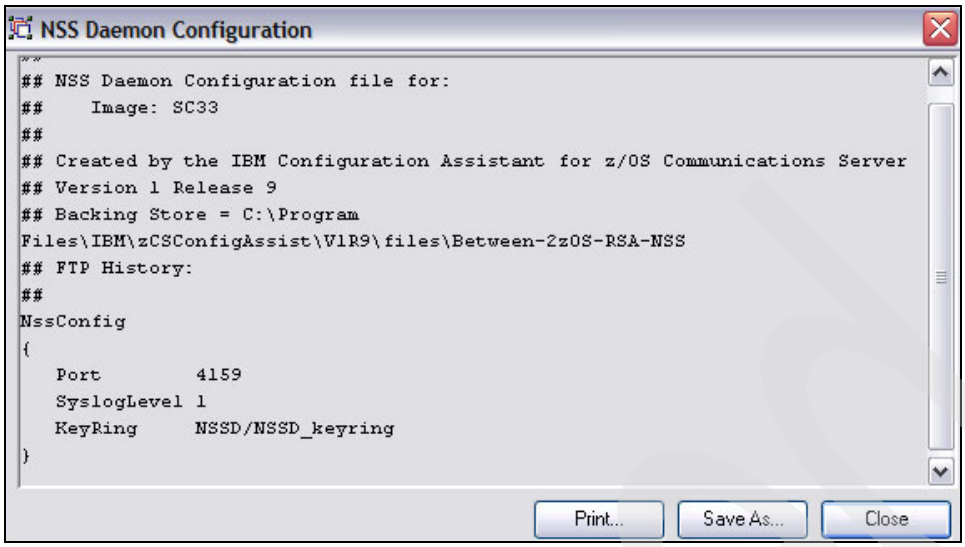


Figure 4-86 NSSD configuration file

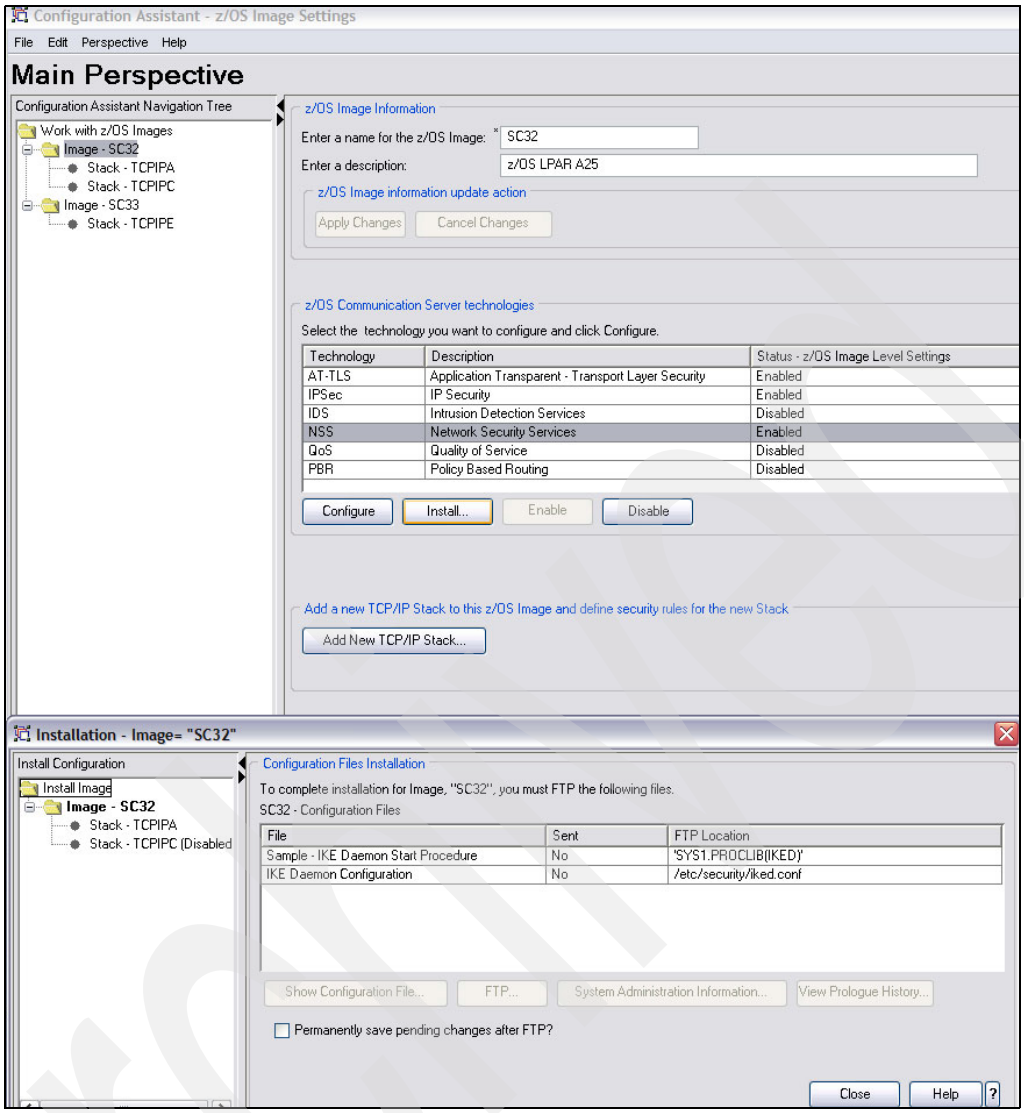


Figure 4-87 From Main Perspective - highlighting NSS and selecting Configure

In Figure 4-87 you see that the Configuration Assistant GUI has generated an IKED start procedure for the NSS client and has also generated an IKE Daemon Configuration file. When we highlight the IKE Daemon Configuration file from this panel and select **Show Configuration File**, we see that the contents in Figure 4-88 on page 193 are just what was shown in Example 4-52 on page 152.

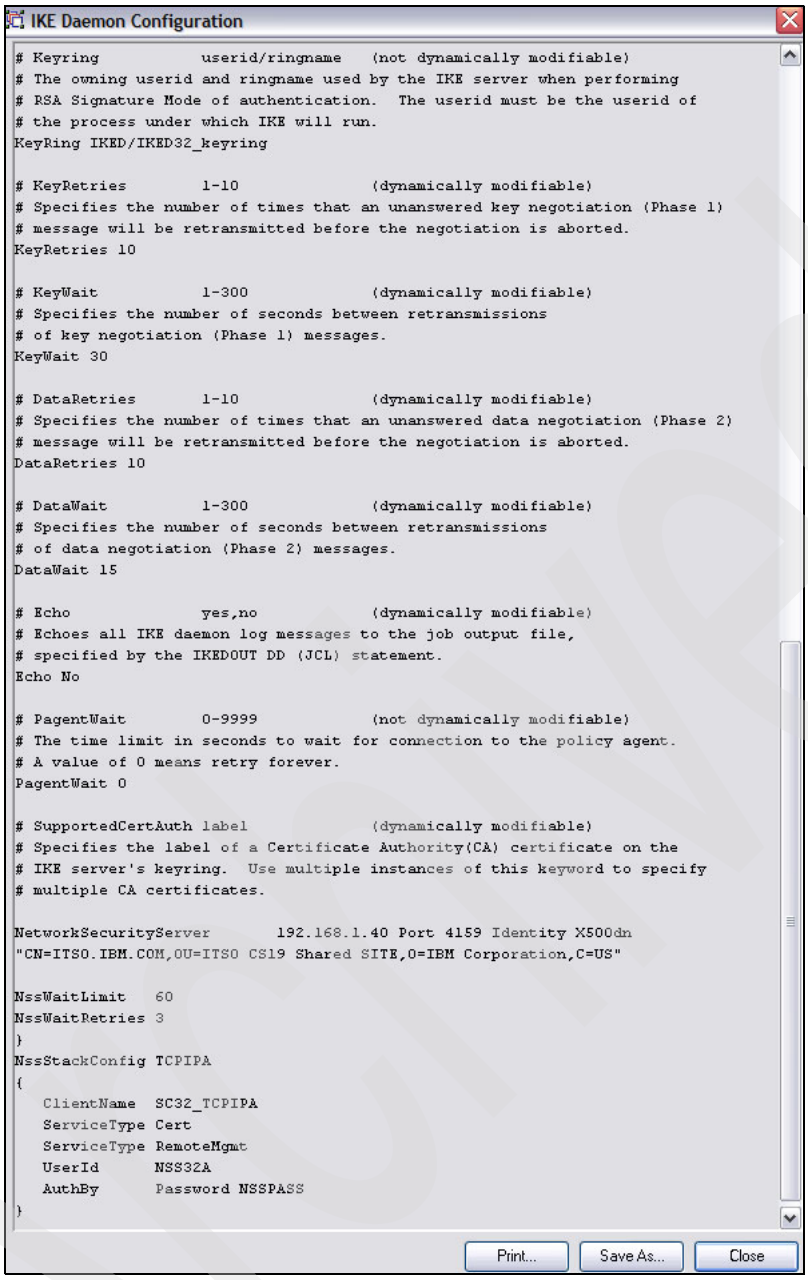


Figure 4-88 IKED configuration file generated by IBM Configuration Assistant

At this point, merge any other backing store files you might need into this configuration. Alternatively, you can take this configuration and merge it into other backing store files, as shown in other parts of this book. Then FTP what you are going to test with to the appropriate nodes, SC32 and SC33.

4.4 Verifying the NSS environment

This section shows the different areas that need to be verified to ensure NSS is running properly.

4.4.1 Make available NSS configuration and policy files

We assume that all necessary syslog daemon and RACF definitions are in place at the server and client nodes. These definitions are described in 4.3.5, “Configuring prerequisites for Network Security Services” on page 135 and 4.3.6, “Configuring authorizations for NSS” on page 140.

The following files must be available at the NSS *server* node:

- NSSD procedure
  - This is described in “Configuring the NSS server” on page 146.
- NSSD configuration file and environment file
  - This is described in “Configuring the NSS server” on page 146.
- AT-TLS policy for NSS client and server communication
  - This is described in “Creating NSS files with IBM Configuration Assistant” on page 156.
  - Reference this policy in the SC33 TCPIPE image file. Our main PAGENT configuration file for SC33 is depicted in Example 4-55.

Example 4-55 Main PAGENT configuration file at SC33: pagent33.conf

```
Loglevel 15
TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH PURGE 600
TcpImage TCPIPE /etc/pagent33_TCPIPE.conf  FLUSH PURGE 600
```

Our TCPIPE image file is depicted in Example 4-56.

Example 4-56 PAGENT Image file for TCPIPE at SC33: pagent33_TCPIPE.conf

```
IPSecConfig  /u/cs02/pagent33_TCPIPE_NSSipsec.conf
TTLSConfig     /u/cs02/TCPIPE_New_NSS_TTLS.policy FLUSH PURGE
```

- IPSec policy file if the NSS server is also enabled for IPSec
  - This is described in “Creating NSS files with IBM Configuration Assistant” on page 156.
  - Reference this policy in the SC33 TCPIPE image file. You see the file referenced in Example 4-56.

The following files must be in place on the NSS *client* node:

- IKED procedure
  - This is described in “Enabling an NSS client to use Network Security Services” on page 151.
- IKED configuration file that indicates the TCP/IP stacks that are to take advantage of Network Security Services
  - This is described in “Enabling an NSS client to use Network Security Services” on page 151. We named our NSS client configuration file iked32RSANSS.conf and placed it in the /etc/security subdirectory of SC32.
- AT-TLS policy for NSS client and server communication
  - This is described in “Creating NSS files with IBM Configuration Assistant” on page 156.
  - Reference this policy in the SC32 TCPIPA image file. Our main PAGENT configuration file for SC32 is depicted in Example 4-57.

Example 4-57 Main PAGENT configuration file at SC32: pagent32.conf

```
LogLevel 15
```

```
TcpImage TCPIPC /etc/pagent32_TCPIPC.conf  FLUSH PURGE 600
TcpImage TCPIPA /etc/pagent32_TCPIPA.conf  FLUSH PURGE 600
```

Our TCPIPA image file is depicted in Example 4-58.

Example 4-58 PAGENT Image file for TCPIPA at SC32: pagent32_TCPIPA.conf

```
IPSecConfig   /u/cs02/pagent32_TCPIPA_NSSipsec.conf
TTLSConfig  /u/cs02/TCPIPA_New_NSS_TTLS.policy    FLUSH PURGE
```

- IPSec policy file for the communication between the NSS client and the NSS server
  - This is described in "Creating NSS files with IBM Configuration Assistant" on page 156.
  - Reference this policy in the SC32 TCPIPA image file. You see the file referenced in Example 4-58.

4.4.2 Initialize NSSD and the NSS client

We had the luxury of being able to take down the TCP/IP stacks on MVS system IDs SC33 and SC32. As a result, we could initialize all procedures from scratch. If this is not possible for you, you will want to execute the MODIFY REFRESH commands for PAGENT and IKED.

NSS server

1. Restart the TCP/IP procedure on SC33:

   ```
   S TCPIPE
   ```

2. Start the Policy Agent to load the new policies into the TCP/IP stacks. From the MVS console, issue the following command:

   ```
   S PAGENT
   ```

3. Start the IKED procedure, because TCPIPE will be the IKE partner to the NSS client named TCPIPA:

   ```
   S IKEDD
   ```

4. Start the NSSD procedure from the MVS console with the following command:

   ```
   S NSSD
   ```

Example 4-59 shows the startup sequence SC33.

Example 4-59 Startup of NSSD at SC33

```
S NSSD
$HASP100 NSSD     ON STCINRDR
IEF695I START NSSD     WITH JOBNAME NSSD     IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 NSSD     STARTED
EZD1359I NETWORK SECURITY SERVICES (NSS) SERVER RELEASE CS V1R9
SERVICE LEVEL CS070402 CREATED ON Apr  2 2007
EZD1357I NETWORK SECURITY SERVICES (NSS) SERVER INITIALIZATION
SEQUENCE HAS BEGUN
IEE252I MEMBER CTINSS00 FOUND IN SYS1.PARMLIB a
EZD1353I NSS SERVER CONFIG PROCESSING COMPLETE USING FILE /etc/securit
y/nssd33.conf
EZD1358I NSS SERVER INITIALIZATION SEQUENCE HAS COMPLETED b
```

a shows that we remembered to move the CTINSS00 member into parmlib; that is, it was found at initialization.

b shows that the NSS server has successfully initialized.

NSS client

1. Restart the TCP/IP procedure on SC32:

   ```
   S TCPIPA
   ```

2. Start the Policy Agent to load the new policies into the TCP/IP stacks. From the MVS console, issue the following command:

   ```
   S PAGENT
   ```

3. Start the IKED procedure, because TCPIPA will be the IKE partner to the NSS client named TCPIPA:

   ```
   S IKEDC
   ```

Example 4-60 shows the startup sequence at SC32.

Example 4-60 Initialization of NSS client: Startup of IKEDC at SC32

```
S IKEDC
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 585
        TO START IKEDC WITH JOBNAME IKEDC.
$HASP100 IKEDC    ON STCINRDR
IEF695I START IKEDC    WITH JOBNAME IKEDC    IS ASSIGNED TO USER
IBMUSER , GROUP SYS1
$HASP373 IKEDC    STARTED
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R9 SERVICE LEVEL CS070402 CREATED ON Apr  3
2007
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/security/ikedN
SSClient.conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPA   IS UP   a
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
EZD1058I IKE STATUS FOR STACK TCPIPC   IS UP b
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPC
EZD1133I IKE STATUS FOR STACK TCPIP    IS ACTIVE WITHOUT IPSECURITY
SUPPORT c
EZD1046I IKE INITIALIZATION COMPLETE
EZD1136I THE IKE DAEMON IS CONNECTED TO THE NSS SERVER AT 192.168.1.40
 PORT 4159 FOR STACK TCPIPA d
```

a shows that TCPIPA has been enabled for IKE.

b shows that TCPIPC has been enabled for IKE.

c shows that TCPIP is not enabled for IP security or for IKE.

d shows that the IKE daemon is using the NSS services on behalf of TCPIPA.

NSS and IKE displays on SC33 and SC32

The first step is to display the configurations on SC33 and SC32.

The configurations for NSS

We display the NSSD server on SC33; see Example 4-61.

Example 4-61 Display of NSSD configuration at SC33

```
F NSSD,DISPLAY
DISPLAY NETWORK SECURITY SERVER CONFIGURATION PARAMETERS:
    PORT        = 4159    a
    SYSLOGLEVEL = 255     (0X00ff) b
```

```
   KEYRING     = "NSSD/NSSD_keyring" c
```

a shows that the NSS server is listening on port 4159.

b shows that we have raised the syslog level for NSS to a very high value so that we can track down any initial problems. The default syslog level from the IBM Configuration Assistant is lower; after we uploaded these files to the MVS systems, we manually altered the syslog levels in the files as we continued testing.

We will need to reduce this syslog level after we are satisfied with our testing, so as not to impose a performance burden on the system.

c shows that the NSSD server is providing IKE client certificates from the named key ring.

Example 4-62 shows the display results of IKEDD on SC33, where the NSSD server resides.

Example 4-62 Display of IKE process at SC33

```
F IKEDD,DISPLAY
EZD1066I IKE MODIFY COMMAND ACCEPTED
DISPLAY IKE CONFIGURATION PARAMETERS:
  VALUES LOADED FROM /etc/security/iked33.conf
  IKESYSLOGLEVEL = 255
  PAGENTSYSLOGLEVEL = 63
  KEYRING = IKED/IKED33_keyring
  KEYRETRIES = 10
  KEYWAIT = 30
  DATARETRIES = 10
  DATAWAIT = 15
  ECHO = NO
  PAGENTWAIT = 0
  NSSWAITLIMIT = 60
  NSSWAITRETRIES = 3
  IKE CONFIGURATION CONTAINS NO SUPPORTEDCERTAUTH LABELS.
  IKE CONFIGURATION CONTAINS NO NETWORKSECURITYSERVER INFO.
  IKE CONFIGURATION CONTAINS NO NETWORKSECURITYSERVERBACKUP INFO.
  IKE CONFIGURATION CONTAINS NO NSSSTACKCONFIG DEFINITIONS. a
```

a shows that the IKEDD procedure on SC33 is itself not an NSS client.

Even though we have IKE running at SC33, IKE itself is not a prerequisite for NSSD on the server node.

To examine NSS from the client perspective, on SC32 we execute the following command; see the resulting display in Example 4-63.

```
   F IKEDC,DISPLAY
```

Example 4-63 Displaying configuration of NSS client: IKEDC on behalf of TCPIPA

```
F IKEDC,DISPLAY
EZD1066I IKE MODIFY COMMAND ACCEPTED
DISPLAY IKE CONFIGURATION PARAMETERS:
  VALUES LOADED FROM /etc/security/ikedNSSClient.conf
  IKESYSLOGLEVEL = 255 a
  PAGENTSYSLOGLEVEL = 255 a
  KEYRING = IKED/IKED32_keyring b
  KEYRETRIES = 10
  KEYWAIT = 30
  DATARETRIES = 10
  DATAWAIT = 15
  ECHO = NO
  PAGENTWAIT = 0
```

```
  NSSWAITLIMIT = 60
  NSSWAITRETRIES = 3
  IKE CONFIGURATION CONTAINS NO SUPPORTEDCERTAUTH LABELS.
  NETWORKSECURITYSERVER   = 192.168.1.40 c
                PORT     = 4159 d
                IDENTITY = CN=ITSO.IBM.COM,OU=ITSO CS19 Shared SITE,
O=IBM Corporation,C=US e
  IKE CONFIGURATION CONTAINS NO NETWORKSECURITYSERVERBACKUP INFO.
  NSSSTACKCONFIG 1: STACK = TCPIPA f
                   CLIENT NAME = SC32_TCPIPA
                    USERID NAME = NSS32A
                    AUTHBY = Password
                    CERT SERVICE: Enabled
                    REMOTEMGMTSERVICE: Enabled
```

a shows the high logging levels which we edited into the files after we transferred them from the IBM Configuration Assistant GUI.

b shows the key ring that the NSS client is using.

c shows the IP address of the NSS server.

d shows the port number that the NSS client must connect to for communication with the NSS server.

e shows the Identity value that is used to find a TLS certificate that provides server authentication. (Remember that we used the x500dn value as the identity.)

f shows the NSS configuration parameters that we coded in the IKE configuration for TCPIPA. You recognize here the symbolic name, the user ID, the authorization method, and that TCPIPA is using both Certificate Management services and Remote Management services.

Performance: Ensure that after your testing is completed, you reduce the logging levels on PAGENT, on IKE, and on the NSS server.

Next, we want to see if the appropriate connections have really been established at the client and server nodes.

AT-TLS connection

Do we really have AT-TLS connections between client and server? We use the **netstat** command to determine the connection status; see Example 4-64.

Example 4-64 TLS at SC33, the NSS server node

```
D TCPIP,TCPIPE,N,TTLS
EZD0101I NETSTAT CS V1R9 TCPIPE 096
TTLSGRPACTION                          GROUP ID          CONNS
GACT1~NSS_SERVER                       00000003              1 a
NSS~TLS~ON                             00000004              0
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

a shows that we have one connection using the AT-TLS action for the NSS server.

We display the connections at the server with the **netstat** command; see Example 4-65.

Example 4-65 Netstat connection at the server site for client NSSD

```
D TCPIP,TCPIPE,N,CONN,CLIENT=NSSD
EZD0101I NETSTAT CS V1R9 TCPIPE 102
```

```
USER ID  CONN     STATE
NSSD     00004095 ESTBLSH      a
  LOCAL SOCKET:   ::FFFF:192.168.1.40..4159
  FOREIGN SOCKET: ::FFFF:10.1.1.50..10029
NSSD     00004092 LISTEN
  LOCAL SOCKET:   ::..4159
  FOREIGN SOCKET: ::..0
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

a shows that we have established a connection (# 4095) with 10.1.1.50 from our listening port of 4159. Is this really a TLS connection? A more specific display tells us the answer; see Example 4-66.

Example 4-66 TLS connection detail for server

```
D TCPIP,TCPIPE,N,TTLS,CONN=4095,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPE 105
CONNID: 00004095
  JOBNAME:      NSSD
  LOCALSOCKET:  ::FFFF:192.168.1.40..4159
  REMOTESOCKET: ::FFFF:10.1.1.50..10029
  SECLEVEL:     TLS VERSION 1 a
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA b
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: NSSSERVERRULEAT-TLS~1
  PRIORITY:       255
  LOCALADDR:      0.0.0.0/0
  LOCALPORT:      4159
  REMOTEADDR:     0.0.0.0/0
  REMOTEPORTFROM: 1024              REMOTEPORTTO: 65535
  DIRECTION:      INBOUND
  TTLSGRPACTION:  GACT1~NSS_SERVER
    GROUPID:                    00000003
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      255
    TRACE:                      255
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
  TTLSENVACTION:  EACT1~NSS_SERVER
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1 c
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~NSS_SERVER d
    HANDSHAKEROLE:              SERVER
    V3CIPHERSUITES:             09 TLS_RSA_WITH_DES_CBC_SHA
                                0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
    TRACE:                      255
    APPLICATIONCONTROLLED:      ON
    CERTIFICATELABEL:           CS19 ITSO SHAREDSITE1
```

```
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

a shows that the connection was established with TLS V1 security.

b shows the negotiated cipher suite of DES.

c shows the key ring used for client authentication.

d shows the sequence of the cipher suites presented by the server. The server sequence of ciphers is the sequence that prevails. We see later that the client also presented the same sequence with DES.

Similar information is obtained from the **pasearch -t** command output, as shown in Example 4-67.

Example 4-67 pasearch -t at the NSS server

```
TCP/IP pasearch CS V1R9                  Image Name: TCPIPE
  Date:                 01/08/2008       Time:  20:37:31
  TTLS Instance Id:     1199827999

policyRule:             NSSServerRuleAt-TLS~1
  Rule Type:            TTLS
  Version:              3                Status:           Active
  Weight:               255              ForLoadDist:      False
  Priority:             255              Sequence Actions: Don't Care
  No. Policy Action:    3
.................
LocalPortFrom:       4159            LocalPortTo:       4159
   RemotePortFrom:     1024             RemotePortTo:     65535
   JobName:                             UserId:
   ServiceDirection:   Inbound
   ServiceDirection:   Inbound

TTLS Action:                    eAct1~NSS_Server
    Version:                     3
    Status:                      Active
    Scope:                       Environment
    HandshakeRole:               Server
    TTLSKeyringParms:
     Keyring:                    TCPIP/SharedRing1   a
    TTLSEnvironmentAdvancedParms:
    SSLv2:                      Off
    SSLv3:                      On
    TLSv1:                      On
    ApplicationControlled:      Off
    HandshakeTimeout:           10
    ClientAuthType:             Required
    ResetCipherTimer:           0
   EnvironmentUserInstance:     0

 TTLS Action:                   cAct1~NSS_Server
   Version:                     3
   Status:                      Active
   Scope:                       Connection
   HandshakeRole:               Server
   Trace:                       255
   TTLSConnectionAdvancedParms:
    ApplicationControlled:      On
```

```
    CertificateLabel:          CS19 ITSO SharedSite1 b
   TTLSCipherParms:
    v3CipherSuites:
     09  TLS_RSA_WITH_DES_CBC_SHA
     0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
     2F  TLS_RSA_WITH_AES_128_CBC_SHA
.....
```

The output from **pasearch** displays the keyrings a and certificate labels b that you may be having problems with because of faulty RACF definitions and configuration files.

At the client node, we also display the **netstat** output; see Example 4-68.

Example 4-68 TTLS activity at TCPIPA

```
D TCPIP,TCPIPA,N,TTLS
 EZD0101I NETSTAT CS V1R9 TCPIPA 497
 TTLSGRPACTION                              GROUP ID            CONNS
 GACT1~NSS_CLIENT                           00000003                1 a
 GACT1~NSS_CLIENT                 (STALE)   00000001                3 b
 NSS~TLS~ON                                 00000004                0
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

a shows that we have one connection using the AT-TLS action for the NSS client.

b shows that we have old, stale connections that are remembered from previous attempts. (You might not always see stale entries, of course.)

We want to examine this connection in more detail, and we therefore issue a generic **netstat** command to view connections; see Example 4-69.

Example 4-69 Connections at TCPIPA for IKEDC

```
D TCPIP,TCPIPA,N,CONN,CLIENT=IKEDC
EZD0101I NETSTAT CS V1R9 TCPIPA 506
USER ID  CONN     STATE
IKEDC    0002063A ESTBLSH      a
  LOCAL SOCKET:   10.1.1.50..10029
  FOREIGN SOCKET: 192.168.1.40..4159
IKEDC    00020632 UDP
  LOCAL SOCKET:   ::..58663
  FOREIGN SOCKET: *..*
IKEDC    00020634 UDP
  LOCAL SOCKET:   0.0.0.0..4500
  FOREIGN SOCKET: *..*
IKEDC    00020633 UDP
  LOCAL SOCKET:   0.0.0.0..500
  FOREIGN SOCKET: *..*
4 OF 4 RECORDS DISPLAYED
END OF THE REPORT
```

a shows that we have established a connection (# 2063A) with 192.168.1.40 and its port 4159, the NSS listening port. Is this really a TLS connection? A more specific display tells us the answer; see Example 4-70.

Example 4-70 The NSS TLS connection

```
D TCPIP,TCPIPA,N,TTLS,CONN=2063A,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPA 518
```

```
CONNID: 0002063A
  JOBNAME:      IKEDC
  LOCALSOCKET:  10.1.1.50..10029
  REMOTESOCKET: 192.168.1.40..4159
  SECLEVEL:     TLS VERSION 1    a
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA b
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: ATTLSNSS4CLIENT~1
  PRIORITY:       255
  LOCALADDR:      0.0.0.0/0
  LOCALPORTFROM:  1024              LOCALPORTTO:  65535
  REMOTEADDR:     0.0.0.0/0
  REMOTEPORT:     4159
  DIRECTION:      OUTBOUND
  TTLSGRPACTION:  GACT1~NSS_CLIENT
    GROUPID:                    00000003
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    CTRACECLEARTEXT:            OFF
    TRACE:                      255
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
  TTLSENVACTION:  EACT1~NSS_CLIENT
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              CLIENT
    KEYRING:                    TCPIP/SHAREDRING1 c
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~NSS_CLIENT
    HANDSHAKEROLE:              CLIENT
    V3CIPHERSUITES: d           09 TLS_RSA_WITH_DES_CBC_SHA
                                0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
    TRACE:                      255
    APPLICATIONCONTROLLED:      ON
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

a shows that the connection was established with TLS V1 security.

b shows the negotiated cipher suite of DES.

c shows the key ring used for client authentication.

d shows the sequence of the cipher suites presented by the client. The server sequence of ciphers is the sequence that prevails. We saw earlier that the server also presented this sequence with DES first. Because both sides presented DES and the server considered it its preferred cipher suite, that is the suite that won out.

IPSec displays for NSS client and NSS server

Example 4-71 Display known NSS clients from perspective of NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display      a
```

```
CS V1R9 ipsec  NSS Client Name: n/a  Tue Jan  8 17:50:58 2008
Primary:  NSS Server      Function: Display           Format:   Detail
Source:   Server          Scope:    n/a               TotAvail: 1
SystemName: SC33

ClientName:                     SC32_TCPIPA
StackName:                      TCPIPA
SystemName:                     SC32
ClientIPAddress:                ::ffff:10.1.1.50
ClientPort:                     10029
ServerIPAddress:                ::ffff:192.168.1.40
ServerPort:                     4159
UserID:                         NSS32A
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
ConnectState:                   connected
TimeConnected:                  2008/01/08 16:33:53
TimeOfLastMessageFromClient:    2008/01/08 16:33:53
```

a shows the command that reveals all NSS clients. We have executed this command from the NSS server site.

Notice all the information that is known about the TLS connection between client and server. You can display the same information in a briefer form by using the **-r short** option; see Example 4-72.

Example 4-72 NSS client short display at NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display -r short

CS V1R9 ipsec  NSS Client Name: n/a  Tue Jan  8 17:55:37 2008
Primary:  NSS Server      Function: Display           Format:   Short
Source:   Server          Scope:    n/a               TotAvail: 1
SystemName: SC33

ClientName              |StackName|SystemName
                        |ClientIPAddress|ClientPort|ServerIPAddress|ServerPort
                        |UserID
                        |RemoteManagementSelected|RemoteManagementEnabled
                        |CertificateServicesSelected|CertificateServicesEnabled
                        |ConnectState
                        |TimeConnected|TimeOfLastMessageFromClient

SC32_TCPIPA             |TCPIPA|SC32
                        |::ffff:10.1.1.50|10029|::ffff:192.168.1.40|4159
                        |NSS32A
                        |Yes|Yes
                        |Yes|Yes
                        |connected
                        |2008/01/08 16:33:53|2008/01/08 16:33:53


1 entries selected
```

Another way to view the same information is by using the **-r wide** option; see Example 4-73 on page 204.

Example 4-73 Wide option for ipsec command from NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display -r wide

CS V1R9 ipsec  NSS Client Name: n/a  Tue Jan  8 17:59:08 2008
Primary:  NSS Server      Function: Display           Format:   Wide
Source:   Server          Scope:    n/a               TotAvail: 1
SystemName: SC33

ClientName|StackName|SystemName|ClientIPAddress|ClientPort|ServerIPAddress|Serve
rPort|UserID|RemoteManagementSelected|RemoteManagementEnabled|CertificateService
sSelected|CertificateServicesEnabled|ConnectState|TimeConnected|TimeOfLastMessag
eFromClient

SC32_TCPIPA|TCPIPA|SC32|::ffff:10.1.1.50|10029|::ffff:192.168.1.40|4159|NSS32A|Y
es|Yes|Yes|Yes|connected|2008/01/08 16:33:53|2008/01/08 16:33:53


1 entries selected
```

If you want to isolate a view of a single NSS client from the server perspective, use the -z <client_symbolic_name> option, as in Example 4-74.

Example 4-74 Information about a single NSS client displayed at NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display -z SC32_TCPIPA

CS V1R9 ipsec  NSS Client Name: n/a  Tue Jan  8 18:02:24 2008
Primary:  NSS Server      Function: Display           Format:   Detail
Source:   Server          Scope:    n/a               TotAvail: 1
SystemName: SC33

ClientName:                     SC32_TCPIPA
StackName:                      TCPIPA
SystemName:                     SC32
ClientIPAddress:                ::ffff:10.1.1.50
ClientPort:                     10029
ServerIPAddress:                ::ffff:192.168.1.40
ServerPort:                     4159
UserID:                         NSS32A
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
ConnectState:                   connected
TimeConnected:                  2008/01/08 16:33:53
TimeOfLastMessageFromClient:    2008/01/08 16:33:53
************************************************************************
```

Now we move to the client system (SC32), and issue some ipsec displays there.

Example 4-75 Display at NSS client: only one of three stacks is NSS-enabled

```
CSO2 @ SC32:/u/csO2>ipsec -w display

CS V1R9 ipsec  NSS Client Name: n/a  Tue Jan  8 18:04:30 2008
Primary:  Stack NSS       Function: Display           Format:   Detail
Source:   IKED            Scope:    n/a               TotAvail: 3
SystemName: SC32
```

```
StackName:                      TCPIP
ClientName:                     n/a
NSServicesSupported:            No
RemoteManagementSelected:       No
RemoteManagementEnabled:        n/a
CertificateServicesSelected:    No
CertificateServicesEnabled:     n/a
NSClientIPAddress:              n/a
NSClientPort:                   n/a
NSServerIPAddress:              n/a
NSServerPort:                   n/a
NSServerSystemName:             n/a
UserID:                         n/a
ConnectionState:                n/a
TimeConnectedToNSServer:        n/a
TimeOfLastMessageToNSServer:    n/a
************************************************************************
StackName:                      TCPIPA          a
ClientName:                     SC32_TCPIPA
NSServicesSupported:            Yes
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
NSClientIPAddress:              10.1.1.50
NSClientPort:                   10029
NSServerIPAddress:              192.168.1.40
NSServerPort:                   4159
NSServerSystemName:             SC33
UserID:                         NSS32A
ConnectionState:                connected
TimeConnectedToNSServer:        2008/01/08 16:33:53
TimeOfLastMessageToNSServer:    2008/01/08 16:33:53
************************************************************************
StackName:                      TCPIPC
ClientName:                     n/a
NSServicesSupported:            No
RemoteManagementSelected:       No
RemoteManagementEnabled:        n/a
CertificateServicesSelected:    No
CertificateServicesEnabled:     n/a
NSClientIPAddress:              n/a
NSClientPort:                   n/a
NSServerIPAddress:              n/a
NSServerPort:                   n/a
NSServerSystemName:             n/a
UserID:                         n/a
ConnectionState:                n/a
TimeConnectedToNSServer:        n/a
TimeOfLastMessageToNSServer:    n/a
************************************************************************

3 entries selected
```

a shows that TCPIPA is the only NSS-enabled stack on SC32.

At this point, you have seen evidence of the TLS connection between the NSS server and client. You have not seen evidence, however, that any connections using IPSec were established with TLS services. That is what you want to look at next.

IPSec tunnels with NSS services

The last part of our verification of NSS services requires that we establish IPSec tunnels and confirm that they have used the NSS server.

Our first display is from the client side, as shown in Example 4-76.

Example 4-76 Are there any IPSEC tunnels with endpoints on SC32?

```
CSO2 @ SC32:/u/csO2>ipsec -p TCPIPA -y display

CS V1R9 ipsec  Stack Name: TCPIPA  Tue Jan  8 18:12:05 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 0


0 entries selected

CSO2 @ SC32:/u/csO2>ipsec -p TCPIPC -y display

CS V1R9 ipsec  Stack Name: TCPIPC  Tue Jan  8 18:12:23 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 0


0 entries selected
```

Example 4-76 shows that both the TCPIPA and the TCPIPC stack do not have a connection established with IP security.

In the next display (Figure 4-89 on page 207), you see output that includes an IKE tunnel between TCPIPC and TCPIPE (using IKE in pre-shared key mode), and also an IKE tunnel established between TCPIPA and TCPIPE (using NSS certificate services for TCPIPA). The IKE tunnel between TCPIPA and TCPIPE used RSA signature authentication.

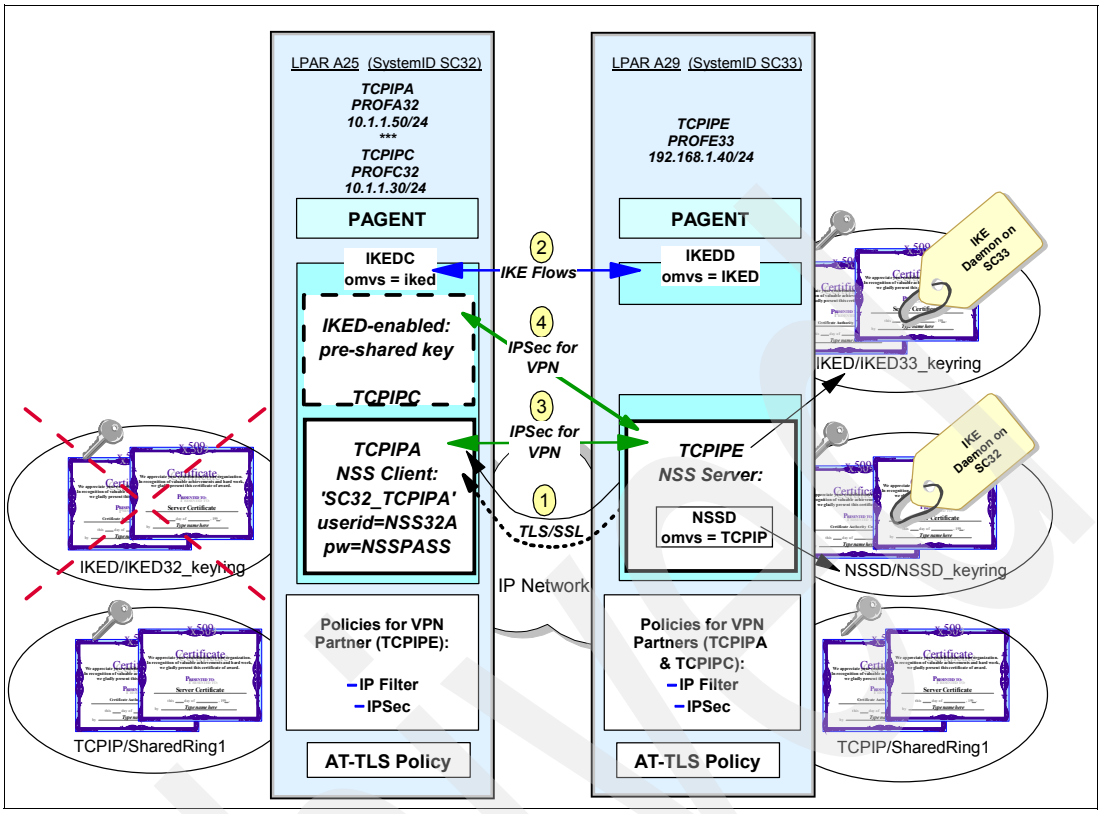


Figure 4-89 TCPIPA is NSS-enabled and TCPIPC is not NSS-enabled

In Figure 4-89, we use IKEDC on SC32 to build an IPSec tunnel between TCPIPA and TCPIPE on SC33. The key ring for TCPIPE is named IKED/IKED33_keyring. The key ring for TCPIPA's certificate management is the NSSD/NSSD_keyring. The certificates required for TCPIPA's certificate management were moved (or copied) from TCPIPA's former IKED32_keyring over to the NSSD key ring.

In Figure 4-89, we also add an IPSec tunnel between TCPIPC and TCPIPE SC33.

TCPIPC is not NSS-enabled. It continues to use the pre-shared key mode of IKE when it authenticates the IKE partners.

Note: Due to time constraints, we decided not to build a separate IKE key ring for SC32 and the TCPIPC stack; we decided instead to share the IKE key ring between TCPIPC and TCPIPE. The solution would have worked equally well if we had created a unique IKE keyring per MVS image; recall the earlier discussion of key ring strategies: "Key rings for the NSS implementation" on page 92.

Figure 4-89 shows four important bullets:

1. This represents the secure TLS connection over which IKEDC communicates with the NSS server on behalf of TCPIPA. This connection is also the only one in which NSSD is directly involved.
2. This represents the secure IKE tunnel over which IKEDC and IKEDD negotiate Security Associations.
3. This represents the secure IPSec connection over which stacks TCPIPC and TCPIPE communicate.

4. This represents the secure IPSec connection over which stacks TCPIPA and TCPIPE communicate.

We ping from SC32 (TCPIPC) to SC33 (TCPIPE) to ensure that we can still bring up an IPSec tunnel without NSS services; see Example 4-77.

Example 4-77 Establish IPSEC tunnel; display the tunnel between TCPIPC and TCPIPE

```
CSO2 @ SC32:/u/csO2>ping -p TCPIPC 192.168.1.40      a
CS V1R9: Pinging host 192.168.1.40
sendto(): EDC5111I Permission denied. (errno2=0x74420291) b
CSO2 @ SC32:/u/csO2>ping -p TCPIPC 192.168.1.40 c
CS V1R9: Pinging host 192.168.1.40
Ping #1 response took 0.001 seconds.


CSO2 @ SC32:/u/csO2>ipsec -p TCPIPC -y display d

CS V1R9 ipsec  Stack Name: TCPIPC  Tue Jan  8 18:23:58 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                    Y2 e
ParentIKETunnelID:           K1
VpnActionName:               IPSec__Silver
LocalDynVpnRule:             n/a
State:                       Active
HowToEncap:                  Transport
LocalEndPoint:               10.1.1.30
RemoteEndPoint:              192.168.1.40
LocalAddressBase:            10.1.1.30
LocalAddressPrefix:          n/a
LocalAddressRange:           n/a
RemoteAddressBase:           192.168.1.40
RemoteAddressPrefix:         n/a
RemoteAddressRange:          n/a
HowToAuth:                   ESP
 AuthAlgorithm:              Hmac_Sha
 AuthInboundSpi:             3388742348
 AuthOutboundSpi:            23680591
HowToEncrypt:                DES
 EncryptInboundSpi:          3388742348
 EncryptOutboundSpi:         23680591
Protocol:                    ALL(0)
LocalPort:                   0
RemotePort:                  0
OutboundPackets:             1
OutboundBytes:               264
InboundPackets:              1
InboundBytes:                264
Lifesize:                    OK
LifesizeRefresh:             OK
CurrentByteCount:            Ob
LifetimeRefresh:             2008/01/08 21:32:46
LifetimeExpires:             2008/01/08 22:23:32
CurrentTime:                 2008/01/08 18:23:58
VPNLifeExpires:              2008/01/09 18:23:32
NAT Traversal Topology:
  UdpEncapMode:              No
  LclNATDetected:            No
  RmtNATDetected:            No
```

```
  RmtNAPTDetected:           No
  RmtIsGw:                   n/a
  RmtIsZOS:                  n/a
  zOSCanInitP2SA:            n/a
  RmtUdpEncapPort:           n/a
  SrcNATOARcvd:              n/a
  DstNATOARcvd:              n/a
**********************************************************************

1 entries selected
```

a and **b**: the first time we ping TCPIPE, the connection does not succeed. (See the following note for an explanation.) The subsequent ping **c** succeeds and brings up a tunnel.

d shows the command we use to display the dynamic tunnel

e shows the tunnel ID.

> **Why the connection fails at the first ping command:** Our IPSec policy has been specified with "OnDemand." ICMP commands, like **ping**, other RAW commands, and UDP protocols, do not have retry logic. Therefore, the first **ping** activates the OnDemand tunnel, making the tunnel ready for the second **ping** command, which succeeds. TCP flows do have retry logic and thus bring up the tunnel so that the subsequent retry succeeds without any error messages.
>
> Other options for activating a dynamic VPN tunnel are manual, with a command, and autoactivate, which initiates the tunnel when TCP/IP and IKE are initialized (and before a user or program may need the tunnel available).

So far IKE and IPSec still function between the SC32 and SC33 images. We know that this stack has used native IKE services, because we find the following message in the local4.log on the SC32 MVS image:

```
IKE: Message instance 101: EZD0990I The IKE daemon is set up to support RSA signature mode
of authentication for stack TCPIPE using the local keyring.
```

Next, we want to bring up a connection between TCPIPA, which is an NSS client, and TCPIPE; see the output in Example 4-78.

Example 4-78 Establish IPSEC tunnel using NSS; display the tunnel between TCPIPA and TCPIPE

```
CS02 @ SC32:/u/cs02>ping -p TCPIPA 192.168.1.40  a
CS V1R9: Pinging host 192.168.1.40
Ping #1 response took 0.001 seconds.

CS02 @ SC32:/u/cs02>ipsec -p TCPIPA -y display b

CS V1R9 ipsec  Stack Name: TCPIPA  Tue Jan  8 18:31:49 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                  Y4 c
ParentIKETunnelID:         K3
VpnActionName:             IPSec__Silver
LocalDynVpnRule:           n/a
State:                     Active
HowToEncap:                Transport
```

```
LocalEndPoint:                10.1.1.50
RemoteEndPoint:               192.168.1.40
LocalAddressBase:             10.1.1.50
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
RemoteAddressBase:            192.168.1.40
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              3854124971
 AuthOutboundSpi:             2364884043
HowToEncrypt:                 DES
 EncryptInboundSpi:           3854124971
 EncryptOutboundSpi:          2364884043
Protocol:                     ALL(0)
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              1
OutboundBytes:                264
InboundPackets:               1
InboundBytes:                 264
Lifesize:                     0K
LifesizeRefresh:              0K
CurrentByteCount:             0b
LifetimeRefresh:              2008/01/08 21:50:21
LifetimeExpires:              2008/01/08 22:31:28
CurrentTime:                  2008/01/08 18:31:49
VPNLifeExpires:               2008/01/09 18:31:28
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
************************************************************************
```

a We ping TCPIPE from TCPIPA and are successful.

b shows the command we use to display the dynamic tunnel.

c shows the tunnel ID for the tunnel between TCPIPA and TCPIPE.

However, the information displayed in Example 4-78 on page 209 does not show that the tunnel was established due to Network Security Services. We can determine this information by reviewing NSSD and IKED log entries. We show you these in the next section.

4.5 Diagnosing the NSSD environment

The two most important publications to use to diagnose problems in an NSS implementation are:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781

It is useful to review this information, because it contains most of the common configuration problems with their solutions. For example, *z/OS Communications Server: IP Configuration Guide*, SC31-8775 contains tables with the following information:

- Common initialization problems
- Common client connection problems
- Obtaining debug information for the server (includes server error codes)
- Component trace procedures for NSS

z/OS Communications Server: IP System Administrator's Commands, SC31-8781, contains the syntax for the **ipsec** command. If you are in the UNIX environment and do not recall the syntax, issue the following from the OMVS shell environment to view the syntax options:

```
ipsec -?
```

You will also be making use of the **netstat** command variations and of **pasearch** to diagnose problems. Be sure to understand how to issue the **racdcert list** commands as well. Here is a list of the other relevant commands:

```
netstat
pasearch
racdcert
```

You will find that the syslog daemon processes are extremely important in diagnosing problems you may run into. The most important messages for diagnosing problems are:

- IKE daemon messages
- TRMD messages
- PAGENT messages
- NSS daemon messages

Your syslog daemon implementation may place the pertinent messages in separate logs or in one large log. However, many of the messages you may need for advanced diagnosis are only available if you raise the logging levels.

Note: Allow your initial logging levels to default when you configure the files with IBM Configuration Assistant. After you upload the various files to the mainframe, and if you discover that NSS is not working as expected, then you may manually edit the PAGENT, IKED, and NSSD configuration files and any of the policy files to raise the logging levels temporarily.

Remember to reset the logging levels to a lower value after you have diagnosed and remedied the problems.

4.5.1 Examples of logging information for diagnosis

In this section we discuss examples of logging information you can use to diagnose problems.

Example 4-79 on page 212 displays messages that you might receive when you cannot obtain a TLS connection between the NSS client and the NSS server.

Example 4-79 At SC33 MVS system log - the NSS server node

```
EZD1150I THE IKE DAEMON FAILED TO CONNECT 3 TIMES TO THE PRIMARY NSS
SERVER AT 192.168.1.40 PORT 4159 FOR STACK TCPIPA
EZD1150I THE IKE DAEMON FAILED TO CONNECT 3 TIMES TO THE PRIMARY NSS
SERVER AT 192.168.1.40 PORT 4159 FOR STACK TCPIPA
```

Example 4-80 shows the SERVAUTH checking that is done when the NSS client attempts to initiate a connection.

Example 4-80 SERVAUTH checking at NSS server

```
NSSD: DBG0032I NSS_VERBOSE ServauthCheck(NSS32A  ,EZB.NSS.SC33.SC32_TCPIPA.IPSEC.CERT) a rc
0 (AUTH) racfRC 0 racfRsn 0
NSSD: DBG0032I NSS_VERBOSE ServauthCheck(NSS32A  ,EZB.NSS.SC33.SC32_TCPIPA.IPSEC.NETMGMT)
rc 0 (AUTH) racfRC 0 racfRsn 0
NSSD: DBG0103I NSS_LIFECYCLE  NSS connID   1 ::ffff:10.1.1.50..11569 - SC32_TCPIPA
,SC32    ,TCPIPA  ,NSS32A   - NSS_ConnectClientReqToSrv request
NSSD: DBG0035I NSS_VERBOSE ConnectClientReqToSrv clientname (SC32_TCPIPA
) connID 1 rc 0 reason 0 CertServices Y RemoteMgmt Y
NSSD: DBG0032I NSS_VERBOSE ServauthCheck(CSO2
2    ,EZB.NETMGMT.SC33.SC33.NSS.DISPLAY) rc 0 (AUTH (cached)) racfRC 0 racfRsn 0
```

a Notice the servauth class that we created: EZB.NSS.SC33.SC32_TCPIPA.IPSEC.CERT.

Also notice the other messages for DISPLAY, NETMGMT, and CertServices that are shown.

Example 4-81 displays the TLS processing when the client requests NSS services.

Example 4-81 Local4.log at NSS Client: TLS processing for NSS client connection

```
IKE: EZD1067I IKE configuration file parameter ( KeyRing ) is non-refreshable on line 65
IKE: EZD1067I IKE configuration file parameter ( PagentWait ) is non-refreshable on line 95
IKE: EZD0911I IKE config processing complete using file /etc/security/ikedNSSClient.conf
IKE:
IKE: Updating CA Cache
IKE: Begin display of CA Cache
IKE: End display of CA Cache
IKE: Update of CA Cache Complete
IKE: EZD1138I The IKE daemon is connecting to the NSS server at 192.168.1.40 port 4159 for
stack TCPIPA
EZD1283I TTLS Event GRPID: 00000005 ENVID: 00000000 CONNID: 00013596  RC:    0 Connection
Init
EZD1282I TTLS Start GRPID: 00000005 ENVID: 00000004 CONNID: 00013596 Initial Handshake
ACTIONS: gAct1~NSS_Client eAct1~NSS_Client cAct1~NSS_Client HS-Client
EZD1284I TTLS Flow  GRPID: 00000005 ENVID: 00000004 CONNID: 00013596  RC:    0 Call
GSK_SECURE_SOCKET_OPEN - 7EA45118 et GSK_SESSION_TYPE -  CLIENT
EZD1284I TTLS Flow  GRPID: 00000005 ENVID: 00000004 CONNID: 00013596  RC:    0 Set
GSK_V3_CIPHER_SPECS -  090A2F
EZD1284I TTLS Flow  GRPID: 00000005 ENVID: 00000004 CONNID: 00013596  RC:    0 Set GSK_FD -
00013596
EZD1284I TTLS Flow  GRPID: 00000005 ENVID: 00000004 CONNID: 00013596  RC:    0 Set
GSK_USER_DATA - 7EABADD0
IKEDC    IKE: EZD1136I The IKE daemon is connected to the NSS server at 192.168.1.40 port
4159 for stack TCPIPA
IKEDC    IKE: EZD1144I The NSS certificate service is available for stack TCPIPA
IKEDC  IKE: EZD1146I The NSS remote management service is available for stack TCPIPA
IKEDC    IKE: Initializing CA Cache with 2 entries for stack TCPIPA
```

```
IKEDC  IKE: Initialization of CA Cache with 2 entries Complete for stack TCPIPA
IKEDC   IKE: EZD1124I NMI connection received from user CSO2     - 1 connections open
IKEDC   IKE: EZD1123I NMI connection from user CSO2    closed - 0 connections remain open
IKEDC    IKE:
IKEDC    IKE: Sending message:
IKEDC    IKE: Source Address: 10.1.1.50 Port: 500
IKEDC    IKE: Destination Address: 192.168.1.40 Port 500
```

If you are having problems with certificates, the log to which the local4.* messages or the NSSD messages are sent can be helpful because it contains information about all the certificates contained in the key ring the NSS is using; see Example 4-82.

Example 4-82 NSSD messages in syslog (local4.)*

```
EZD1359I Network security services (NSS) server Release CS V1R9 Service Level CSCS070402
created on Apr  2 2007
EZD1357I Network security services (NSS) server initialization sequence has begun

EZD1337I NSS server is using TCP port 4159
EZD1353I NSS server config processing complete using file /etc/security/nssd33.conf
DBG00005I NSS_CERTINFO Certificate (IKE Daemon on SC32) does not have a private key
DBG00008I NSS_CERTINFO Certificate (IKE Daemon on SC32) is not self-signed and marked as
trusted
EZD1324I Certificate ( IKE Daemon on SC32 ) cannot be used to create a signature and is not
a Certificate Authority certificate a
DBG00005I NSS_CERTINFO Certificate (My Local Certificate Authority) does not have a private
key
DBG00005I NSS_CERTINFO Certificate (SC33 RACF CA) does not have a private key
DBG00005I NSS_CERTINFO Certificate (SC33 Server) does not have a private key
DBG00008I NSS_CERTINFO Certificate (SC33 Server) is not self-signed and marked as trusted
EZD1324I Certificate ( SC33 Server ) cannot be used to create a signature and is not a
Certificate Authority certificate
DBG00005I NSS_CERTINFO Certificate (WINXP Client2) does not have a private key
DBG00005I NSS_CERTINFO Certificate (WINXP Client222) does not have a private key
DBG00002I NSS_CERTINFO Certificate (My Local Certificate Authority) can be used as a
certificate authority certificate b
DBG00002I NSS_CERTINFO Certificate (SC33 RACF CA) can be used as a certificate authority
certificate
DBG00002I NSS_CERTINFO Certificate (WINXP Client2) can be used as a certificate authority
certificate
DBG00002I NSS_CERTINFO Certificate (WINXP Client222) can be used as a certificate authority
certificate
EZD1328I Certificate repository NSSD/NSSD_keyring successfully processed for NSS
initialization c
EZD1358I NSS server initialization sequence has completed
DBG00043I NSS_VERBOSE SERVAUTH - waiting for ECB to be posted
DBG00037I NSS_VERBOSE (00001.ezd.nss.connID....) connID 1 added to clienttable
```

Although there are many messages shown in Example 4-82 that look alarming, many of these certificates are not important to our implementation.

a This certificate is the IKE certificate for our client; it is unimportant that it is not a CA certificate. It is, however, vitally important that it have a private key and that it can be used to create a signature.

b This certificate has signed our client's IKE certificate; it is important that this is a CA certificate.

c The IKE key ring named is the one we expected.

4.6 Worksheet for NSSD implementation

In this section we provide a worksheet that you can copy and fill in for use with your own NSS implementation. Before you fill in the worksheet, it is helpful to create a simple network diagram similar to the one we created for our NSS implementation, as shown in Figure 4-32 on page 133. Developing a logical network diagram will assist you in your implementation.

Example 4-83 Worksheet reflecting your NSS implementation

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon? Your response is:

   –

2. What is the superuser user ID associated with the OMVS segment under which the NSS Daemon is to run? Your response is:

   –

3. With which TCP/IP stack or stacks does the NSS daemon establish affinity? Your response is:

   –

4. What is the address or the DNS name that the NSS client will use to connect to the NSS server? Your response is:

   –

5. What is the MVS System ID under which each NSS client is to run? What are the TCP/IP stack names that are to use the IKE services of the NSS client? Your responses are:

   – MVS id:

     • TCPname:

6. What “symbolic client name” name do you want to assign to each TCP/IP stack that is to request authentication of the NSS server? Your answer is:

   –

7. What “client ID” do you want to assign to each TCP/IP stack that is to request authentication of the NSS server? Your response is:

   –

8. What type of authentication should our client user ID present: a password or a passticket? Your response is:

   – Authentication Type:

   – Value if applicable:

9. What is the user ID associated with the OMVS Segment of the client’s IKEDC procedure? Your answer is:

   –

10. What is the name of the key ring that NSS should use to manage all of the clients’ private keys and certificates? Your response is:

    –

11. What is the certificate label assigned to each IPSec RSA certificate for each TCP/IP stack? Display with **`racdcert listring(IKED_keyring) id(IKED).`** Your response is:

    – For client named _____ the label is: _____

12. With which user ID must the NSS Client certificate be associated when it is moved to the NSSD key ring? (That is, what is the name of the user ID under which NSSD will run?) Your response is:

    –

13. What is the name of the key ring on the NSS node that is used for the TLL/SSL secure connection between the NSS server and client? Your response is:

    –

14. What is the name of the key ring on the NSS client node that is used for the TLL/SSL secure connection between the NSS client and server? Your response is:

    –

15. What are the IP addresses of the NSS server and the NSS client for which you may have to establish IP filters on the NSS client node? Your responses are:

    – Client:

    – Server:

16. If the NSS server has implemented IPSec, what are the IP addresses of the NSS server and the NSS client for which you may have to establish IP filters on the NSS server node? Your responses are:

    – Client:

    – Server:

17. What are the TSO user IDs of the IPSec administrators who are authorized to manage the VPN? Your response is:

    –

4.7 References

For additional information about Network Security Services, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *Enterprise Systems Architecture/390 Reference Summary*, SA22-7209

Part 3

Policy-based networking

For as long as Information Technology (IT) components have been shared, we have needed to make (and enforce) choices as to who is allowed to access specific IT applications, and which applications get priority over others.

Historically, such choices have been implemented individually for each component; for example:

- Setting up application access and task priorities in each computer system (such as using ftp.data and telnetglobals for security-specific configuration of File Transfer Protocol (FTP) and Telnet, respectively)
- Defining network traffic priorities in each piece of network equipment

Ultimately, however, those application and traffic decisions must be determined by business priorities, also called *business policies*.

As the complexity of IT environments has increased, it has become increasingly difficult to configure each system and network component individually, and yet still ensure that the overall system usage (and especially the network) matches the required business policies. Consequently, policy-based networking has emerged as a standards-based approach for defining policies in one place and applying them broadly across the entire IT environment.

In this part, we discuss the z/OS Communications Server Policy Agent (Chapter 5, “Policy Agent” on page 221) and its use in defining:

- Quality of Service
- IP filtering
- IP security

- Network Address Translation traversal support
- Intrusion Detection Services
- Policy-based routing

IBM Configuration Assistant for z/OS Communication Server is provided to assist with configuration of the following z/OS Communications Server TCP/IP functions:

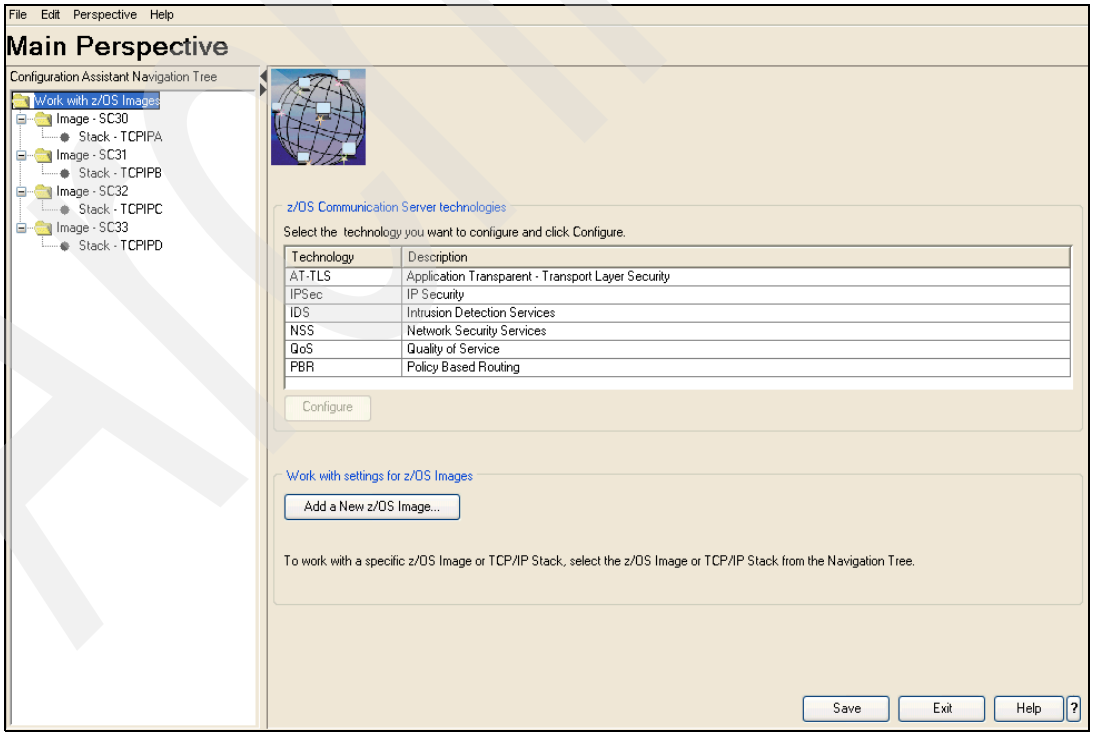
- Network Security Services (NSS)
- Quality of Service (QoS)
- IP filtering
- IP Security (IPSec)
- Network Address Translation (NAT) traversal support
- Application Transparent TLS (AT-TLS)
- Intrusion Detection Services (IDS)
- Policy Based Routing (PBR)

The Configuration Assistant GUI replaces the GUIs that were previously provided as Web downloads for configuring the following functions:

- zQoS Manager for configuring QoS
- zIDS Manager for configuring IDS
- Network Security Configuration Assistant (NCSA) for configuring AT-TLS and IPSec

Configuration files created by these previously existing GUIs can be imported into the IBM Configuration Assistant GUI, which makes migration easy.

The Configuration Assistant GUI is presented in a stack-oriented manner, rather than in a functional- or technology-oriented manner. The stack-oriented manner provides a view of what is being configured in each image, as shown here.



Configuration Assistant GUI main panel

To provide this view, the Configuration Assistant GUI is enhanced to combine the configuration data of the policy-based TCP/IP functions (technologies) into a single configuration file. One or more of the TCP/IP functions can be configured for one or more TCP/IP stacks; all of the configuration data is kept together in a single configuration file.

The Configuration Assistant GUI provides an internal function to look for configuration problems within a single technology, and now it also checks configuration errors and inconsistencies across multiple technologies. Additionally, the management of the configuration data files has been extended. Configuration data files can be kept on the local workstation file system, or they can be kept on a remote host (z/OS) system (as either a UNIX System Services file or MVS data set).

To download the GUI and for additional information, see the IBM Communications Server product support Web site:

http://www.ibm.com/software/network/commserver/zos/support/

5

Policy Agent

The Policy Agent (PAGENT) is a component within server platforms that is responsible for implementing policy decisions. This chapter focuses on the z/OS Communications Server Policy Agent and its related security functions. Policy Agent enforces a set of rules and policies that dictate how users, applications, and organizations can access and use IT resources. We show how policy-based networking introduces a centralized policy storage approach, and how it interacts with other security functions.

The following topics are discussed in this chapter.

Section	Topics
5.1, “Policy Agent description” on page 222	Using Policy Agent as a central repository for policies. Basic concepts of the Policy Agent.
5.2, “Implementing PAGENT on z/OS” on page 229	Basic Policy Agent installation.
5.3, “Backup and migration considerations” on page 240	Finer points of migrating from one release of Policy Agent to a newer release, and of merging separate policy files.
5.4, “Setting up Traffic Regulation Management Daemon” on page 254	Setting up Traffic Regulation Management Daemon (TRMD) to work in conjunction with Intrusion Detection Services (IDS) to handle Policy Agent messages.

5.1 Policy Agent description

As illustrated in Figure 5-1, the z/OS Communications Server PAGENT implements policy-based networking for the z/OS environment.

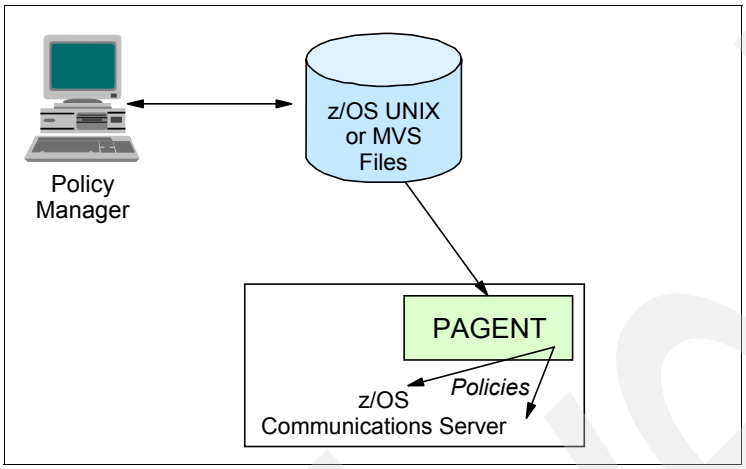


Figure 5-1 Policy-based networking and PAGENT

Policy definitions are usually contained in local configuration flat files; some policy types may, alternatively, be stored in an LDAP server. The policies are used to control network security, traffic prioritization, bandwidth management, network behavior, and resource balancing for the z/OS environment. The Policy Agent reads these configuration files, parses the policies, and stores the policy definitions in the TCP/IP stack. The policies are then acted on by a TCP/IP stack.

Important: The current release of PAGENT allows policies to be read from configuration files, a Lightweight Directory Access Protocol (LDAP) server, or both. With the availability of a "centralized policy server" on z/OS through distributed policy services, LDAP is no longer the only way to maintain a central repository for policies. We therefore recommend using flat text files for building networking policies.

As a result, in our case we exclusively used the IBM Configuration Assistant for z/OS Communications Server GUI to create our flat file text configuration files.

The policies supported by PAGENT are discussed in detail in the subsequent chapters as follows:

- Chapter 7, "Quality of Service" on page 295
- Chapter 8, "IP filtering" on page 319
- Chapter 9, "IP Security" on page 373
- Chapter 10, "Network Address Translation traversal support" on page 425
- Chapter 11, "Application Transparent Transport Layer Security" on page 443
- Chapter 12, "Intrusion Detection Services" on page 471
- Chapter 13, "Policy-based routing" on page 501

5.1.1 Basic concepts

Architecturally, policy-based networking typically involves the following components:

- Policy management service

  A graphical user interface (GUI) for specifying, editing, and administering policies, such as the new Configuration Assistant.

- Policy repository

  A place to store and retrieve policy information, such as a configuration file.

- Policy decision point (PDP)

  A resource manager or policy server that is responsible for handling events and making decisions based on those events, and updating the Policy Enforcement Point configurations appropriately. Policy Agent (PAGENT) is our PDP, as shown in Figure 5-2.

- Policy enforcement point (PEP)

  A PEP exists in network nodes such as routers, firewalls, and hosts. It enforces the policies based on the “if condition then action” rule sets it has received from the PDP. In Figure 5-2, the TCP/IP stack serves as our PEP.

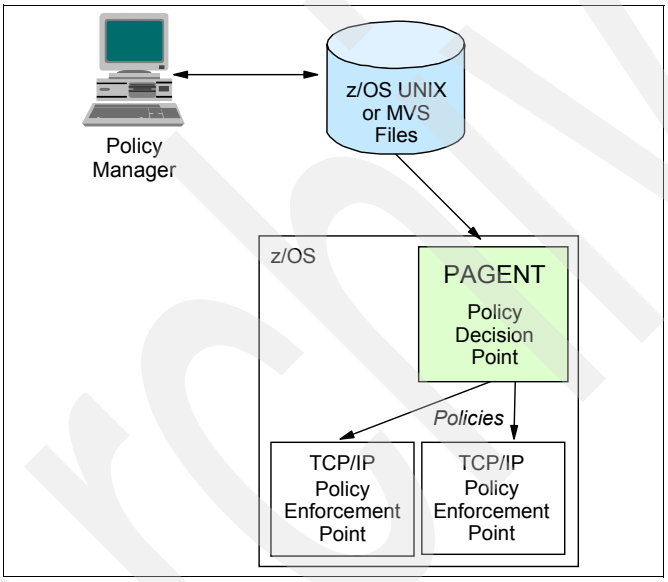


Figure 5-2 Policy system model

Where and how we define policies

Policies are defined in the Policy Agent configuration file, as shown in Figure 5-3 on page 224. Keep policy names unique, because policy objects with duplicate names run the risk of being deleted by PAGENT.

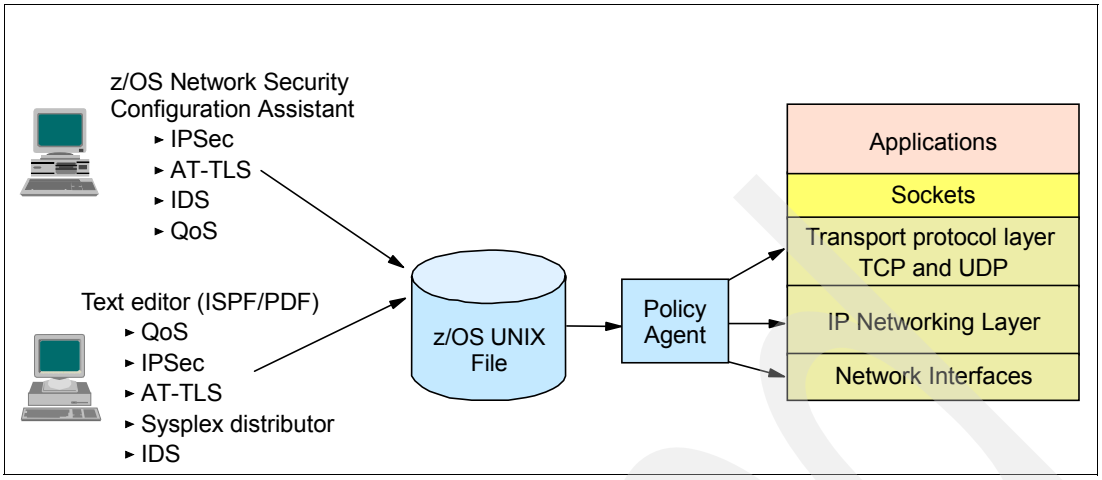


Figure 5-3 Configuring Policy Agent

The following PAGENT policies can be stored in a configuration text file format:

- Quality of Service (QoS)
- IDS
- IP Security (IPSec) Virtual Private Network (VPN)
- IP filtering
- Application Transparent TLS (AT-TLS)
- Sysplex Distributor
- Policy-based routing

Figure 5-4 show the components involved in the networking policies of z/OS Communications Server.

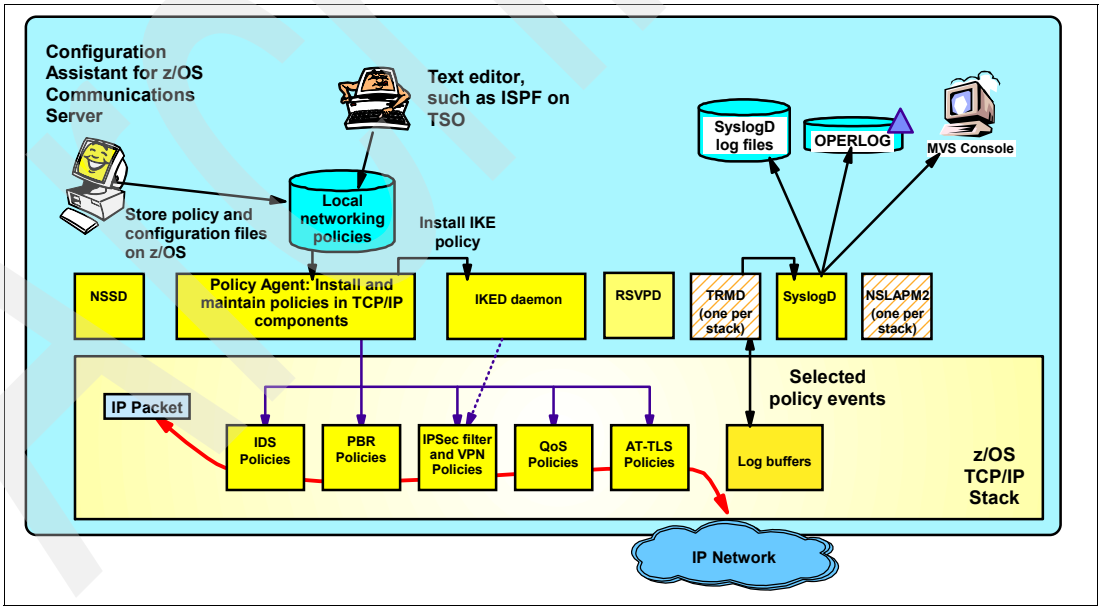


Figure 5-4 Full Policy Agent implementation on z/OS

In Table 5-1 on page 225, we show the policy types that are used by the TCP/IP stack. Some of the necessary address spaces for these policies can be shared by multiple TCP/IP stacks. Other address spaces are unique to a particular stack. In the table, R means required for use of the policy type and O means optional.

Table 5-1 Address spaces for Policy Agent

Policy type	Shared by multiple TCP/IP stacks in an LPAR					Unique to a particular TCP/IP stack			
	PAGENT	NSSD[a]	IKED	RSVPD	SYSLOGD	TRMDA	NSLAPM2A	TRMDB	NSLAPM2B
QoS	R			O	R	R	O	R	O
IDS	R				R	R		R	
AT-TLS	R				R	R		R	
IPSec filters	R				R	R		R	
IPSec static VPNs	R				R	R		R	
IPSec dynamic VPNs	R	O	R		R	R		R	
PBR	R				R	R		R	

a. NSSD is actually shared by all stacks in all LPARs in the NSSD domain (which could be a sysplex or span a multiple sysplex environment).

To aid in setting up policies, we recommend using the IBM Configuration Assistant for z/OS Communications Server. It can build the configuration flat files for QoS, IDS, AT-TLS, IPSec, and Policy-based routing (PBR) policies, as you see in Table 5-2.

Table 5-2 Configuration methods for Policy Agent files

Configuration and policy definitions	Manual edit (ISPF)	Configuration Assistant
Policy Agent configuration file	Yes	No
Syslogd configuration file	Yes	No
IKED configuration file	Yes	Yes
NSSD configuration file	Yes	Yes
RSVPD configuration file	Yes	No
QoS policy file	Yes	Yes
IDS policy file	Yes	Yes
AT-TLS policy file	Yes	Yes
IPSec policy file	Yes	Yes
PBR policy file	Yes	Yes

Table 5-2 also shows that the same policy files can be built using a manual edit. When you examine the listing of configuration files for Policy Agent and its policy types, you see that several of the files must be configured using manual edit: the Policy Agent, SyslogD, and RSVPD configuration files. The remaining two files, the IKED and NSSD configuration files, may be defined using either the manual edit process or the Configuration Assistant.

We recommend using the Configuration Assistant GUI whenever possible to avoid syntax errors that can occur with manual editing. There is also a “health check” feature in the GUI that can even detect many of the logic errors that you might introduce when building policies.

This GUI is available for download from the Web at:

http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=D400&uid=swg24013160&loc=en_US&cs=UTF-8&lang=en

5.1.2 The policy model

Service policies consist of policy rules and policy actions. When the policy rule is true, one set of actions is initiated. When the policy rule is false, a different set of actions is initiated. The policy rule is the condition. Conditions can involve both time specifications and traffic filters; however, if both are used, then both have to match in order for the condition to be true. The policy action is the action to be performed. To learn more about Policy Agent rules, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example of a policy rule and action statement

The example policy definition in Figure 5-5 on page 227 causes the stack to discard all Telnet requests from subnet 10.12.4.224 to 10.12.4.254 on Port 23. Note how we have restricted the range to a single port (23), although there might be other Telnet servers at other ports available on the IP stack, including a UNIX Telnet server. It may not be 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 on the z/OS systems, you need to include a policy rule for each of them. Policy Agent blocking is not meant to be a replacement for firewall filtering. PAGENT should be considered only as a second or third line of defense for certain types of actions.

Rules that intend to block traffic apply to both inbound and outbound traffic, whether or not the traffic originates at the z/OS Communications Server - TCP/IP component.

For actions specifying a permission of blocked, TCP traffic is handled differently from UDP or RAW traffic, 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 TCP connection control block (TCB) is built. Any other QoS rules (for example, TOS settings or similar) can now be applied to the outbound connection. If the inbound rule permits the connection but the outbound connection 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. Inbound blocking rules are ignored. When the SYN/ACK arrives back at the z/OS Communications Server 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.

With a flat file policy definition, the *source* indicates the source of the data flow, and the *destination* signifies the target of the data traffic. Therefore, in some cases the source is z/OS Communications Server IP and in others, the source is the remote host.

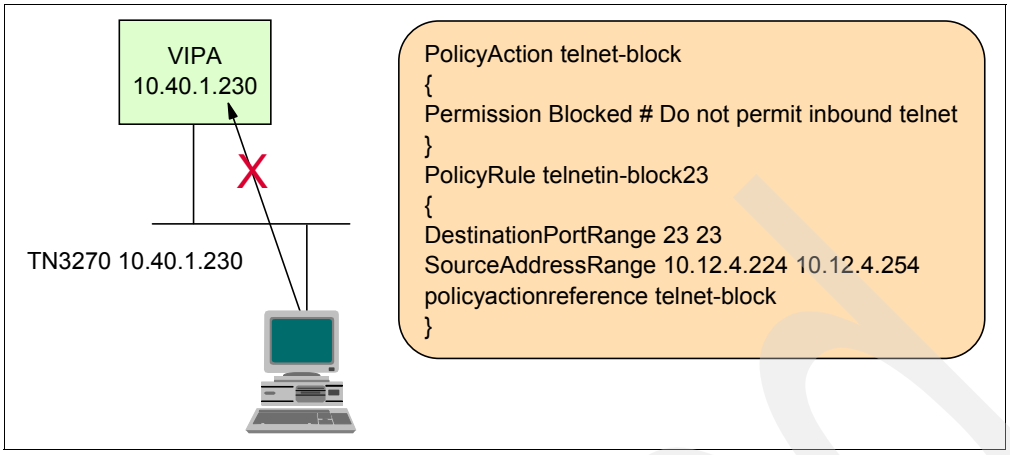


Figure 5-5 Policy rule and action statement

The TCP/IP stack receives policies from the following two sources:

- Policy Agent, which usually has its policies stored in a flat file
- The PROFILE.TCPIP statements
  - IPCONFIG IPSECURITY SOURCEVIPA
  - IPSEC
  - ipsec rules
  - ENDIPSEC

If using the PROFILE.TCPIP statements without Policy Agent IPSec policies, note the following:

- The IPSEC/ENDIPSEC block statement is ignored, if IPSECURITY is not specified on the IPCONFIG statement.
- Only one IPSEC/ENDIPSEC block statement block should appear in the profile. Any subsequent statement blocks are ignored.
- If you code IPCONFIG IPSECURITY with no IPSEC/ENDIPSEC block statement, only local traffic will flow through the stack (that is, loopback addresses).
- No routing functions are supported by the stack through these statements even with DATAGRAMFWD coded in the PROFILE.TCPIP. Routing function can only be performed based on the policies read in from PAGENT.

Example 5-1 shows the sample shipped in member SAMPPROF of the SEZAINST data set.

Example 5-1 Example IPCONFIG for IPSECURITY support

```
; IPCONFIG IPSECURITY
; ------------------------------------------------------------------------
; Configure IPSECURITY default filter rules
; ------------------------------------------------------------------------
;
; Example IPSEC default filter rule. This rule permits
; outbound TCP traffic from local IP address 1.1.1.1 port 23 to
; remote IP address 2.2.2.2. The same rule also permits
; inbound TCP traffic from remote IP address 2.2.2.2 to local
; IP address 1.1.1.1 port 23.
;
```

```
; IPSEC LOGDISABLE NOLOGIMPLICIT
; Rule   SrcIp   DestIp  Log   Prot      SrcPort    DestPort  Secclass
; IPSECR 1.1.1.1 2.2.2.2 NOLOG PROTO TCP SRCPORT 23 DSTPORT *
; ENDIPSEC
```

Note: TCP/IP does not use both sets of policies simultaneously. It uses the IPSEC policies from the profile when PAGENT is not active, and swaps to the PAGENT policies when PAGENT is active.

PAGENT installs two default policy rules, which deny all inbound and outbound traffic. The active policies can be displayed from the UNIX shell using the **pasearch** command. We displayed our active policies and used the **pasearch** command to establish the presence of the default rules, which can be seen in Example 5-2.

Example 5-2 The pasearch command showing default policies

```
policyRule:             DenyAllRule_Generated__________Inbnd
  Rule Type:            IpFilter
  Version:              3                 Status:           Active
  Weight:               102               ForLoadDist:      False
  Priority:             2                 Sequence Actions: Don't Care
  No. Policy Action:    0
  IpSecType:            policyIpFilter
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:     24:00
   Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC: 04:00
   TimeZone:            Local
  IpSec Condition Summary:                NegativeIndicator: Off
   IpFilter Condition:
     FromAddr:          All
     ToAddr:            All
    Destination Address:
     FromAddr:          All
     ToAddr:            All
    Service Condition:
     Protocol:          All
     Direction:         Inbound
     RouteType:         Either            SecurityClass:    0

policyRule:             DenyAllRule_Generated__________Outbnd
  Rule Type:            IpFilter
  Version:              3                 Status:           Active
  Weight:               101               ForLoadDist:      False
  Priority:             1                 Sequence Actions: Don't Care
  No. Policy Action:    0
  IpSecType:            policyIpFilter
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
```

```
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00            To TimeOfDay:      24:00
   Fr TimeOfDay UTC:    04:00            To TimeOfDay UTC:  04:00
   TimeZone:            Local
 IpSec Condition Summary:                NegativeIndicator: Off
   IpFilter Condition:
```

Important: When dealing with policies that deny all traffic, it is imperative to permit traffic to some essential services.

The services listed in Table 5-3 have to be available to the stack.

Table 5-3 Stack services

Service	Direction	Source port	Destination port	Protocol
Resolver	Outbound	53	Any	TCP or UDP
Resolver	Inbound	53	Any	TCP or UDP
Omproute - RIPV1	Outbound	520	520	UDP
Omproute - RIPV1	Inbound	520	520	UDP
Omproute - RIPV2	Outbound	520	520	UDP and IGMP
Omproute	Inbound	520	520	UDP and IGMP
Omproute - OSPF	Outbound			IP and IGMP
Omproute - OSPF	Inbound			IP and IGMP

Even though not listed under essential services, PING using protocol Internet Control Message Protocol (ICMP) can be useful during problem determination.

5.2 Implementing PAGENT on z/OS

The Policy Agent runs as a UNIX process. As such, it can be started either from the UNIX System Services shell or as a started task. We used a started task procedure to start Policy Agent.

5.2.1 Starting PAGENT as started task

The sample started task procedure for PAGENT can be found in TCPIP.SEZAINST(EZAPAGSP). We used the following PAGENT started task procedure on our system; see Example 5-3.

Example 5-3 PAGENT started task procedure

```
//PAGENT   PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZAPAGSP
```

```
//*
//* 5694-A01 (C) Copyright IBM Corp. 1998, 2005
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV1R9
//*
//PAGENT   EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//*      PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
//*            etc/pagent3.conf -l SYSLOGD'
//*
//* Provide environment variables to run with the desired
//* configuration.  As an example, the data set or file specified by
//* STDENV could contain:
//*
//*   PAGENT_CONFIG_FILE=/etc/pagent2.conf
//*   PAGENT_LOG_FILE=/tmp/pagent2.log
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//*STDENV   DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV   DD DSN=TCPIP.PAGENT.ENV(PAGENT),DISP=SHR
//* Sample z/OS UNIX file containing environment variables:
//STDENV   DD PATH='/etc/pagent.sc30.env',PATHOPTS=(ORDONLY)1
//*
//* Output written to stdout and stderr goes to the data set or
//* file specified with SYSPRINT or SYSOUT, respectively.  But
//* normally, PAGENT doesn't write output to stdout or stderr.
//* Instead, output is written to the log file, which is specified
//* by the PAGENT_LOG_FILE environment variable, and defaults to
//* /tmp/pagent.log.  When the -d parameter is specified, however,
//* output is also written to stdout.
//*
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

You can use environment variables either configured in an IBM MVS data set or z/OS UNIX file specified by the STDENV DD to run with the required configuration. We configured our environment variables in a z/OS UNIX file, /etc/pagent.sc30.env, as shown in Example 5-4.

Note: Ensure that the z/OS UNIX file pointed to by STDENV, and the files contained in this STDENV file, have the correct permission bits set to allow PAGENT access to these files.

Example 5-4 STDENV data set contents

```
LIBPATH=/lib:/usr/lib:/usr/lpp/ldapclient/lib:.  1
PAGENT_CONFIG_FILE=/SC30/etc/pagent.sc30.conf 2
PAGENT_LOG_FILE=/SC30/tmp/pagent.sc30.log     3
PAGENT_LOG_FILE_CONTROL=300,3                 4
_BPXK_SETIBMOPT_TRANSPORT=TCPIPA
TZ=EST5EDT
```

We configured the following environment variables for the Policy Agent:

1 LIBPATH enables PAGENT to search the dynamic link libraries needed to act as an LDAP client.

2 PAGENT_CONFIG_FILE specifies the specific PAGENT configuration file to use.

3 PAGENT_LOG_FILE specifies the log file name used by PAGENT.

4 PAGENT_LOG_FILE_CONTROL defines the number of PAGENT log files and their size in kilobytes. In our case, we requested three log files, each 300 KB in size. These are used in a round-robin fashion.

To configure PAGENT to use SYSLOGD to log messages, you can define PAGENT_LOG_FILE=SYSLOGD. In this case, PAGENT_LOG_FILE_CONTROL has no meaning.

- In our case, although we do not have the RESOLVER_CONFIG variable configured, PAGENT establishes an affinity to the proper TCP/IP stack through BPXK_SETIBMOPT_TRANSPORT=TCPIPA. The TcpImage statement in the Policy Agent configuration file also determines to which TCP/IP stack PAGENT will install policies.
- For the Policy Agent to run in your local time zone, you might have to specify the time zone in your working location using the TZ environment variable, even if you have the TZ environment variable configured in /etc/profile.

  An alternative to this procedure is to insert a time zone variable into the CEEPRM member of SYS1.PARMLIB. This sets a common time zone for nearly all UNIX processes without having to code the TZ variable individually for each process.

Note: Most z/OS UNIX applications that start as MVS started tasks cannot use environment variables that have been configured in /etc/profile.

Before we started PAGENT, we defined it with the correct security authorizations, as described next.

Defining the security product authorization for PAGENT

Because the Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the Policy Agent from a z/OS procedure library.

To set up the security definitions for PAGENT, perform the following steps:

1. Define the PAGENT started task to RACF.
2. Define a user ID for the PAGENT started task.
3. Associate this user ID with the PAGENT started task.
4. Give authorized users access to start and stop PAGENT.
5. Restrict access to the pasearch command to authorized users.
6. Set up TTLS Stack Initialization access control.

Define the PAGENT started task to RACF

To set up a started task, you have to define a profile for it in the resource class called STARTED. First, you have to activate this class if it is not already active. This resource class is RACLISted so that the profiles are kept in RACF data space for improved performance. It is also defined as a GENERIC class to allow generic profiles to be created in this class for more efficient searches.

In most installations, this may already have been done; therefore, you might not have to issue commands to define the STARTED class RACLIST and GENERIC. We simply mention it here for completeness.

You must specify two qualifiers for the profile names in STARTED class. We defined PAGENT.*, and then we refreshed RACLIST and GENLIST to update the in-storage profiles with this new information. Example 5-5 shows the RACF commands for this.

Example 5-5 Define the PAGENT started task to RACF

```
SETROPTS CLASSACT(STARTED)
SETROPTS RACLIST(STARTED)
SETROPTS GENERIC(STARTED)
RDEFINE  STARTED  PAGENT.*
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED)  REFRESH
```

Define a user ID for the PAGENT started task

We defined a PAGENT user ID with default group TCPGRP and with an OMVS segment.

This user ID has to be defined with UID=0. But only one user ID can have UID=0 in the system, and it is normally already assigned to user BPXROOT in most installations. Therefore, you have use the SHARED parameter in the definition. A home directory is also assigned to this user ID. Example 5-6 shows the command that we used.

Example 5-6 Define a user ID for the PAGENT started task

```
ADDUSER  PAGENT DFLTGRP(TCPGRP) OMVS(UID(0) SHARED HOME('/'))
```

Associate this user ID with the PAGENT started task

We used the RALTER command to associate the PAGENT user ID and its group TCPGRP to the PAGENT started task. RACF stores this information in the STDATA field of the profile. Example 5-7 shows the command that we used.

Example 5-7 Associate user ID with the PAGENT started task

```
RALTER STARTED PAGENT.* STDATA(USER(PAGENT) GROUP(TCPGRP))
```

Give authorized users access to start and stop PAGENT

To control which users can start PAGENT and thus reduce the risk of an unauthorized user starting and affecting policy-based networking, we define a profile named MVS.SERVMGR.PAGENT in resource class OPERCMDS and give authorized users access to this facility. Activate the OPERCMDS class and RACLIST it, if not already done in your installation. Example 5-8 shows the commands that we used.

Example 5-8 Give authorized users access to start and stop PAGENT

```
SETROPTS CLASSACT(OPERCMDS)
SETROPTS RACLIST (OPERCMDS)
RDEFINE  OPERCMDS (MVS.SERVMGR.PAGENT) UACC(NONE)
PERMIT  MVS.SERVMGR.PAGENT CLASS(OPERCMDS) ACCESS(CONTROL) ID(PAGENT,CSO1,CSO2)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Restrict access to the pasearch command to authorized users

The **pasearch** command is used to obtain details of the security policies on your system. This is a very sensitive command and needs to be protected. The profile to protect this resource is of the form EZB.PAGENT.sysname.tcpprocname.*, where:

EZB	Constant
PAGENT	Constant for this resource type
sysname	The system name
tcpprocname	The TCP/IP proc name
*	For all policy type options

The profile is defined in the SERVAUTH class. Example 5-9 shows the commands that we used.

Example 5-9 Restrict access to the pasearch command to authorized users

```
RDEFINE  SERVAUTH EZB.PAGENT.SC30.TCPIPA.* UACC(NONE)
PERMIT   EZB.PAGENT.SC30.TCPIPA.* CLASS(SERVAUTH) ID(PAGENT,CSO1,CSO2) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

Set up TTLS Stack Initialization access control

If you are using Application Transparent Transport Layer Security (AT-TLS), z/OS will not allow any socket-based applications to start before PAGENT is up and running so as to make sure that all the security policies are enforced. But some essential applications need to start before PAGENT.

To allow this, you have to define a resource profile EZB.INITSTACK.sysname.tcpprocname in the SERVAUTH class. The resource name consists of the following parts:

EZB	Constant
INITSTACK	Constant for this resource type
sysname	The system name
tcpprocname	The TCP/IP proc name

The RACF commands that we used for this are shown in Example 5-10.

Example 5-10 Set up TTLS Stack Initialization access control

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.SC30.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC30.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
         WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

5.2.2 Starting PAGENT from UNIX

The PAGENT executable resides in /usr/lpp/tcpip/sbin. There is also a link from /usr/sbin. Make sure that your PATH statement contains either /usr/sbin or /usr/lpp/tcpip/sbin. To start PAGENT in the z/OS UNIX System Services shell, issue the following command:

```
pagent -c /etc/pagent.sc30.conf -l SYSLOGD &
```

The Policy Agent uses the configuration file /etc/pagent.sc30.conf and logs output to the syslog daemon (SYSLOGD). To run PAGENT in the background, the start command is suffixed with an ampersand (&).

Refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, to resolve any EZZ errors encountered at PAGENT startup time.

5.2.3 Stopping PAGENT

You can stop the Policy Agent as follows:

- By using the operator command P PAGENT from System Display and Search Facility (SDSF) or the system console.
- By using the **kill** command in the z/OS UNIX shell. Example 5-11 shows how to find the process ID for PAGENT, which is then killed with the **kill -s** command. The pid can also be found in /tmp/pagent.pid.

Example 5-11 Stopping PAGENT from UNIX

```
CS02 @ SC30:/u/cs02>ps -ef | grep PAGENT
 BPXROOT   16842831   83951672  - 16:00:27 ttyp0001  0:00 grep PAGENT
 BPXROOT   67174676         1  -   Oct 14 ?         1:13 PAGENT
CS02 @ SC30:/u/cs02>kill -s TERM 67174676
CS02 @ SC30:/u/cs02>ps -ef | grep PAGENT
 BPXROOT   33620047   83951672  - 16:01:10 ttyp0001  0:00 grep PAGENT
CS02 @ SC30:/u/cs02>
```

When the Policy Agent is shut down normally (using KILL or STOP), if the PURGE option is configured, then all QoS, IDS, and AT-TLS policies are purged from this stack. IPSec and PBR policies are not automatically purged.

5.2.4 Disabling PAGENT policies for IPSec

PAGENT policies can be disabled by using the following command:

```
ipsec -f default
```

This reverts back to the TCP/IP configuration file policies. To convert back to PAGENT policies, issue the following command:

```
ipsec -f reload
```

5.2.5 Basic configuration

Beyond the RACF and SyslogD prerequisites for Policy Agent, the configuration and policy definitions require at a minimum:

- Definition of the MVS procedure or the UNIX process to start Policy Agent
- Definition of a Policy Environment file
- Definition of a Policy Agent Configuration File, which can also include policy definitions

However, you may also organize the Policy Agent configuration and policy definitions as follows:

- Definition of the MVS procedure or the UNIX process to start Policy Agent.
- Definition of a Policy Environment file.
- Definition of a Policy Agent Configuration File.
  - This file points to TCP “image files” for each TCP/IP instance.
  - Each “image file” can point to one or more “policy type” files.

An illustration of the Definition of a Policy Agent Configuration File approach is in Figure 5-6.

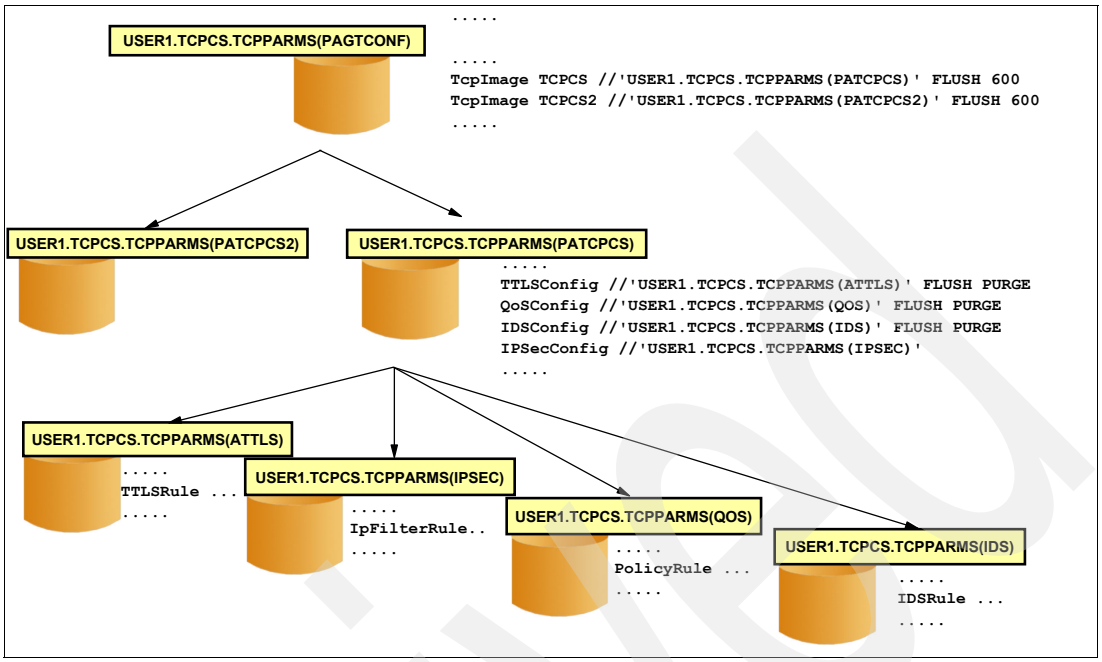


Figure 5-6 Structuring Policy Agent configuration and policy files

Figure 5-6 illustrates the main policy configuration file. There are two TCP/IP images (stacks or instances) that will not share a single policy file. The main configuration file points to two distinct image files which have been built for the stacks by using the TCPIMAGE statement:

- TCPPCS2 uses an image file named PATCPCS2, with its policies imbedded within it.
- TCPCS uses an image file named PATCPCS that points to multiple “policy type” files. We identify the multiple policy type files with statements like TTLSCONFIG, QOSCONFIG, and so on. These individual policy type files are depicted at the bottom of Figure 5-6.

We examine the top-level configuration file first: the Policy Agent configuration file. Before defining policies, some basic operational characteristics of the Policy Agent need to be configured in the PAGENT configuration file. In this section, we explain the following configuration steps:

- Defining the TcpImage statements
- Defining the appropriate logging level

Defining the TcpImage statements

The Policy Agent can be configured to install policies on one or more TCP/IP stacks or images. Each TCP/IP stack is configured using a TcpImage statement in the main configuration file. A secondary configuration file can be defined for each stack; a set of stacks can share configuration information in the main configuration file; or a combination of these techniques can be used.

To install different sets of policies to different stacks, configure each image with a different secondary configuration file. In this case, each image can be configured with a different policy refresh interval, if required. The refresh interval used for the main configuration file will be the smallest of the values specified for the different stacks.

In Figure 5-7, we show PAGENT's configuration file identifying, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed and the different configuration files that should be used by each.

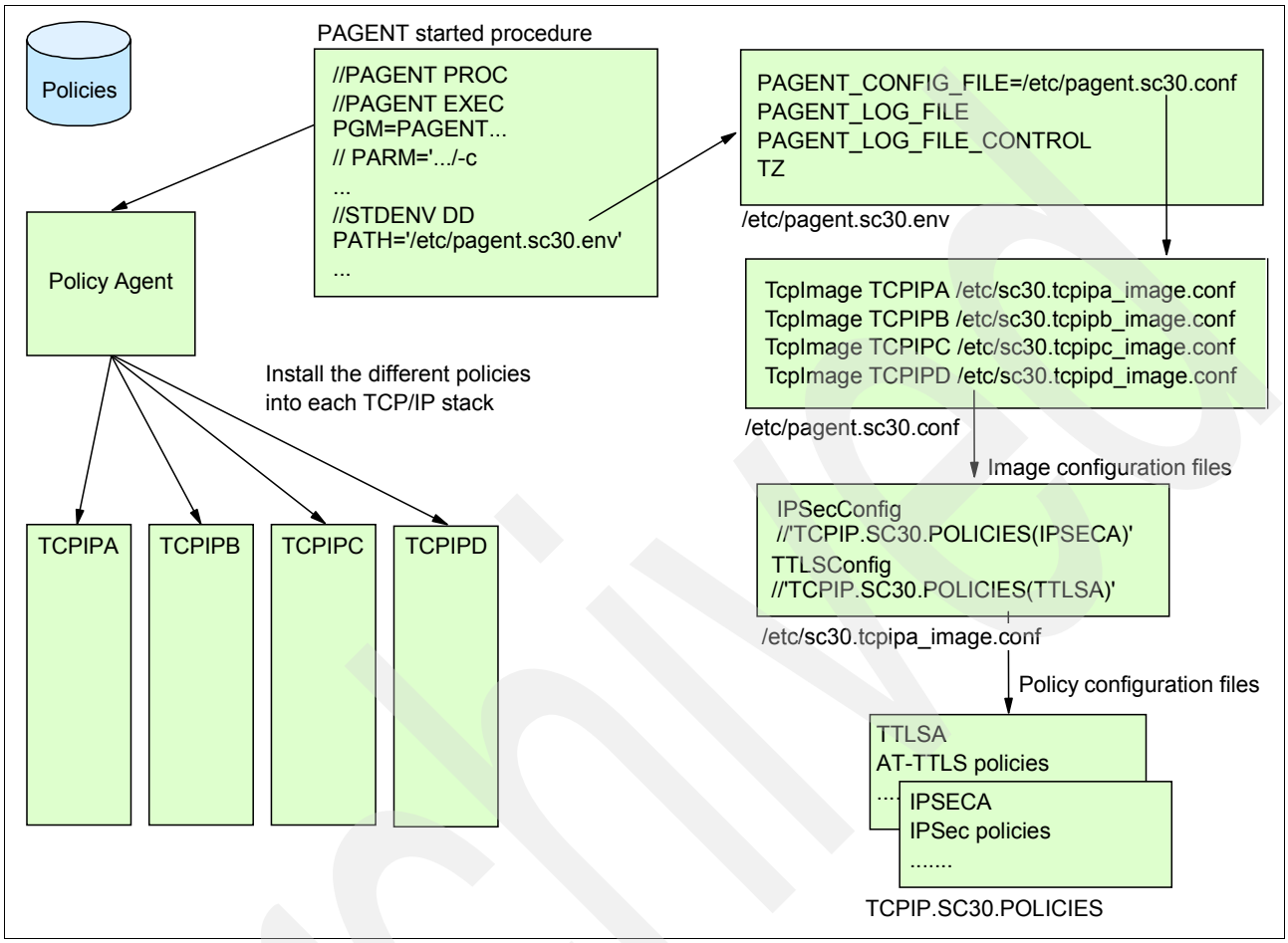


Figure 5-7 Multiple stacks, multiple policy definitions

Note: When the main configuration file is an MVS data set, it is reread at each refresh interval (which is the smallest of the individual stack refresh intervals), regardless of whether it has actually been changed. Because PAGENT restarts all stack-related processing when the main configuration file is reread, this effectively makes the refresh interval for all stacks the same as this smallest configured interval.

To install a common set of policies to a set of stacks on the same LPAR, do not specify secondary configuration files for each image. In this case, there is only one configuration file (the main one), and the policy information contained in it is installed to all of the configured stacks. Different refresh intervals can also be configured for each image, but that might not be useful in this case.

In Figure 5-8 on page 237, we show PAGENT's configuration file identifying, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed—but in this case, installing the same policies into each.

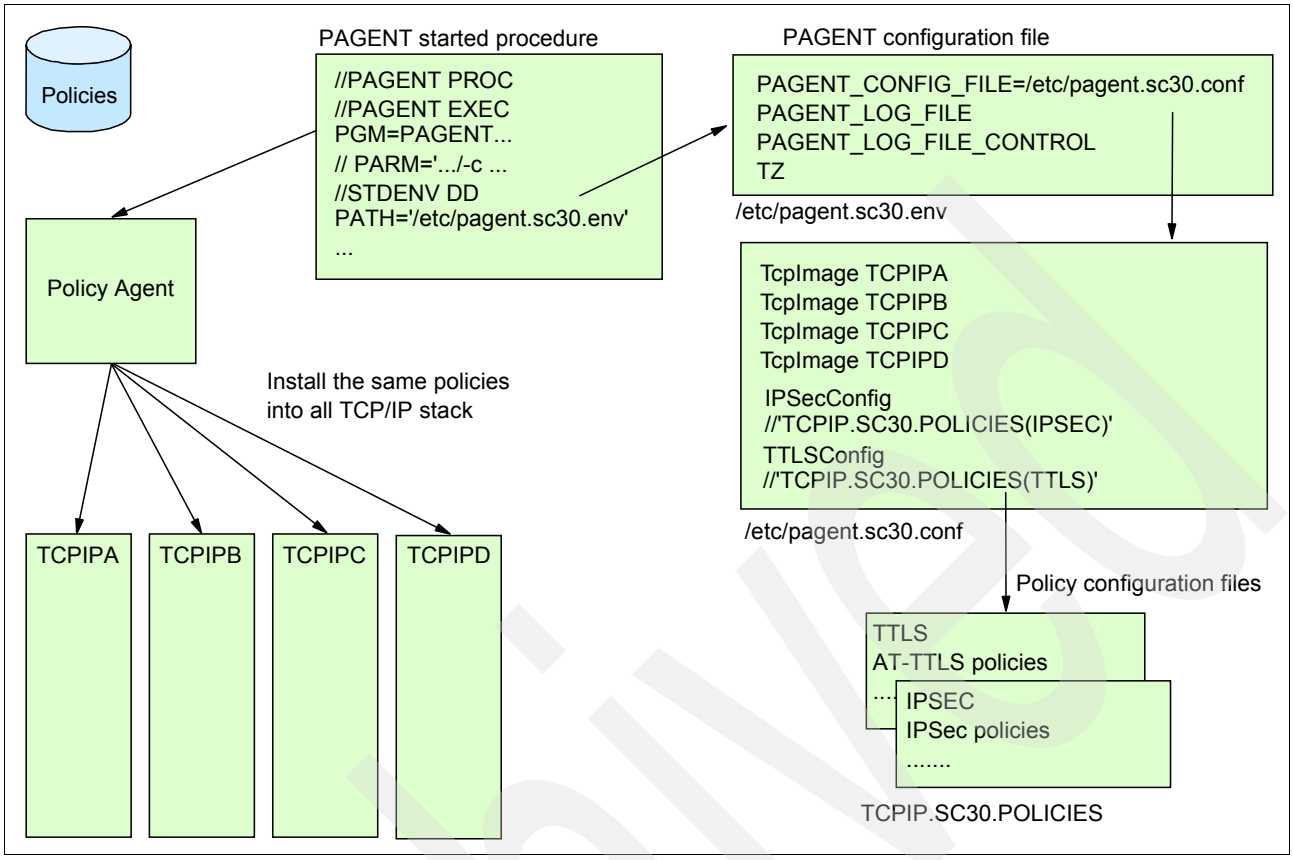


Figure 5-8 Multiple stacks, single policy definition

It is possible that TCP/IP stacks that are configured to the Policy Agent are not started or even defined. The Policy Agent will fail when trying to connect to those stacks and log appropriate error messages.

The Policy Agent does not end when any (or all) stacks end. When the stacks are restarted, active policies are automatically reinstalled. When the Policy Agent is shut down normally (that is, using KILL or STOP commands), and the TcpImage statement option PURGE is coded, all policies will be purged from this stack. The TcpImage statement specifies a TCP/IP image and its associated configuration file to be installed to that image. The following example installs the policy control file /etc/pagent.sc30.conf to the TCPIPA TCP/IP image, after flushing the existing policy control data:

```
TcpImage TCPIPA /etc/pagent.sc30.conf FLUSH
```

Defining the appropriate logging level

The LogLevel statement is used to define the amount of information to be logged by the Policy Agent. The default is to log only event, error, console, and warning messages. This might be appropriate for a stable policy configuration, but more information might be required to understand policy processing or to debug problems when first setting up policies or when making significant changes. Specify the LogLevel statement with the appropriate logging level in the main configuration file.

Note: The maximum logging level (511) can produce a significant amount of output, especially with large LDAP configurations. This is not of concern if z/OS UNIX log files are used, because Policy Agent will round-robin (circulate) a set of finite size files. (The environmental variable PAGENT_LOG_FILE_CONTROL identifies the number and size of these files.) However, when using the syslog daemon as the log file, the amount of log output produced should be taken into consideration.

Considerations when defining policy rules

When you define and code the policy rules direction, source, and destination, consider when policy rules are applied:

- For TCP, the policies are applied at TCP connection setup.
- For UDP, a policy rule is applied every time a UDP datagram is being received or sent.
- For other protocols (such as ICMP, Open Shortest Path First (OSPF), and so on), every time an IP datagram is being received or sent, the policy rules are applied.
- The policies are remapped when the policy definitions are being updated or refreshed. The rules are remapped for every ACK segment in a TCP flow to adjust for time-of-day-related policies.

5.2.6 Coding policy definitions in a configuration file

The configuration shown in Example 5-12 is based upon the “Multiple stacks, multiple policy definitions” scenario shown in Figure 5-7 on page 236. In this scenario, the policy definitions have been configured in the PAGENT configuration file, and we use a two-level PAGENT configuration file to define the policy in a multiple IP stacks environment.

Example 5-12 PAGENT configuration file

```
#
#  IBM Communications Server for z/OS
#  SMP/E distribution path: /usr/lpp/tcpip/samples/IBM/EZAPAGCO
#
#  Licensed Materials - Property of IBM
#  5694-A01
#  (C) Copyright IBM Corp. 1998, 2005
#  Status = CSV1R9
# LogLevel Statement
Loglevel 255 1
# TcpImage Statement 2
TcpImage TCPIPA /etc/sc30.tcpipa_image.conf FLUSH
TcpImage TCPIPB /etc/sc30.tcpipb_image.conf FLUSH
TcpImage TCPIPC /etc/sc30.tcpipc_image.conf FLUSH
```

The log level 1 is set with the integer that specified the level of logging/tracing. We used LogLevel 255, which means all messages except trace messages are captured. The supported levels are:

- 1 - SYSERR - System error messages
- 2 - OBJERR - Object error messages
- 4 - PROTERR - Protocol error messages
- 16 - EVENT - Event messages
- 32 - ACTION - Action messages
- 64 - INFO - Informational messages
- 128 - ACNTING - Accounting messages
- 256 - TRACE - Trace messages

The TcpImage statement 2 defined the TCP/IP stacks to be policed. Up to five parameters can be configured for this statement, as described here:

- The first parameter specifies the TCP/IP stack name on which the policy must be installed (TCPIPA, TCPIPB, and TCPIPC).
- The second parameter is the path of the image configuration file for the associated TCP/IP stack.
- The third parameter allows you to specify whether the Policy Agent deletes all the policies existing in the TCP/IP stack when it is started, with FLUSH specified.

> **Note:** The policies installed in the TCP/IP stack will be deleted at PAGENT startup time *only* if the FLUSH parameter is specified. This prevents the policies from being deleted unexpectedly if PAGENT terminates abnormally.

- The fourth parameter is used when you want to remove policies when you cancel PAGENT. You can restart PAGENT afterward, pointing to a configuration file but no policies defined, with PURGE specified.
- The fifth and last TcpImage statement parameter (not specified in our example) specifies the time interval, in seconds, for checking the creation or modification time of the configuration files and for refreshing policies from the LDAP server. The default value is 1800 seconds (30 minutes).

> **Note:** Dynamic monitoring for the configuration file is only supported for z/OS UNIX files. MVS data sets are not monitored for changes.

Policy Agent log file

When you start the Policy Agent as a started task, the output messages are written to the log file, which can be specified by the PAGENT_LOG_FILE environment variable and defaults to /tmp/pagent.sc30.log. The log file is created when the Policy Agent is activated, if it does not already exist.

5.2.7 Refreshing policies

There are two commands used to refresh policies in PAGENT:

- The F PAGENT,REFRESH command triggers Policy Agent to reread its config files and, if requested, downloads policy objects from the LDAP server. If the FLUSH parameter is specified on the TcpImage configuration statement, policy statistics being collected in the TCP/IP stack are reset, because FLUSH deletes and reinstalls all policies.
- The F PAGENT,UPDATE command is different from the REFRESH command, because PAGENT only installs or removes from the stack (as appropriate) any new, changed, or deleted policies.

> **Recommendation:** The UPDATE command should be used in most cases. The REFRESH command should only be used if you suspect that policies have somehow become out of sync between the TCP/IP stack and Policy Agent.

5.2.8 Verification

Use the z/OS UNIX **pasearch** command to query information from the z/OS UNIX Policy Agent. The command is issued from the UNIX System Services shell. We used the **pasearch** command to display all the policy entries for our TCP/IP stack named TCPIPA using the following command:

```
pasearch -p TCPIPA
```

The default is to return all policy entries for all TCP/IP stacks. The value used for TcpImage (in our example, TCPIPA) must match one of the values specified on the TcpImage statement in the Policy Agent configuration file.

5.2.9 For additional information

Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding Policy Agent.

5.3 Backup and migration considerations

Keep the following backup and migration considerations in mind regarding the Policy Agent.

5.3.1 Backup considerations

The backing store file

The backing store file or files should be maintained on a central server of some kind, whether a SAMBA server or an NFS server. The IBM Configuration Assistant for z/OS Communications Server contains a locking mechanism that prevents more than one administrator at a time from working with the file. This, however, does not prevent an administrator from creating a copy of the file and then restoring it on top of a file just updated by someone else. Strict control needs to be maintained over the backing store files so that one administrator does not accidentally overlay the work performed by a different administrator.

It is worthwhile to preserve the Java™ backing store files in a server that is backed up nightly. By default, the backing store file is stored on the workstation. If you save the backing store file on the mainframe, you can use MVS methods of backing up or archiving the file. The ASCII policy files can also be backed up using standard dump and restore procedures. However, with this version of the IBM Configuration Assistant, it is not possible to convert the ASCII file to a backing store file.

At this level of the IBM Configuration Assistant code, after you decide to start editing the ASCII policy files manually, you must continue to do so unless the changes have been minor (such as changing the trace levels of the processes). The ability to move between manual editing and GUI editing is a known requirement.

After the FTP of the policy file to the mainframe you see the comment cards, shown in Example 5-13, which have been added to the policy file; the comments show the location of the backing store file (a).

Example 5-13 Location of backing store file and FTP History

```
********************************* Top of Data ********************************
##
## AT-TLS Policy Agent Configuration file for:
```

```
##    Image: LPARA29
##    Stack: TCPIPD
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R9\files\LPARA29.TTLS a
## FTP History:
## 2007-10-01 10:28:04  CSO2 to 10.1.1.40
## 2007-09-26 08:26:26  CSO2 to 10.1.1.40
##   This version contains only FTP ATTLS policy
##
```

The backing store file in Example 5-13 on page 240 has been stored on the workstation. The backing store file can also be stored on the mainframe. If you have performed the following three setup steps, the backing store file can also automatically be stored on MVS during the ftp of the policy file:

1. On IBM Configuration Assistant, set up the Preferences panel to store the file on the mainframe in either an HFS or zFS, or in a partitioned data set.
2. Select the **File --> Save As** pull-down on IBM Configuration Assistant and establish a mainframe location and name for the binary Java object backing store file.
3. When you execute the Install option on Configuration Assistant to ftp the policy file to the mainframe target, select the check box that allows you to automatically save the backing store after the FTP transfer completes.

The examples that follow show you how to do this on IBM Configuration Assistant.

As shown in Figure 5-9 on page 242, you first see the file pull-down with the Preferences... selection highlighted.

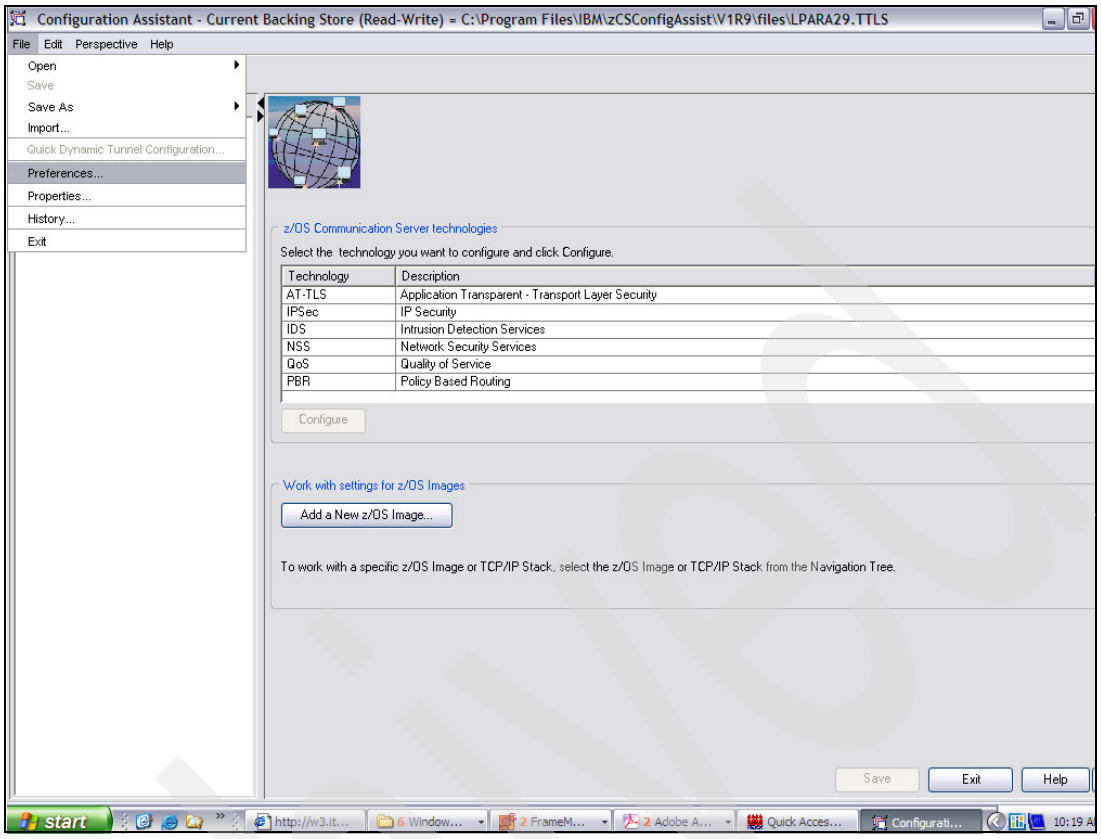


Figure 5-9 Setting preferences to save the backing store file

Pressing **Preferences...** moves you to Figure 5-10, which shows how we want to FTP the policy file to MVS DASD for safekeeping.

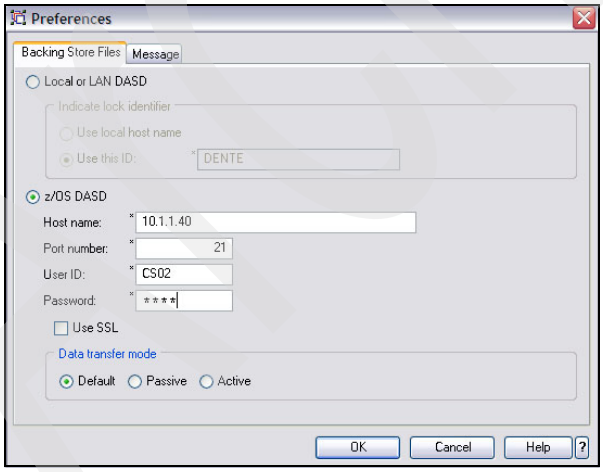


Figure 5-10 Specifying MVS location for saving the backing store file

Press **OK** and return to one of the Perspective panels of IBM Configuration Assistant.

Next, select the image file that you want saved in a backing store file at the mainframe and use the **`File --> Save As`** pull-down to provide the path for the mainframe backup; see Figure 5-11 on page 243.

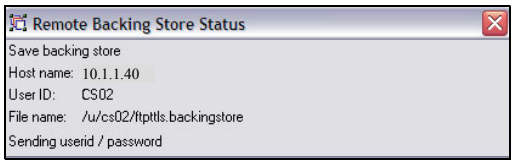


Figure 5-11 Specifying path for mainframe location of backing store file

In the final step, you FTP the policy file to the mainframe. If you have indicated that the backing store file should also be sent to the mainframe, then this step occurs at the same time.

Note: The policy files and backing store files cannot be FTPed if a lock is on the file and the file is in Read-only mode. A warning message appears in this instance and you have the opportunity to release the lock by deleting the file ending in .LCK.

For example, if there is a lock on a file named file /u/cs02/FTPandTN3270.backingstore, you will see that another file exists named /u/cs02/FTPandTN3270.backingstore.LCK.

If different personnel are responsible for different types of policies, you can maintain multiple backing store files, each named differently. If the decision is made to merge them, you can do so using the Import utility on the IBM Configuration Assistant. Merging separately maintained files is described in “Merging (Importing) backing store files” on page 243.

Note: The point is to maintain backups of critical files, like the backing store file and various configuration and policy files.

Role of the Centralized Policy Server

The Centralized Policy Server can simplify policy management and administration. Instead of using local policy files for each of multiple TCP/IP images, you can reduce the number of individual files needing to be backed up by storing the policy files themselves directly on a Centralized Policy Server. The files backed up at the Policy Server represent the bulk of the files required for a PAGENT implementation, thus limiting the scope of the backup services required. The subject of Centralized (or Distributed) Policy Services is discussed in Chapter 6, “Central Policy Server” on page 257.

Merging (Importing) backing store files

For various reasons, you may have multiple backing store files that contain different policy configurations. For example, you may have an IDS administrator who keeps a version of the backing store file separate from the one containing QoS policies. During testing you may have created separate AT-TLS policies for individual applications, like CICS sockets, FTP, and TN3270. There may come a time when you want to merge these separate backing store files.

Rule: To merge backing store files, ensure that the image names and TCP/IP stack names are consistent across all copies of the backing store file.

Explanation: Backing store files are easy to merge if all policy administrators have agreed to name their MVS image the same within the IBM Configuration Assistant.

For example, if two administrators are both working on policies for a stack named TCPIPD, and one administrator names their MVS image SC33 and the other names their MVS image LPARA29, the backing store files are merged as two separate images on the IBM Configuration Assistant Main Perspective as follows:

- Image SC33
  - TCPIPD
    - AT-TLS for TN3270
- Image LPARA29
  - TCPIPD
    - AT-TLS for FTP

If the image names are the same, then the same backing store files are merged as follows:

- Image SC33
  - TCPIPD
    - AT-TLS for TN3270
    - AT-TLS for FTP

This rule states that the image names and TCP/IP stack names should be consistent between files that are being merged. Consistency makes it simple to merge backing store files. In our case, we made the mistake of not providing that consistency and had to execute additional steps in order to merge several policy files. So here we explain how to handle a merge if the image and stack names are inconsistent. Next, we fix our problem and show how easy a merge is if the image and stack names are consistent.

Inconsistent backing store files: how to merge them

We began with the Main Perspective of our existing backing store file; see Figure 5-12 on page 245.

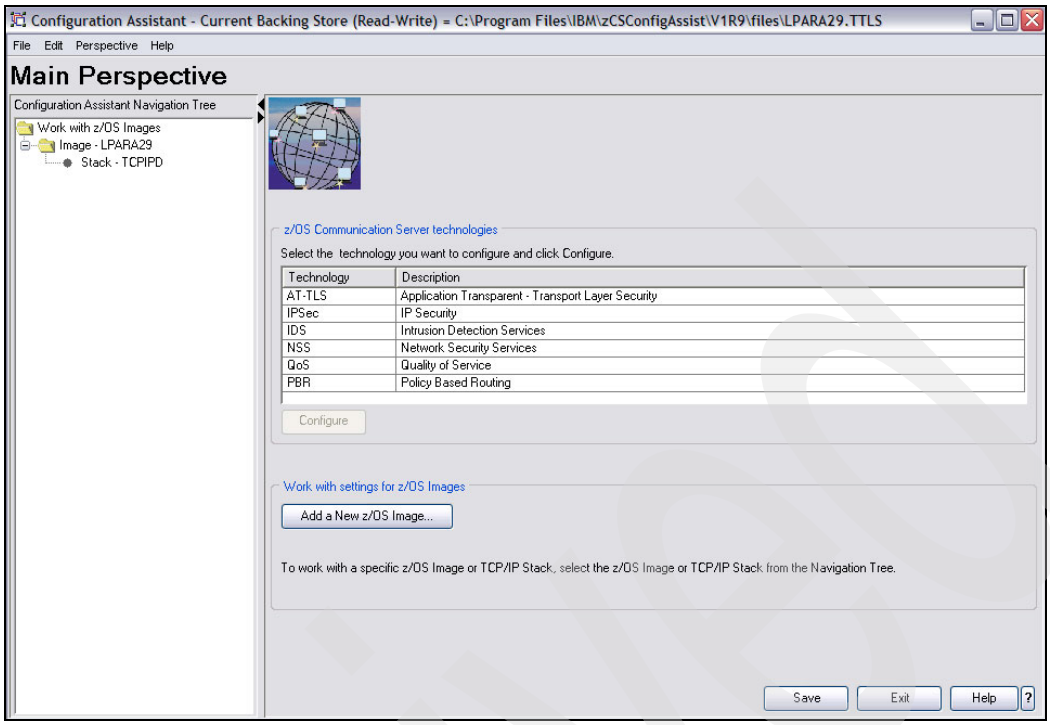


Figure 5-12 Main Perspective of backing store file on workstation

We selected **`File --> Import`** from the file pull-down and were presented with the panel shown in Figure 5-13.

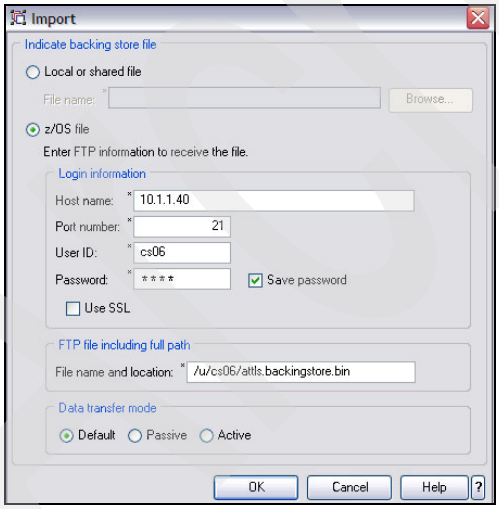


Figure 5-13 Storing the backing store file

We pressed **OK** and received a Remote backing store status panel, which indicated the progress of the retrieval from the backing store repository. Finally, we saw that the import process was complete, as shown in Figure 5-14 on page 246.

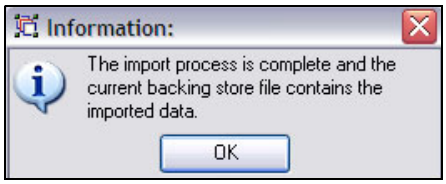


Figure 5-14 Completed import process

Next, we clicked **OK**. Our Main Perspective, as shown in Figure 5-15, indicated that we had not quite accomplished what we wanted. Because our imported backing store file used a different name for the MVS image, the IBM Configuration Assistant now considered this to be a completely separate MVS image and imported it as such into the single backing store file.

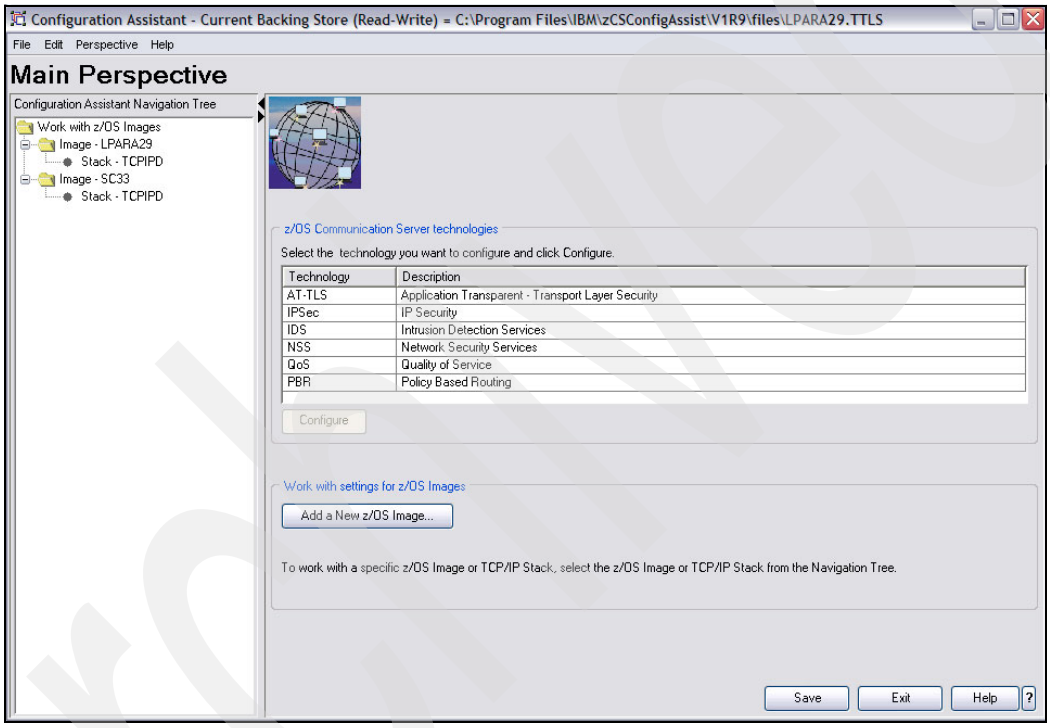


Figure 5-15 Unsuccessful import due to inconsistent image names

The Main Perspective showed that one image was named LPARA29 and the other image was named SC33. In reality, we had intended for both of those images to be the same image: SC33. Fortunately, this situation is easy to fix and to merge properly.

We decided as a group that our image names would be the system names in our sysplex. Therefore, for this LPAR A29 the image name will be SC33. The names of our TCP/IP stacks were already consistent.

Next, we highlighted the imported file that was named SC33 and right-clicked it. The menu that appeared offered the option to delete the image, which we selected. We were presented with the warning shown in Figure 5-16 on page 247.

Figure 5-16 Warning message - Delete the z/OS Image

When we responded with `Yes`, the image we had imported was removed from the current backing store file. We renamed the existing image from LPARA29 to SC33 by keying over the image name shown in Figure 5-17.

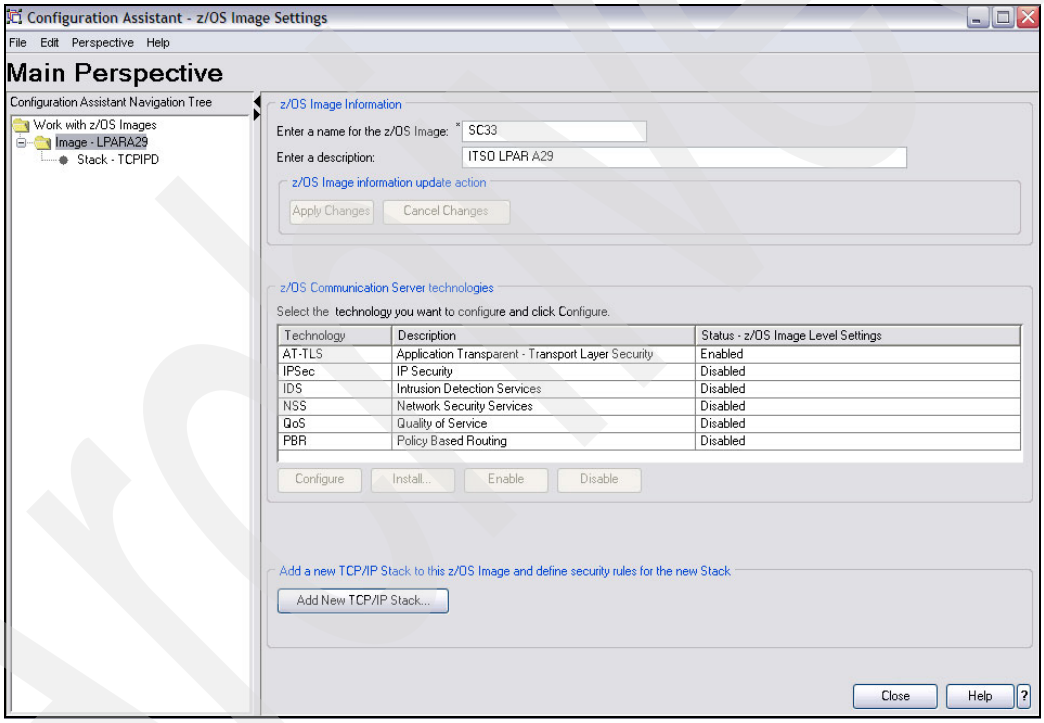


Figure 5-17 Renaming the backing store image file to a consistent value

When we pressed **Enter**, the Apply Changes button came into focus. We selected **Apply Changes**, and the Main Perspective changed the left column to show that the image name was changed to SC33; see Figure 5-18 on page 248.

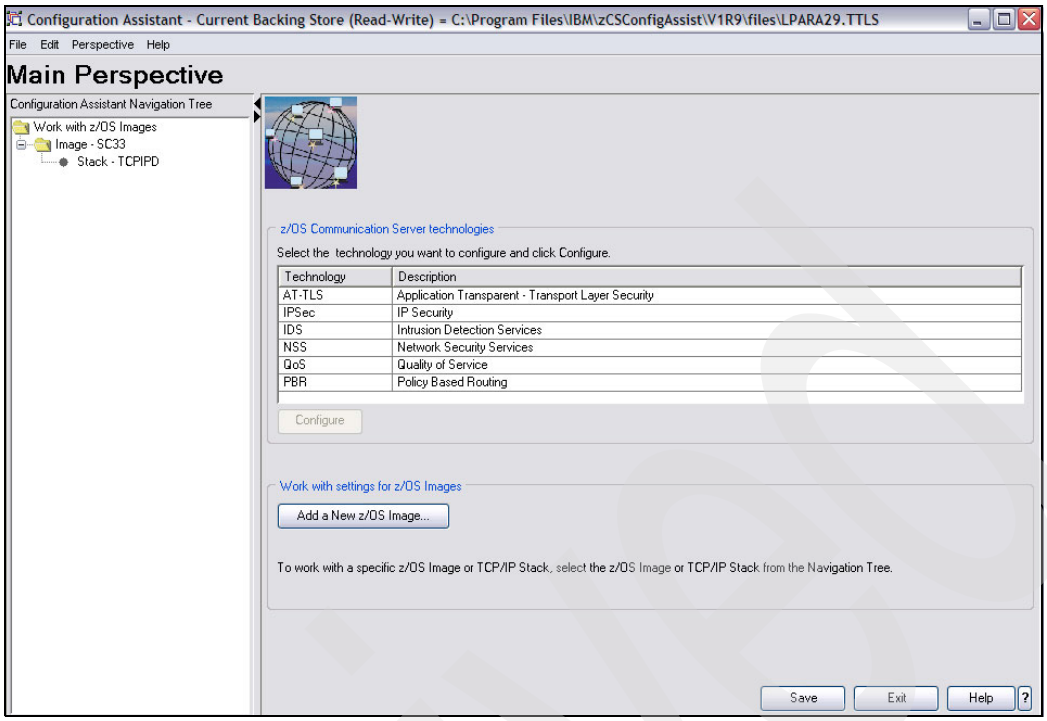


Figure 5-18 Main Perspective before renaming the image file

We saved this new version under a different name: FTPandTN3270.TTLS.backingstore by using **`File --> Save As`**. We identified the host location in the panel shown in Figure 5-19.

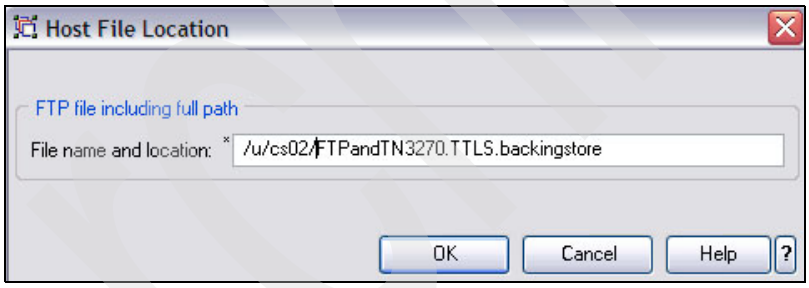


Figure 5-19 Saving a new backing store file at the host

Next we imported the backing store file with the TN3270 AT-TLS definitions in it as we did before. We selected **`File --> Import`** and named the file located at the mainframe. When the import was completed, the Main Perspective looked as shown in Figure 5-20 on page 249.

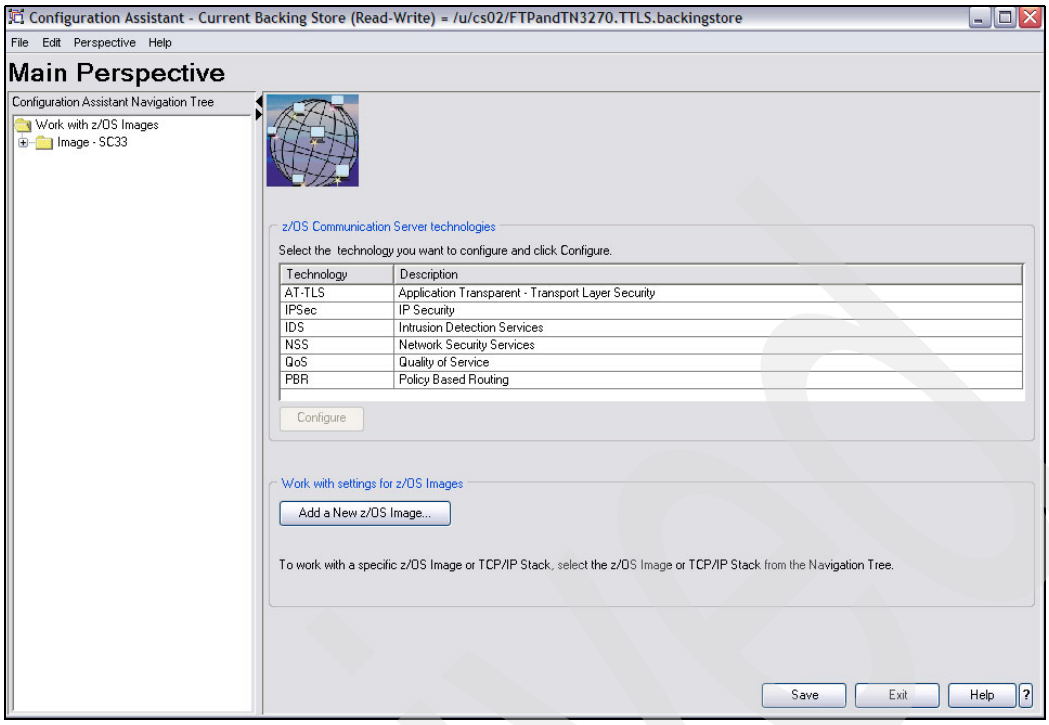


Figure 5-20 Main Perspective after the import and merge

The Import Activity Summary Report, shown in Figure 5-21, reveals more than what is visible here. It tells us that many more of the reusable objects from the merged files have been consolidated instead of repeated. We closed the report in order to verify, with a look at the Connectivity Rules, that we really did have both FTP and TN3270 rules combined.

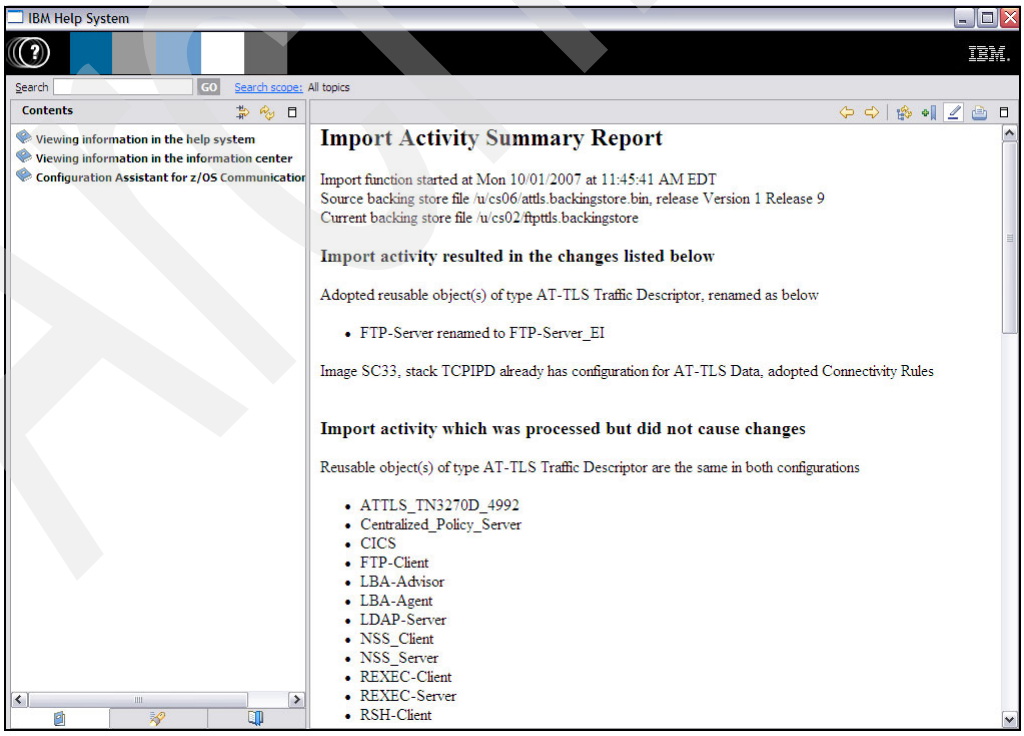


Figure 5-21 Results of Import Activity Summary Report

Therefore, we selected the AT-TLS Perspective to verify the merged AT-TLS definitions. We highlighted the TCPIPD stack definitions in the left column. Initially, our panel looked as shown in Figure 5-22.

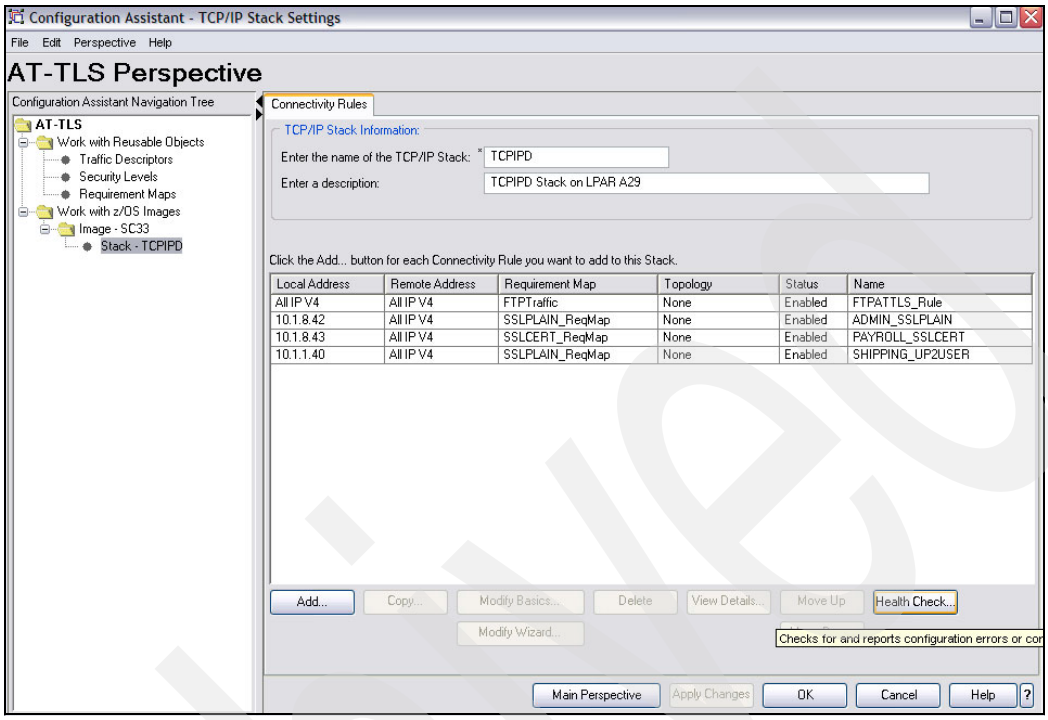


Figure 5-22 Initial AT-TLS Perspective

It showed that we did indeed import both FTP and TN3270 definitions. We saved the new backing store file with **`File --> Save As`**.

Our final step was to FTP the merged file into the mainframe and to point our PAGENT policy files to the new file. We FTPed to the mainframe and named the file as shown in Figure 5-23.

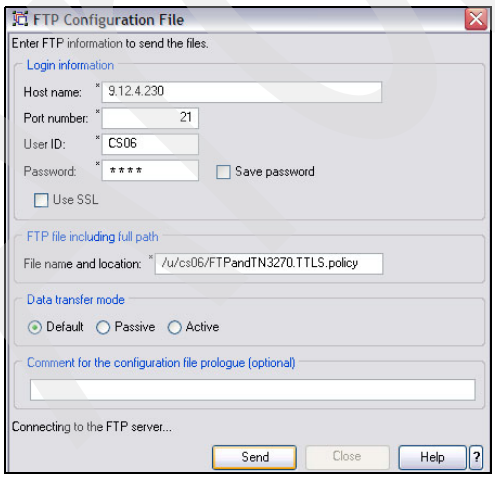


Figure 5-23 FTPing the merged AT-TLS policy file to MVS

If there is a lock on the backing store file, you may receive an error message indicating that the file cannot be saved at the host or on the workstation. Figure 5-24 on page 251 displays such a message.

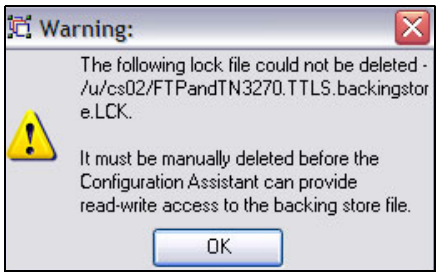


Figure 5-24 Error message if lock is still on a backing store file

In such a case, you must delete the lock file at the mainframe or on the workstation. We logged into the mainframe in this case and saw that there was indeed a lock on the file, as shown by a in Example 5-14.

Example 5-14 File is locked

```
-rw-r-----   1 CS02 SYS1       30234 Oct  1 11:43 FTPandTN3270.TTLS.backingstore
-rw-r-----   1 CS02 SYS1         162 Oct  1 11:10 FTPandTN3270.TTLS.backingstore.LCK a
```

We deleted the lock using the following command:

```
rm FTPandTN3270.TTLS.backingstore.LCK
```

Then we FTPed the file again successfully from IBM Configuration Assistant.

Next, we navigated to **File --> History** and examined at the history of the file transfers from this Configuration Assistant node. We also saw the history of the names of the backing store files, whether stored on the workstation or on the mainframe; see Figure 5-25.

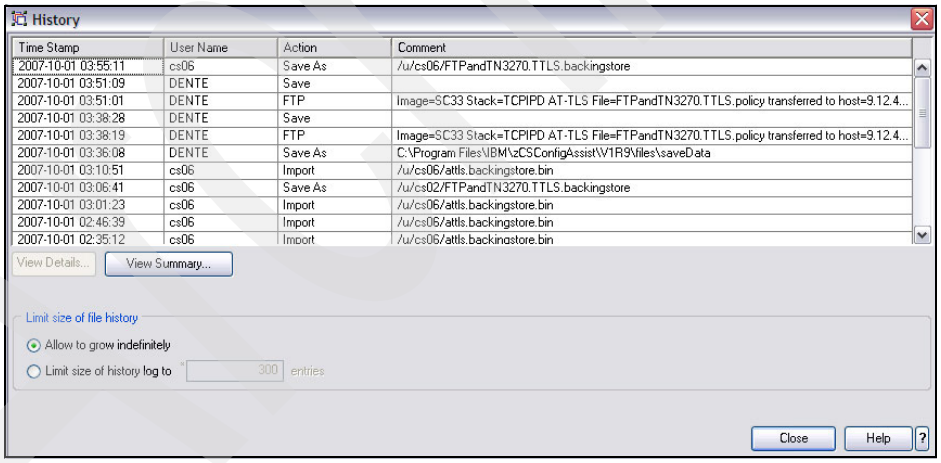


Figure 5-25 History of file transfers and names of backing store files

The View Summary... button allows you to view this information in another format.

Example 5-15 shows our revised TCPIPD Image file at the mainframe; PAGENT could now read the merged AT-TLS policy file with these definitions.

Example 5-15 Our revised TCPIPD Image file with new definitions

```
RoutingConfig /etc/pagent33_PBR.conf
QoSConfig     /etc/pagent33_QoS.conf
TTLSConfig     /u/cs06/FTPandTN3270.TTLS.policy     FLUSH PURGE
```

Because our IBM Configuration Assistant was still open at this point and we were still using the new file we created, at the mainframe we saw that there was a lock on the file, as shown in Example 5-16.

Example 5-16 Backing store file is locked

```
-rw-r-----   1 CS06     SYS1       56510 Oct  1 15:59 FTPandTN3270.TTLS.backingstore
-rw-r-----   1 CS06     SYS1         162 Oct  1 15:58 FTPandTN3270.TTLS.backingstore.LCK
```

When we exited from the Configuration Assistant, the assistant provided us another opportunity to back up the backing store file. When we finally exited, the backing store file with the lock (FTPandTN3270.TTLS.backingstore.LCK) in the mainframe directory disappeared.

At the mainframe console, we refreshed PAGENT with the UPDATE option:

```
f pagent,update
```

This ensured that only the changed configuration files (and not all the policy files) were reread; see Example 5-17.

Example 5-17 Update looks only for changed configuration files

```
F PAGENT,UPDATE
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : NONE
CS02 @ SC33:/u/cs02>pasearch -t > ttlsoutput.merged
CS02 @ SC33:/u/cs02>
```

We executed, from the OMVS shell, the **pagent** command with the TTLS option to verify that all the policies were really loaded:

```
CS02 @ SC33:/u/cs02>pasearch -t > ttlsoutput.merged
```

We viewed the output in the redirected file (ttlsoutput.merged) and confirmed that all changes were successful. Then we tested both FTP and TELNET with AT-TLS successfully, as described in Chapter 15, “Secure File Transfer Protocol” on page 579 and Chapter 14, “Telnet security” on page 539.

Example 5-18 verifies that FTP with AT-TLS is still functioning within the merged AT-TLS policy file.

Example 5-18 FTP with AT-TLS is functioning within merged AT-TLS policy file

```
D TCPIP,TCPIPD,N,TTLS,CONN=6A8A
 EZD0101I NETSTAT CS V1R9 TCPIPD 939
 CONNID: 00006A8A
   JOBNAME:      FTPDD1
   LOCALSOCKET:  ::FFFF:10.1.1.40..21
   REMOTESOCKET: ::FFFF:10.1.100.224..3737
   SECLEVEL:     TLS VERSION 1
   CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
   CERTUSERID:   N/A
   MAPTYPE:      PRIMARY
 TTLSRULE: FTPATTLS_RULE~1
   TTLSGRPACTION:  ENABLEGRPACT~SC33
   TTLSENVACTION:  EACT1
   TTLSCONNACTION: CACT1~FTP-SERVER
```

```
  TTLSCONNACTION: CACT1~FTP-SERVER
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

5.3.2 Migration considerations

In this section we discuss various migrations considerations.

Migrating test configurations into production configurations

Note: You can choose one of several methods to test a new set of policy definitions.

For example, you might create a new backing store file for the new policy. Then, after testing with this file and its policy, you might make a backup copy of the production backing store file and merge the test backing store into the production copy using the Import function of the IBM Configuration Assistant. If the test with the merged production file is unsuccessful, you would still have a backup of the production backing store. The Import function is described in “Merging (Importing) backing store files” on page 243.

Another way to accomplish the integration of a test policy into a production set of policies does not require the IBM Configuration Assistant Import function. Simply copy the production backing store file into a backup file. Next, add the new definitions into the production backing store. Then test, and if the results are unsatisfactory, you still have the production backup to return to.

Migrating from previous releases of PAGENT or the GUI

Methods of migrating from previous releases are described here.

Migrating from an existing flat policy file

If you already have an existing flat policy file from a previous release, you can continue to use it. Simply add the new AT-TLS policy to the suite of policies already defined.

Migrating from an existing IBM Configuration Assistant policy file

If you already have an existing backing store policy file from a previous release, you can open the current IBM Configuration Assistant for z/OS Communications Server and, using the file pull-down, open the existing file. It will be upgraded to the new structure within the IBM Configuration Assistant. Then simply add the new AT-TLS policy to the suite of policies already defined. Figure 5-26 on page 254 illustrates the migration panel presented to you.

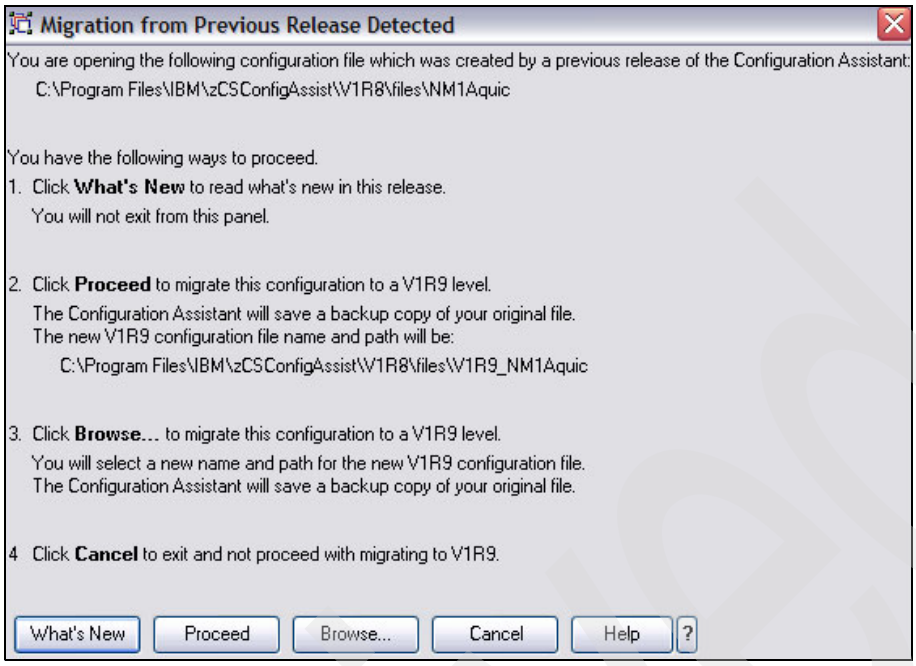


Figure 5-26 Migration from previous release

The panel provides the option of learning about changes in the IBM Configuration Assistant or the option of migrating the file to the new level.

5.4 Setting up Traffic Regulation Management Daemon

The Traffic Regulation Management Daemon (TRMD) can be viewed as a syslog daemon message writer. TRMD handles syslogd event recording for Intrusion Detection Services (IDS), which includes traffic regulation policies, and for IPSec services.

Setting up the started task procedure

A sample TRMD procedure can be found in TCPIP.SEZAINST(EZATRMDP). To associate the TRMD procedure with our TCP/IP job name, we set the RESOLVER_CONFIG environment variable to point to our TCPIP.DATA file, as shown in Example 5-19. TRMD can be started from the z/OS UNIX shell or as a started task.

Example 5-19 TRMD procedure parameters

```
//TRMD   EXEC PGM=EZATRMD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)',
//  'ENVAR("RESOLVER_CONFIG=//''TCPIP.TCPPARMS(DATAA30)''"',
//  '"LIBPATH=/usr/lib)"/-d 1')
```

To start TRMD as a started task, use the S TRMD command from the MVS console or SDSF. To automatically start TRMD when the TCP/IP stack is started, add TRMD to the AUTOLOG statement in the TCP/IP profile, as shown in Example 5-20.

Example 5-20 Autologging TRMD from TCP/IP

```
AUTOLOG
   TRMD JOBNAME TRMD
ENDAUTOLOG
```

Starting TRMD from z/OS UNIX

Only a superuser can run TRMD from the z/OS UNIX shell. Ensure that the following environment variables are correctly set before starting TRMD:

```
RESOLVER_CONFIG - To determine which stack TRMD will use
TZ - To ensure that the syslogd records are correctly timestamped
```

We set the environment variables 1, and started 2 and stopped 3 TRMD with the **kill** command, as shown in Example 5-21.

Example 5-21 Starting and stopping TRMD

```
CS02 @ SC30:/u/cs02>su
CS02 @ SC30:/u/cs02>export TZ="EST5EDT" 1
CS02 @ SC30:/u/cs02>echo $TZ
EST5EDT
CS02 @ SC30:/u/cs02>
CS02 @ SC30:/u/cs02>export RESOLVER_CONFIG="//'TCPIPA.TCPPARMS(DATAA30)'" 1
CS02 @ SC30:/u/cs02>echo $RESOLVER_CONFIG
//'TCPIPA.TCPPARMS(DATAA30)'
CS02 @ SC30:/u/cs02>
CS02 @ SC30:/u/cs02>trmd                                                    2
CS02 @ SC30:/u/cs02>ps -ef | grep trmd
 BPXROOT      65581   83951684  - 11:58:18 ttyp0001  0:00 grep trmd
 BPXROOT      65601          1  - 11:57:58 ttyp0001  0:00 trmd
CS02 @ SC30:/u/cs02>kill -s TERM 65601 3
CS02 @ SC30:/u/cs02>ps -ef | grep trmd
 BPXROOT   16842817   83951684  - 11:59:03 ttyp0001  0:00 grep trmd
CS02 @ SC30:/u/cs02>
```

Defining the security product authorization for TRMD

RACF is required to start the Policy Agent from a z/OS procedure library. Define a user ID for TRMD with a UID of 0 and define it to the RACF started class list, as shown in Example 5-22.

Example 5-22 RACF definitions for TRMD

```
//RACFDEF EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 ADDUSER  TRMD     DFLTGRP(TCPGRP)  OMVS(UID(0) SHARED HOME('/'))
 RDEFINE  STARTED  TRMD.* STDATA(USER(TRMD))
 SETROPTS RACLIST(STARTED) REFRESH
 SETROPTS GENERIC(STARTED) REFRESH
```

TRMDSTAT

TRMDSTAT is a utility that produces reports from IDS syslog records (summary and detailed). It reads a log file and analyzes the log records from TRMD. The following reports are available:

- Overall summary of logged connection events
- IDS summary of logged events
- Reports of logged connection events
- Reports of logged intrusions defined in the ATTACK policy
- Reports of logged intrusions defined in the TCP policy
- Reports of logged intrusions defined in the UDP policy
- Reports of statistics events

5.5 Additional information sources for PAGENT

For additional information about PAGENT, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782

Tip: For descriptions of security considerations that affect specific servers or components, read "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

6

Central Policy Server

In this chapter, we explain how a Policy Agent can become a Central Policy Server in order to provide a common repository for the policy files of Policy Agent clients. The centralized or distributed policy services are provided over connections between server and clients that are secured with AT-TLS. The distributed policy services provide a security mechanism based upon client login passwords or pass tickets.

The following topics are discussed in this chapter.

Section	Topic
6.2, “Basic concepts” on page 259	Basic concepts behind a centralized or distributed policy service
6.3, “Configuring distributed (centralized) policy services” on page 261	Implementation tasks for configuring the policy services at the server and the client nodes
6.4, “Activating and verifying the policy services environment” on page 288	Pertinent commands and messages used to verify correct operation of policy services
6.5, “Diagnosing the centralized policy services environment” on page 293	Pertinent displays and manuals that are helpful in diagnosing policy services problems
6.6, “References” on page 294	References to additional information to help you configure Policy Agent and Central Policy Services

6.1 Background

The Policy Agent is one of several components that provide a more general function known as *policy-based networking*. The primary components of the Policy Agent are:

- IBM Configuration Assistant for z/OS Communications Server - provides a GUI to define policies in flat files
- Policy Agent - manages policy definitions
- **pasearch** command - displays policy definitions
- The TCP/IP stack - enforces policy
- Sysplex Distributor - uses policy performance data to influence routing decisions
- IKE daemon - negotiates dynamic IP security associations with peers
- TRM daemon - logs events for some policy types
- **ipsec** command - displays system information and manages security associations
- nslapm2 SNMP subagent - monitors QoS policies according to the Networking SLAPM2 MIB

As discussed in Chapter 5, “Policy Agent” on page 221, policy-based networking is a way of accomplishing a set of network goals through the use of policy definitions. For example, one network goal may be to provide better quality of service (QoS) for one set of traffic as compared to another set. Policies can be defined to set the IPv4 type of service (TOS) or IPv6 traffic class for the two sets of traffic, to assist in obtaining the required quality of service from the network.

When the Policy Agent is started, the main configuration file is identified using a standard search order. This file in turn can point to one or more image configuration files using the TcpImage statement or PEPInstance statement.

In Figure 6-1, the main PAGENT configuration file points to two Image configuration files. The main configuration file can point to common files for all policy types *except* QoS: AT-TLS, IDS, IPSec and Routing or Policy-based routing (PBR).

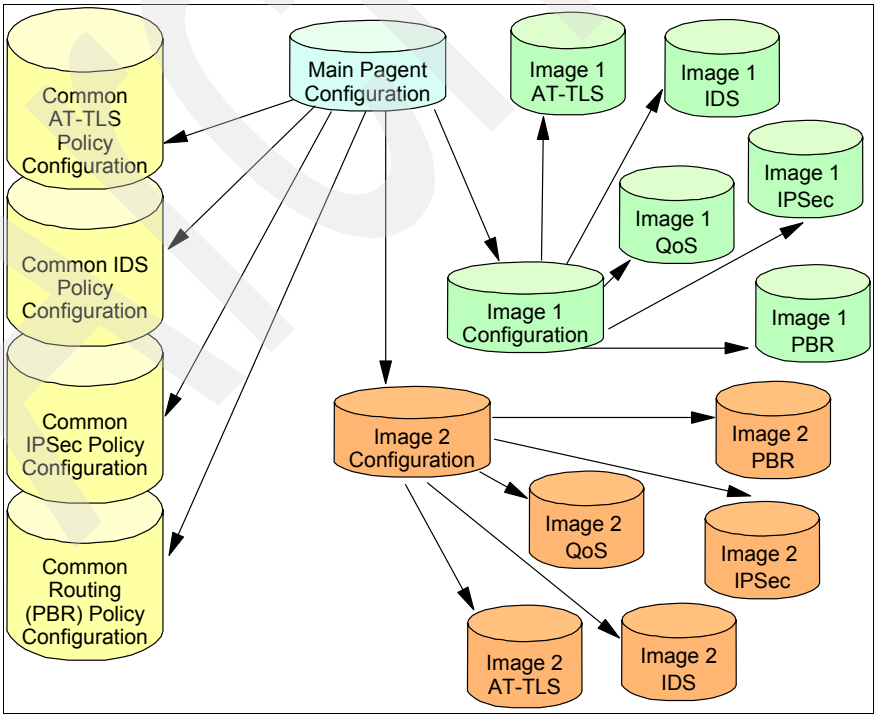


Figure 6-1 Hierarchy of PAGENT configuration files

Each image configuration file is used to configure policies for one TCP/IP stack. The image files can in turn point to image-specific files for the different policy types: AT-TLS, IPSec, IDS, QoS, and Routing or Policy-based Routing (PBR). A given common configuration file applies to all TCP/IP stacks. This allows policy definitions that can be shared among all the TCP/IP stacks to be placed in the common file, and those that are unique to be placed in each image-specific file.

The image QoS file is optional; QoS definitions can be placed directly in the image configuration file instead of in a separate file. In addition, the statements in the image configuration files can instead be placed directly in the main configuration file simply by issuing the TcpImage statement without a separate image file path name. Such definitions will be shared only by all TCP/IP stacks that do not have their own separate image configuration file.

In a standard implementation of Policy Agent, each node running PAGENT maintains a copy of the policy definition and configuration files. It is possible to create a common policy repository in an LDAP server for QoS and IDS, but not for the other policy types. Yet the idea of a common repository remains attractive. Such a repository can be consolidated using flat files for all the policy types on a Central Policy Server.

6.2 Basic concepts

The problem being solved by centralized policy servers is primarily one of policy management. The Policy Agent configuration shown in Figure 6-1 on page 258 needs to be replicated on each system. If the IBM Configuration Assistant for z/OS Communications Server is used to configure policy definitions, it also must be replicated on (or have connectivity to) each system. It is not possible to use LDAP as a centralized policy repository, because the LDAP implementation only supports the QoS and IDS policy types.

The solution is to use the Policy Agent itself as a centralized policy repository, introducing the roles of *policy server* and *policy client.* The policy server provides centralized administration and management of policy definitions. The IBM Configuration Assistant only needs connectivity to the policy server if no local policies are defined on the policy clients. Figure 6-2 on page 260 shows an overview of the distributed (or centralized) policy services solution.

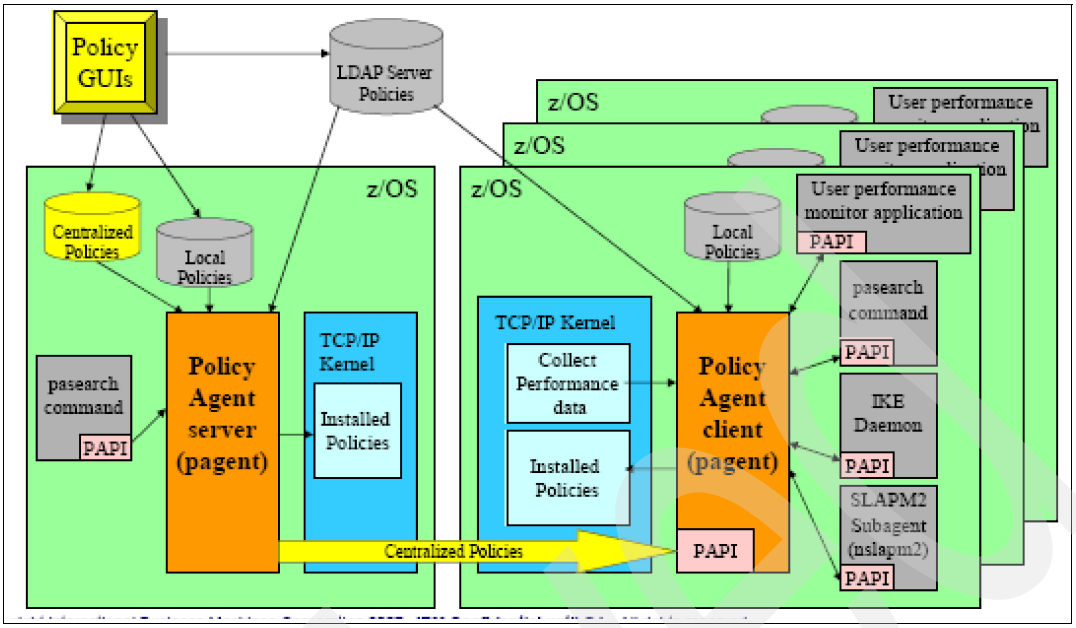


Figure 6-2 The Centralized Policy Services solution

On the right side of Figure 6-2 are a number of policy clients. Each policy client can use a local configuration file, or LDAP policies, if needed. The major policy applications are shown for each policy client.

On the left side is shown the policy server. Centralized policies are defined, but are not installed in any TCP/IP stacks on the policy server. These centralized policies are retrieved by the policy clients using the existing Policy Agent API (PAPI). The IBM Configuration Assistant can be used to define the centralized policies and the local policies for the policy server and policy clients. The user of the IBM Configuration Assistant only needs to have a connection to the policy server node, and not to every node that requires policies.

Two other GUIs are available to populate the LDAP server with QoS and IDS policies: the zQoS Manager and the zIDS Manager.

To take full advantage of the centralized policy services solution, local policies should not be defined on the policy clients. The policy server is not itself considered a policy client, so local policies on the policy server are normal and expected.

Local or remote policies can be used for each policy type, and for each TCP/IP stack, individually. These choices can be changed at any time without restarting the Policy Agent. This allows a gradual and controlled migration from local to remote policies.

Also note that secure, long-running connections are used between the policy clients and the policy server. The policy server utilizes local AT-TLS policies to accomplish this. However, it is not possible to use AT-TLS policies on the policy client because of the “chicken-and-egg” problem: if the AT-TLS policies reside on the policy server, the policy client would need to connect to the policy server in order to obtain the policies that secure that very connection. For this reason, the policy clients are configured as SSL clients, using local definitions in the image configuration files.

6.3 Configuring distributed (centralized) policy services

The problem being solved by distributed policy services, also known as centralized policy services, is primarily policy management. Policies are administered by one site, the server, with policy clients retrieving their policies from that single repository.

Note: The tasks, examples, and references in this chapter are based on the configuration shown in Figure 6-3.

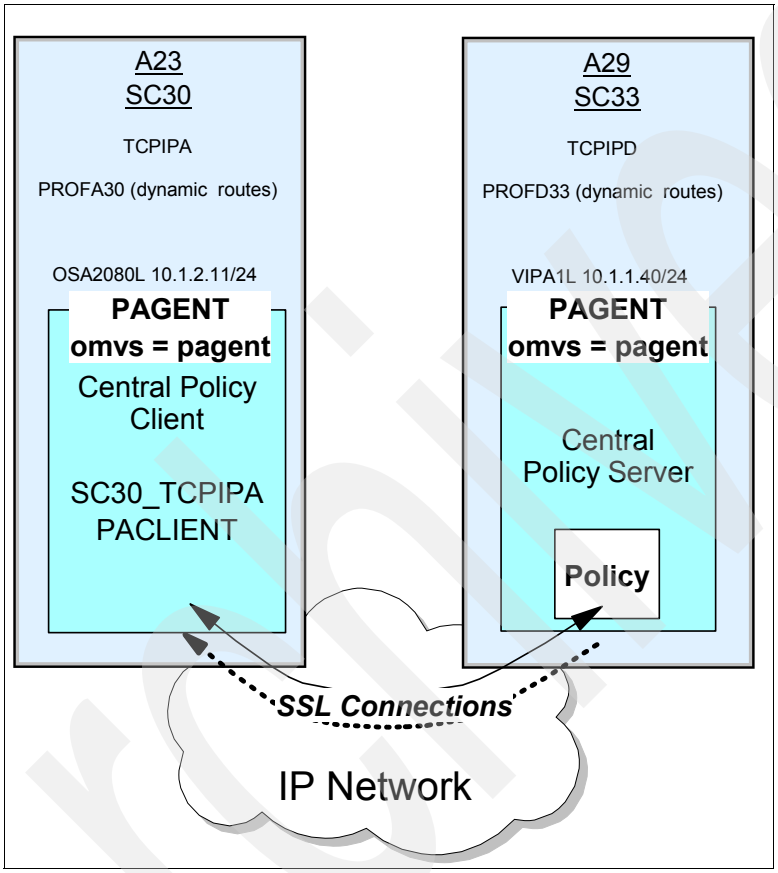


Figure 6-3 Relationship between Central Policy Server and Policy Client

In Example 6-3 on page 267, our example server has an MVS system ID of SC33 and our policy client has an MVS system ID of SC30. The name of the Policy Agent procedure at each MVS system is PAGENT, and it is associated with the OMVS segment or user ID called `pagent`. The client image name known to the server is SC30_TCPIPA. The RACF user ID associated with the policy client is PACLIENT. This information is required in order to establish the RACF environment for the policy server and policy client relationship.

6.3.1 Configuring the base environment

The base environment for distributed policy services (centralized policy services) has four prerequisites:

- An existing Policy Agent
- A set of policies we want to distribute to clients

- A server certificate: either a private USER certificate or a SITE certificate with the appropriate key ring authorizations
- A RACF environment that authenticates the user:
  - Either one or more RACF user IDs with passwords that can be presented for authorization to the server when the client connects
  - Or one or more PassTicket definition sets for authorization

First, a working, existing Policy Agent is needed. Refer to Chapter 5, “Policy Agent” on page 221 for information about how to create this environment.

Next, the policies that we want to distribute to clients are needed. For information about how to build a set of policies, refer to any of the following content:

- “Quality of Service” on page 295
- “Policy-based routing” on page 501
- “Telnet security” on page 539
- “Secure File Transfer Protocol” on page 579
- Chapter 8, “IP filtering” on page 319
- Chapter 9, “IP Security” on page 373

Figure 6-4 illustrates the two categories of policies that a central policy server may distribute to its clients: Common Policies and Stack-Specific Policies. Note that there is no Common Policy category for the QoS policy type.

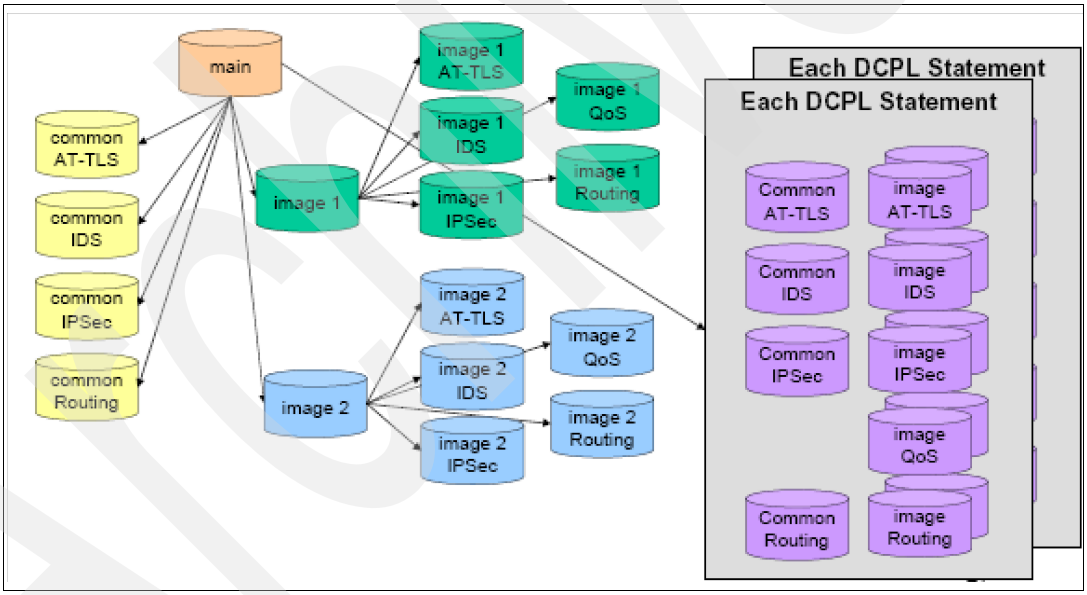


Figure 6-4 DynamicConfigPolicyLoad (DCPL): What central policy server distributes to policy clients

Then, we need a key ring and certificate environment for the SSL connection between policy server and its clients. Therefore we require a server certificate for the policy server. We already had one available: a site certificate that we used for both the TN3270 server and the FTP server. We decided to use the same one and to associate PAGENT with that certificate.

Note: If your enterprise needs more stringent security controls, you may not want to employ the shared SITE certificate for the policy server. Instead, you may want to create a new USER certificate that is distinct from the other server certificates.

For more information about certificate management, refer to Chapter 3, “Certificate management in z/OS” on page 31. You can read about how we created that certificate and

the Certificate Authority certificate in 15.3.2, “Configuration of FTP native TLS security” on page 586. We repeat some of that information here to highlight that the policy agent server needs to be permitted to the key ring with its keys and certificates. The certificate definitions we need for policy server are described in “Certificate management for central policy services” on page 263.

Finally, we need a RACF identity for our policy clients. We do not use a client SSL certificate for our client authentication; instead we use a user ID and either a password or a pass ticket. In our scenario, we show you how to establish a user ID and password in “RACF user ID and password for the policy client” on page 265. To learn how to set up the pass ticket environment, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Certificate management for central policy services

Example 6-1 shows the commands that are used to create a shared key ring, to create the CA certificate, and to create the SITE certificate. Most of these commands will already have been executed for other types of servers. We show you these commands again (1 through 5) in order to put the new commands (5** and 7*) into context. Command sequences 6** and 7** associate the PAGENT user ID of PAGENT with the ability to access the shared key ring and its keys and certificates.

Example 6-1 Commands to create shared key ring, CA, and SITE certificates

```
1
SETROPTS CLASSACT(DIGTCERT)

SETROPTS CLASSACT(DIGTRING)

2
RACDCERT ID(TCPIP) ADDRING(SharedRing1)

SETROPTS RACLIST(DIGTCERT) REFRESH

SETROPTS RACLIST(DIGTRING) REFRESH

RACDCERT LISTRING(*) ID(TCPIP)

3

RACDCDERT CERTAUTH GENCERT SUBJECTSDN( O('IBM Corporation')OU('ITSO Certificate
Authority') C('US')) NOTBEFORE(DATE(2007-09-11)) NOTAFTER(DATE(2008-09-11))
KEYUSAGE(CERTSIGN) WITHLABEL(‘CS19 ITSO CA1')

SETROPTS RACLIST(FACILITY) REFRESH

RACDCERT CERTAUTH LIST

4
RACDCERT SITE GENCERT  SUBJECTSDN(CN('ITSO.IBM.COM') OU('ITSO CS19 Shared Site')
C('US')) WITHLABEL('CS19 ITSO SharedSite1') SIGNWITH(CERTAUTH LABEL(‘CS19 ITSO CA1’)

SETROPTS RACLIST(FACILITY) REFRESH

RACDCERT SITE LIST

5
RACDCERT ID(TCPIP) CONNECT(CERTAUTH LABEL('CS19 ITSO CA1’)RING(SharedRing1)
USAGE(CERTAUTH)
```

```
RACDCERT ID(TCPIP) CONNECT(SITE LABEL('CS19 ITSO SharedSite1') RING(SharedRing1)
USAGE(PERSONAL) DEFAULT)

SETROPTS RACLIST(DIGTCERT) REFRESH

SETROPTS RACLIST(DIGTRING) REFRESH

RACDCERT LISTRING(*) ID(TCPIP)

6**
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(PAGENT) ACCESS(CONTROL)

SETROPTS RACLIST(FACILITY) REFRESH

7**
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(PAGENT)  ACCESS(UPDATE)

SETROPTS RACLIST(FACILITY) REFRESH
```

In Example 6-1 on page 263 we were able to take advantage of many of the steps we executed in other chapters to create a key ring and certificates. However, because the Policy Agent is associated with the OMVS segment owned by user ID PAGENT, we had to execute steps **6**** and **7**** to associate that user ID with access to the certificates, the key ring, and the keys stored in the ring.

Here we review the steps in Example 6-1 on page 263:

1 It is unnecessary to perform this step again. It was already executed in 15.3.2, “Configuration of FTP native TLS security” on page 586 when we were preparing to create key rings and certificates for the TLS/SSL and AT-TLS environments.

2 It is unnecessary to perform this step again. It was already executed in 15.3.2, “Configuration of FTP native TLS security” on page 586 when we created the shared key ring for our installation.

3 It is unnecessary to perform this step again. It was already executed in 15.3.2, “Configuration of FTP native TLS security” on page 586 when we created the Certificate Authority certificate.

4 It is unnecessary to perform this step again. It was already executed in 15.3.2, “Configuration of FTP native TLS security” on page 586 when we created the server SITE certificate.

5 It is unnecessary to perform this step again. It was already executed in 15.3.2, “Configuration of FTP native TLS security” on page 586 when we connected the two certificates (CA and server SITE) to the shared keyring owned by the user ID TCPIP.

6** Every user, whether a server or client, requires access to the private key of the shared SITE certificate. These commands provide that access to the user ID PAGENT.

7** Owners of a key ring need READ access to it. Non-owners of a key ring need UPDATE access to it. The PAGENT procedure is owned by the user ID PAGENT and must therefore be permitted to the key ring owned by the user ID TCPIP.

Policy client user ID versus policy client name (symbolic name)

You must learn to distinguish between two *types of identities* that the central policy server uses in authenticating the policy client: the client user ID and the policy client name.

Client user ID

In our scenario, we assigned PACLIENT as the client user ID. The definitions for PACLIENT are presented in “RACF user ID and password for the policy client” on page 265 of this chapter.

Policy client name (symbolic name)

In our scenario, we assigned a policy client name of SC30_TCPIPA. The policy client name is also known as the symbolic name of the policy client. We chose to use the MVS system name of the client node together with an underscore character and the TCP/IP stack name of the client to represent the policy client name. This naming convention is also the default naming convention if a definition for a policy client name is omitted from the central policy server implementation.

Policy clients are matched to a DynamicConfigPolicyLoad statement in the server's main PAGENT configuration file based upon the “policy client name.” We show you how to use the statement later in this chapter.

If you assign a naming convention for the policy client names, you can use symbolic expressions to reduce the amount of coding required at the central policy server on the DynamicConfigPolicyLoad statement.

RACF user ID and password for the policy client

At connect time, the client must present a user ID and either a password or passticket to the server node. The user ID is first used to authenticate policy clients when they connect, and then to access the SERVAUTH profiles that determine which policy types the client is allowed to use.

Therefore we created a policy client user ID of PACLIENT with a password of CLIENTPA to satisfy these requirements. In a production environment, you would apply stringent password rules to this client, but for our test purposes, our selected password sufficed. In Example 6-2 you see the JCL job stream we used to create this user ID.

Example 6-2 Creating the policy client user ID and password in RACF

```
//ADDPACLI JOB (999,POK),'ADD A USER',CLASS=A,REGION=0M,
//     MSGCLASS=T,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM SYSAFF=*
//*
//* Use this job to create a user ID that does not use TSO and
//* require no particular RACF authority.
//*
//IKJEFT   EXEC PGM=IKJEFT01
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
   ADDUSER PACLIENT PASSWORD(NEW2DAY) +    1
    NAME('ID for Comm Server') +
    OWNER(xxxxx) UACC(READ) DFLTGRP(SYS1) +
    AUTHORITY(JOIN) GRPACC +
    OMVS(AUTOUID HOME(/u/paclient) PROGRAM(/bin/sh)) 2
  CONNECT   PACLIENT GROUP(SYS1) OWNER(xxxxx) AUTHORITY(JOIN) +
```

```
           UACC(ALTER)
  ALU PACLIENT PASSWORD(CLIENTPA) NOEXPIRED 3
  PASSWORD USER(PACLIENT) NOINTERVAL
  ADDSD     'PACLIENT.**' UACC(NONE) OWNER(PACLIENT)
  SETROPTS REFRESH RACLIST(TSOPROC ACCTNUM TSOAUTH)
  DEFINE ALIAS   (NAME('PACLIENT') RELATE('UCAT.COMCAT'))
/*
//DEFINEZ  EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(PACLIENT.HFS) -
            LINEAR CYLINDERS(1 1) SHAREOPTIONS(3)    -
            VOLUMES(COMST4))
/*
//CREATEZ  EXEC   PGM=IOEAGFMT,
//           PARM=('-aggregate PACLIENT.HFS -compat
//             -owner PACLIENT -group SYS1 -perms o755')
//SYSPRINT DD     SYSOUT=*
//STDOUT   DD     SYSOUT=*
//STDERR   DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//CEEDUMP  DD     SYSOUT=*
//*
//COPYPRO EXEC PGM=IKJEFT01
//PDS1   DD DISP=SHR,
//       DSN=WTSCPLX5.USS.PROFILE(NEWPROF)
//HFS1   DD PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//       PATH='/u/paclient/.profile'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  PROF MSGID WTPMSG
  OCOPY INDD(PDS1) OUTDD(HFS1) TEXT
/*
//CHOWN1     EXEC PGM=BPXBATCH,
//   PARM='SH chown paclient /u/paclient/.profile'
//STDOUT     DD  PATH='/tmp/stdout',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//STDERR     DD  PATH='/tmp/stderr',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//*
```

The user ID 1 for the policy client only needs a password 2 and the OMVS segment 3. The policy client user ID does not need to be a superuser. Although the user ID standards at our test site provide for a TSO segment automatically, the policy client user ID does not need one.

Number of policy client user IDs

Each policy client may have its own unique user ID and password, or you may provide a single, shared user ID and password for a group of policy clients.

Recall that the policy client user ID is used for two purposes:

- For authentication when the client connects to the policy server
- For access to the SERVAUTH resources that we define next.

Note: If you want to use PassTicket authentication instead of a password, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for information about how to set up that type of authorization.

RACF SERVAUTH authorizations for the policy client user ID

Notice that the OMVS segment for the user ID PACLIENT did not make PACLIENT a superuser. Remote policy clients are never defined as a superuser. If a user of policy services is not a superuser, that user must be authorized to various policy types through the SERVAUTH facility. You can use wildcards for this type of definition, as shown in Example 6-3.

Example 6-3 Grant access to read policies for a non-superuser using a wildcard definition

```
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.* UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.* CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

You can use wildcards on any segment of the SERVAUTH class definition. We used a wildcard only in the last field. Notice the syntax of the SERVAUTH class that is being defined:

```
EZB.PAGENT.SC33.SC30_TCPIPA.*
```

The first two fields represent the standard EZB.PAGENT designation for Policy Agent SERVAUTH classes. The third field represents the MVS system name of the MVS node. In this case, it is the system name of the MVS running the policy server. The third field is usually represented by the TCP/IP stack name in SERVAUTH class definitions. However, this is not what you find in this definition. Instead you find SC30_TCPIPA, the policy client name. This is a name different from the client user ID of PACLIENT. This is a name you assign when you configure the policy client's PAGENT configuration file. The final field represents the policy type field: QOS, IDS, IPSEC, Routing, or TTLS. Our example shows a wildcard which represents any policy type.

The Policy Agent examines all requests from policy clients to verify that the client has SERVAUTH authority to the policy type at the server node. This means that the SERVAUTH class definition includes the policy client name as one of the segments and the PERMIT command authorizes the policy client user ID to the SERVAUTH class.

As an alternative, you can use individual definitions for each policy type as we did for PACLIENT in Example 6-4 in case we needed this granularity later.

Example 6-4 Grant access to read policies for a non-superuser: discrete definitions

```
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.QOS UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.QOS CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.IDS UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.IDS CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.IPSEC UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.IPSEC CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.TTLS UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.TTLS CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
RDEFINE  SERVAUTH EZB.PAGENT.SC33.SC30_TCPIPA.Routing UACC(NONE)
PERMIT   EZB.PAGENT.SC33.SC30_TCPIPA.Routing CLASS(SERVAUTH) ID(PACLIENT) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

RACF authorizations to BPX.DAEMON for the policy client user ID

In Example 6-5 you see the permissions we granted to the policy client user ID. Although not required, it does provide additional security in the z/OS UNIX environment.

Example 6-5 BPX.DAEMON authorization for policy client user ID

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(PACLIENT) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

6.3.2 Configuring the policy server

We configured the policy server environment for the very simple scenario shown in Figure 6-5.

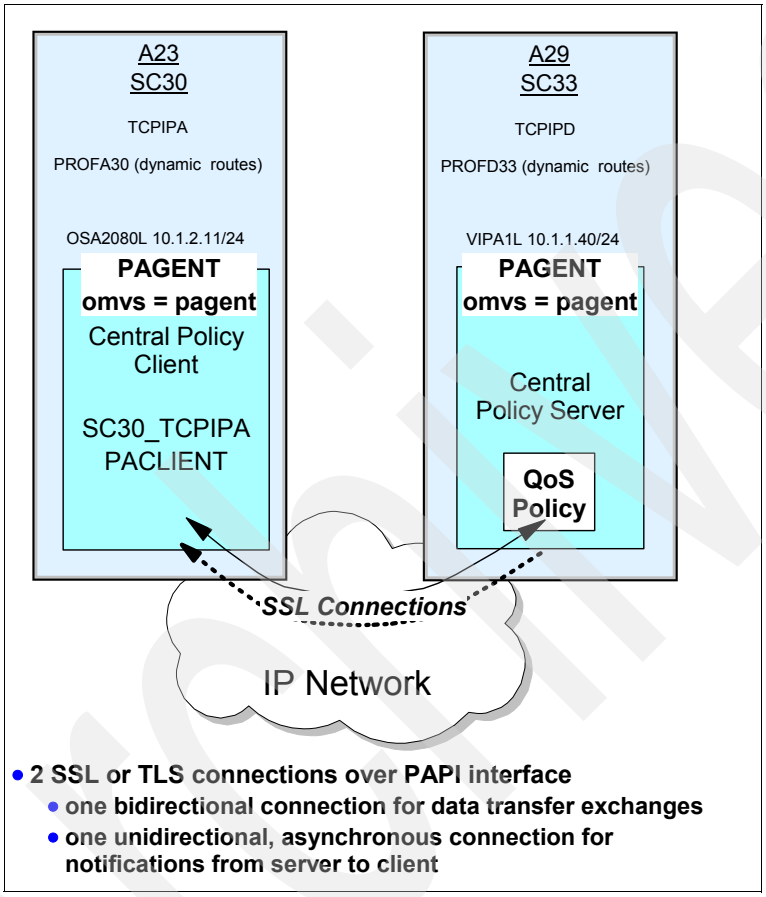


Figure 6-5 QoS policy maintained at central policy server and sent to policy client

Our scenario is set up to transfer only the QoS policy from the server to the client. We are using the password for user ID authentication. The resulting view of the successful connections between the policy server and the policy client reveal that there are two SSL or TLS connections that are established. One of the connections is used for the actual data transfer between the server and the client. The other connection is asynchronous and is used only for notifications to the client.

Note: The AT-TLS policy that is to be configured at the server node can define either SSL or TLS connections. We define the default, which is TLS connections.

In Example 6-6, you see the configuration of the policy server file that resides at TCPIPD on LPAR A29 with MVS System ID of SC33. The main configuration file for PAGENT on the LPAR contains only two image statements: for TCPIPD (a) and for TCPIPE (b). TCPIPD is to be the centralized (or distributed) policy server. TCPIPE is not to be a client yet. Note that

there is no TcpImage file for TCPIPA on LPAR A23 (z/OS System ID of SC30). TCPIPA is to be the policy client with which we test.

Example 6-6 pagent33.conf at LPAR A29 (system name of SC33)

```
#  ******* Being used as a Central Policy Server ******************
#  ******* TCPIPA on SC30 is the client for QoS  ******************
#  ************************************************************
# Loglevel 15
  TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH PURGE 600 a
  TcpImage TCPIPE /etc/pagent33_TCPIPE.conf  FLUSH PURGE 600 b
# ClientConnection Statement
#   The Policy Agent acting as a policy server uses the ClientConnectio
#   statement to specify the listening port.  The Policy Agent acting a
#   policy client uses this connection to retrieve remote policies.
#   example:
#     ClientConnection 4502
      ClientConnection 16310 c
# DynamicConfigPolicyLoad Statement
#   The Policy Agent acting as a policy server uses the DynamicConfigPoLicyLoad
#   statement to obtain the file names of the configuration files to be loaded into
#   policy clients.
#
#     DynamicConfigPolicyLoad SC30_TCPIPA d
      DynamicConfigPolicyLoad ^(.+)_(.+)$ e
      {
        RefreshInterval     1800
        PolicyType          QoS f
        {
          PolicyLoad  /etc/pagent33_QoSnew.conf g
        }
#     }
```

When we first set up the PAGENT environment for TCPIPD at LPAR A29 (sysid of SC33), we copied the sample for the main policy file from /usr/lpp/tcpip/samples/pagent.conf into a file named /etc/pagent33.conf at LPARA29. The examples within this main policy file already had a placeholder for an important set of definition statements for the distributed policy services scenario:

- `ClientConnection`
- `DynamicConfigPolicyLoad`

Our pagent33.conf was already successfully tested at this point and was being used for two TCP/IP images on LPAR A29 (SC33): TCPIPD itself, and TCPIPE. Recall that the main configuration file is coded manually; the IBM Configuration Assistant is not employed to create this file.

Note the TcpImage statements at **a** and **b** in Example 6-6. These **T**cpImage statements existed before for the two stacks in this LPAR represented by MVS System Name of SC33. We simply uncommented the two new statements (ClientConnection and DynamicConfigPolicyLoad) for the distributed policy environment and filled in the appropriate values; see **c** and **d** in Example 6-6.

The ClientConnection statement **c** identifies the listening port that the policy server binds to and over which it listens for client connections. The default port is 16310 despite the declaration in the sample file that the default file is 4502. The DynamicConfigPolicyLoad statements, **d** and **e**, identify the clients that are authorized to retrieve policies from the server, as well as the policy type that is to be transferred to the authorized client and the location of the file that defines the policy for that policy type.

The DynamicConfigPolicyLoad statement labeled **d** shows a hard-coded client name of SC30_TCPIPA. The DynamicConfigPolicyLoad statement labeled **e** shows a string that allows a kind of variable name or symbolic name to be presented by a client: `^(.+)_(.+)$`. This type of coding uses what is known as *regular expressions* in the C and C++ coding languages and in UNIX.

This is the client name that we used for the definition of the SERVAUTH class in Example 6-3 on page 267 and Example 6-4 on page 267. The explanation for this string can be found in the sample policy file and in *z/OS Communications Server: IP Configuration Guide*, SC31-8775 and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Our value, *^(.+)_(.+)$*, means that we begin the string with any number or type of characters separated by an underscore and followed by any number or type of characters. Therefore, our naming convention for the policy client name, SC30_TCPIPA, matches this pattern. So does another possible client named SC31_TCPIPB. The use of variables or regular expressions in this statement allows a single DynamicConfigPolicyLoad statement to apply to multiple clients.

When the central policy server receives an incoming request from the policy client, it searches for a matching client name in the DynamicConfigPolicyLoad statements using the following order:

1. A clientname that has an exact match to the policy client name.
2. A longest regular expression name that matches the pattern presented by the client name.
3. If there is no matching clientname or if the matching client name does not have a corresponding PolicyType parameter as represented in the client request, a default remote file is used. The default file has the name pagent_remote.<policytype>.

Creating variable strings for the DynamicConfigPolicyLoad statement

It is beyond the scope of this book to list all the possible ways to create a variable string; refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for more information about this topic. However, Table 6-1 lists and explains the meaning of various regular expressions to provide insight into how our string was constructed.

Table 6-1 Regular expressions mapping table

Character	Meaning	IBM-1047 EBCDIC hex value
^	Beginning of variable string	x'5F'
.	Match any single character	x'4B'
*	Partial wildcard only because must follow a character or expression; cannot stand on its own; any number or type of character. A full wildcard can be represented with .*	x'5C'
+	One or more occurrences of a previous character	x'4E'
(expression)	Groups a sub-expression	(is x'4D') is x'5D'
[character-character]	In sequence: specified character through specified character	[is x'AD'] is x'BD'
$	End of variable string	x'5B'

Important: Policy Agent syntax and particularly the symbolic expressions in the DynamicConfigPolicyLoad statement require the use of US English Code Page 1047. The default in many 3270 emulators is US English Code Page 037.

This point is of particular interest when typing the caret (^) character. With code page 1047, the ^ is represented by EBCDIC x'5F'. With code page 037, however, the ^ character is represented by EBCDIC x'B0'.

If x'B0' is encountered to represent ^ during parsing of an expression, the expression is not understood and the policy server sends a default policy to the client instead of the intended policy.

Use care when coding files that include regular expressions. There are several techniques you can use to ensure that the code page you use will allow your files to be properly parsed:

- Use a 3270 emulator and ensure that you have specified IBM-1047 as the code page for the session in which you are editing the regular expressions. Even in this case you may need to remap one of the keys on the keyboard to represent the caret character.
- Use your standard TN3270 emulator to code the files. When you are finished editing, enable "hex on" in the ISPF oedit process. Verify that the characters you have used in your expressions map correctly to IBM-1047. (See the mapping provided in Table 6-1 on page 270 for help, or refer to *Enterprise Systems Architecture/390 Reference Summary*, SA22-7209, for the code page mappings.)
- Use your workstation to code the files, and then ftp them onto the mainframe. Verify the correct hexadecimal EBCDIC mapping.
- Telnet into the UNIX subsystem to edit the files in OMVS. Use vi or a similar UNIX editor. Again, verify the correct hexadecimal EBCDIC mapping by entering the ISHELL or the OMVS shell and turning hex viewing on.

How will you know that your coding is successful? The **pasearch -c -p <policyclientname>** command that we illustrate later in this chapter tells you which files have been loaded; the syslog daemon log file tells you if parsing failed.

We have not yet completed the explanation of several definitions in the policy server's configuration file. Example 6-7 shows that you can include a CommonPolicyLoad statement in the pagent.conf file.

Example 6-7 pagent33.conf file for central policy server: DynamicConfigPolicyLoad Statement

```
# DynamicConfigPolicyLoad Statement
#   The Policy Agent acting as a policy server uses the DynamicConfigPo
#   statement to obtain the file names of the configuration files to be
#   policy clients.
#   example:
#     DynamicConfigPolicyLoad remote*
#     {
#       RefreshInterval     1800
#       PolicyType          IPSec
#       {
#         CommonPolicyLoad  /etc/pagent_remote.ipsec   c
#       }
#       PolicyType          TTLS
#       {
#         PolicyLoad        /etc/pagent_remote.ttls
#       }
#     }
```

```
#
#     DynamicConfigPolicyLoad SC30_TCPIPA d
      DynamicConfigPolicyLoad ^(.+)_(.+)$ e
      {
        RefreshInterval     1800
        PolicyType          QoS f
        {
          PolicyLoad  /etc/pagent33_QoSnew.conf g
        }
#     }
```

In Example 6-7 on page 271, you see at label c that you can also point to a Common Policy that should be loaded into the client. There is no Common Policy for a QoS policy, which explains why we have not even attempted to define a CommonPolicyLoad parameter for our scenario.

We have already reviewed the significance of the policy client name (d and e) that is part of the DynamicConfigPolicyLoad statement itself. Part of this statement is a reference to the type of policy that the client is to retrieve. For our definition, we allow the client to retrieve the QoS policy f that is loaded from the location etc/pagent33_QoSnew.conf g.

The QoS policy file named /etc/pagent33_QoSnew.conf was created earlier with the IBM Configuration Assistant and is shown in Example 6-8.

Example 6-8 Excerpts from pagent33_QoSnew.conf

```
##
## QoS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
##     And for all other stacks in the Sysplex (to be distributed by Central Server)
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R9\files\TCPIPA.qosPolicy
## FTP History:
## 2007-09-20 06:08:33  cs02 to 9.12.4.230
## 2007-09-18 05:43:42  cs02 to 9.12.4.230
##
######################################################################
# PolicyRule statements
######################################################################

policyRule 0~1
{
   PolicyRulePriority        65000
   SourcePortRange           16310
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~1
}
......
######################################################################
# PolicyAction Statements
######################################################################

PolicyAction action~1
{
   PolicyScope               DataTraffic
   OutgoingTOS               11100000
```

}

The QoS policy in Example 6-8 on page 272 is to be loaded into both TCPIPD and TCPIPA. The Central Policy Server treats this policy as a local policy for TCPIPD; it is a distributed policy for TCPIPA.

Configure the AT-TLS policy rules for the central policy server

We are using SSL between the client and server. The server needs a policy to establish the SSL environment. The client does not need a policy; we will see that the client implicitly requests SSL services through a definition in the policy client configuration file.

We already have working AT-TLS rules for several environments. Therefore we have already enabled AT-TLS in the TCPIPD stack on MVS SC33. See Chapter 11, “Application Transparent Transport Layer Security” on page 443 for information about how to enable AT-TLS in the TCP/IP stack.

We need to create a new AT-TLS policy for the central policy server using the IBM Configuration Assistant. We initialize the GUI from our workstation, and then move to the AT-TLS perspective to reexamine the AT-TLS rules already in place. Next, we select the option to create a new backing store file so that we can test the central policy connections in isolation. (We can later merge these with our previously created AT-TLS policies.)

Go to: **File** -> **Open** -> **Create new backing store**. You see our selection in Figure 6-6 on page 273.

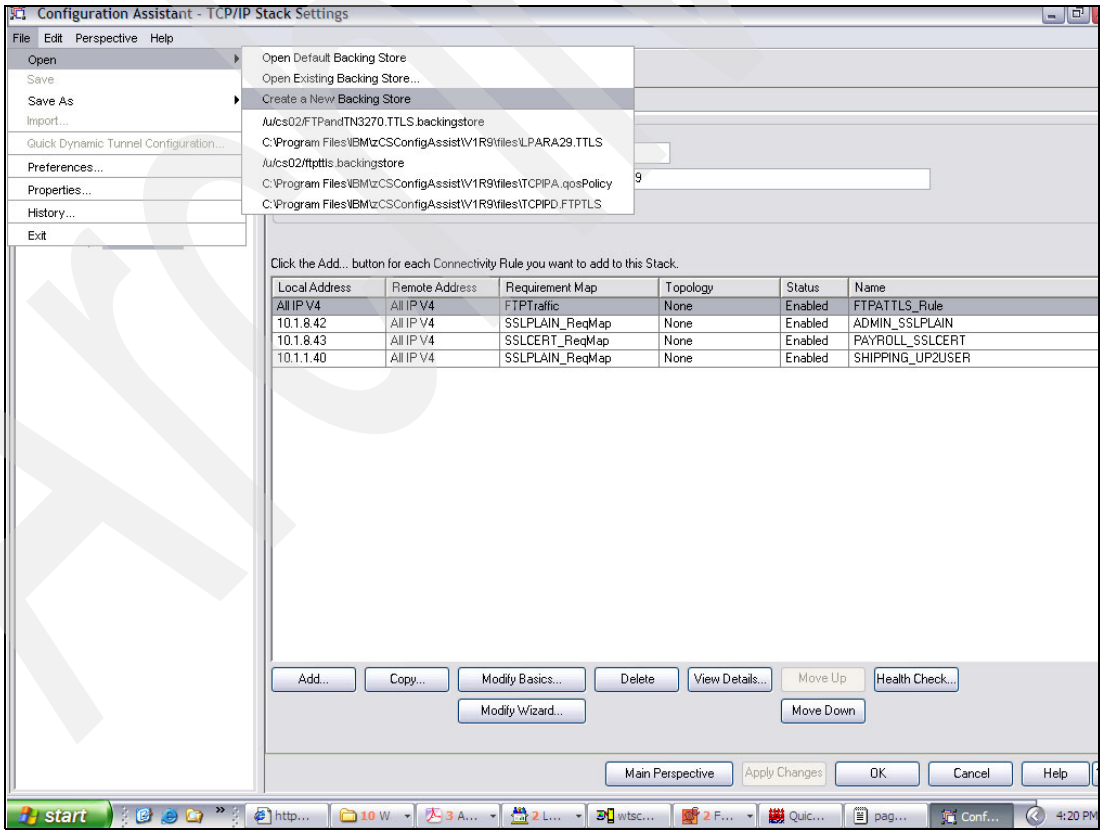


Figure 6-6 Creating a new backing store

We are asked to enter the location of the new backing store, which we do in Figure 6-7.

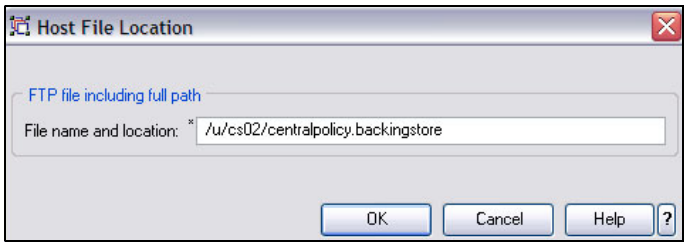


Figure 6-7 Name of new backing store file and its location

After the backing store file is created on the mainframe, the lock on the existing file with which we opened the IBM Configuration Assistant is released. We are now working on the new backing store file and we are returned to the main perspective of IBM Configuration Assistant.

Note: We chose to test the AT-TLS connection for the centralized policy server in isolation from the other AT-TLS policies by creating a new backing store file that we can later Import into the production backing store file. The Import function is discussed in "Merging (Importing) backing store files" on page 243.

Another method that we could have used to test the AT-TLS connection would have involved copying the existing backing store file into a backup and then adding the new definitions into the production backing store. With this method, we would not need to perform the IBM Configuration Assistant Import function. See "Migrating test configurations into production configurations" on page 253 for more information about that topic.

We indicate on the Main Perspective panel that we want to add a new z/OS image. We name the image "SC33," because this is the MVS System name. We do not add a comment because there are existing comments in the file we will later merge into.

Note: It is easier to merge definitions if the MVS image names are consistent. That is, the image name in the existing backing store files is SC33. Therefore, we also choose SC33 for this image name. Refer to the discussion of this topic in Chapter 5, "Policy Agent" on page 221 for more details.

We next add a TCP/IP stack name: TCPIPD. Again, we do not add a comment. We then highlight the stack name, highlight the AT-TLS technology that we want to define, and select **Enable**; see Figure 6-8.

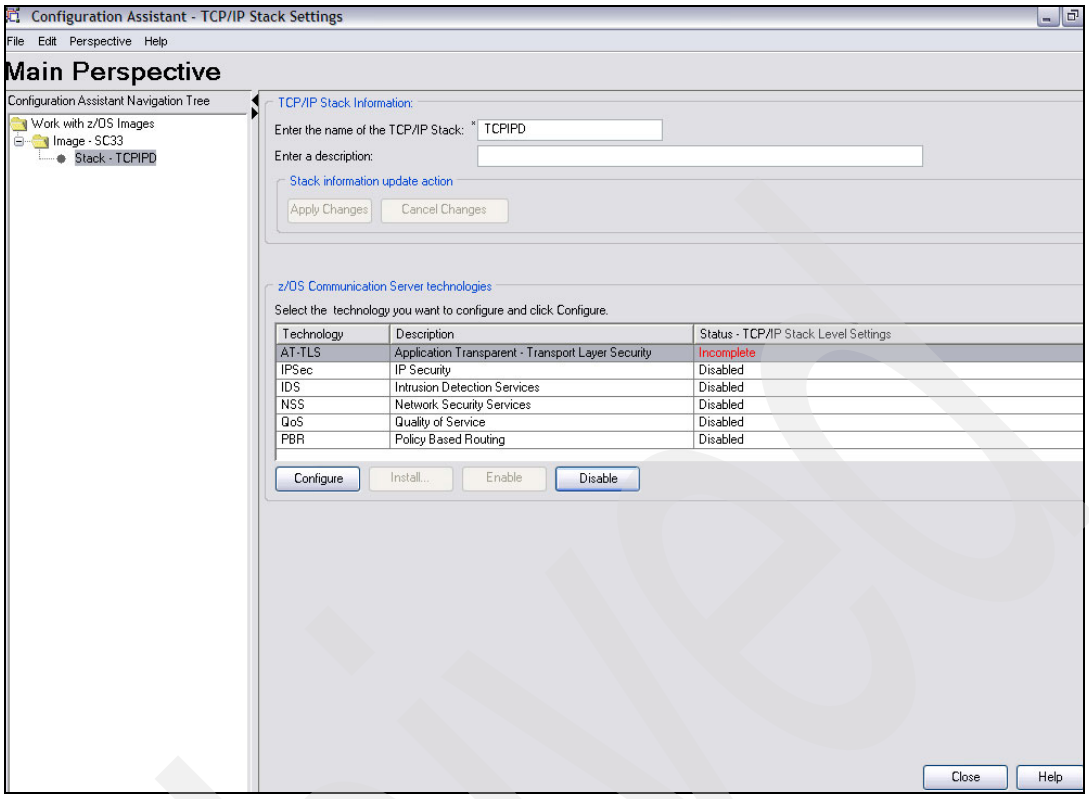


Figure 6-8 AT-TLS policy definition: enabling AT-TLS

The status column now showed `Incomplete` because we needed to configure the policy. We pressed the **Configure** button on this panel. We were asked if we wanted to create a connectivity rule, to which we responded `Yes`. We created the rule with all IPv4 addresses both locally and remotely, as you have seen in other examples in this book. We named the rule CentralPolicyRule, as shown in Figure 6-9, and did not take the default rule name.

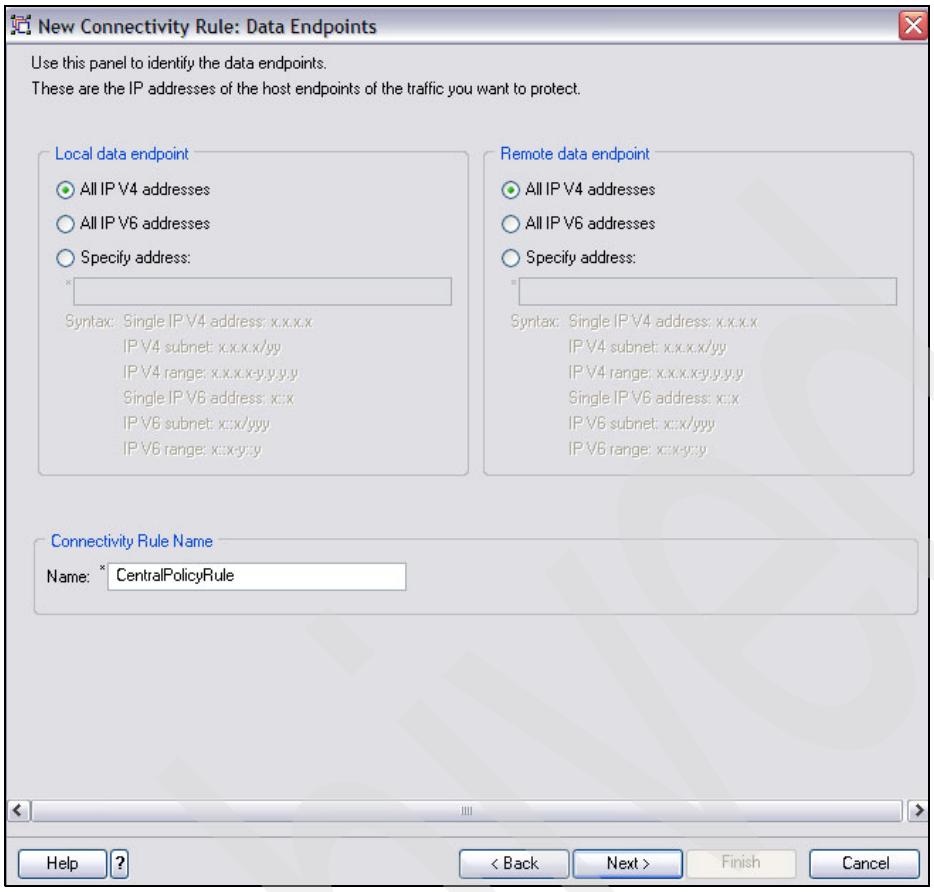


Figure 6-9 Our central policy server rule

In your case, select **Next** and on the subsequent panel select **Add for Beginners...**. This lets you work with requirement maps. Select **Next** until you get to the panel for giving a name to the Requirement map, as shown in Figure 6-10 on page 277.

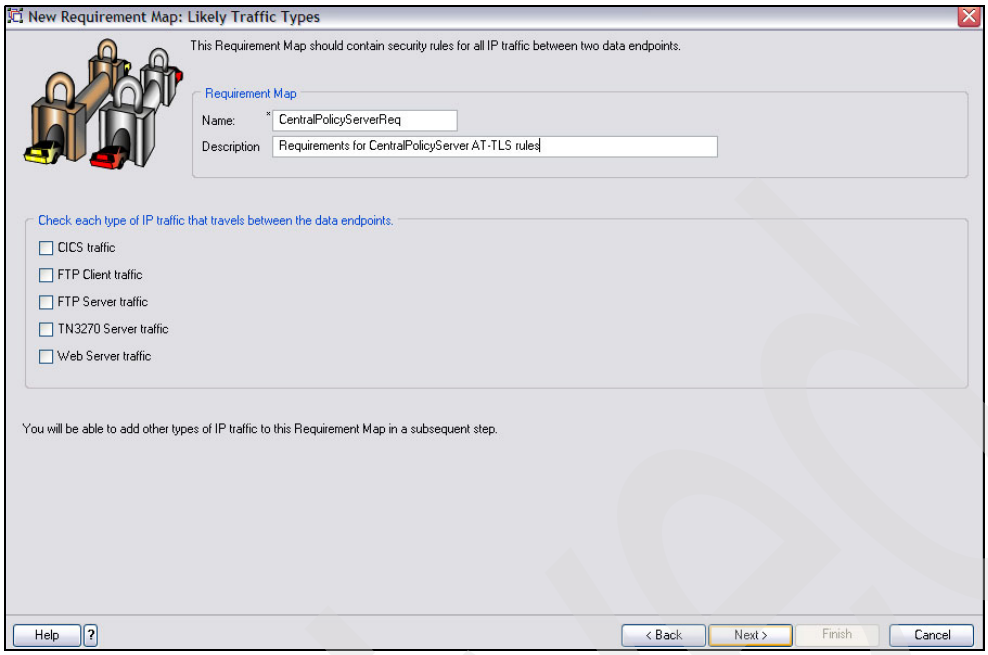


Figure 6-10 Requirement map for central policy server

Do not select an IP traffic type, and select **Next** after entering the name of the Requirement map. This action takes you to the Traffic Descriptor page where you can highlight Central_Policy_Server; see Figure 6-11.

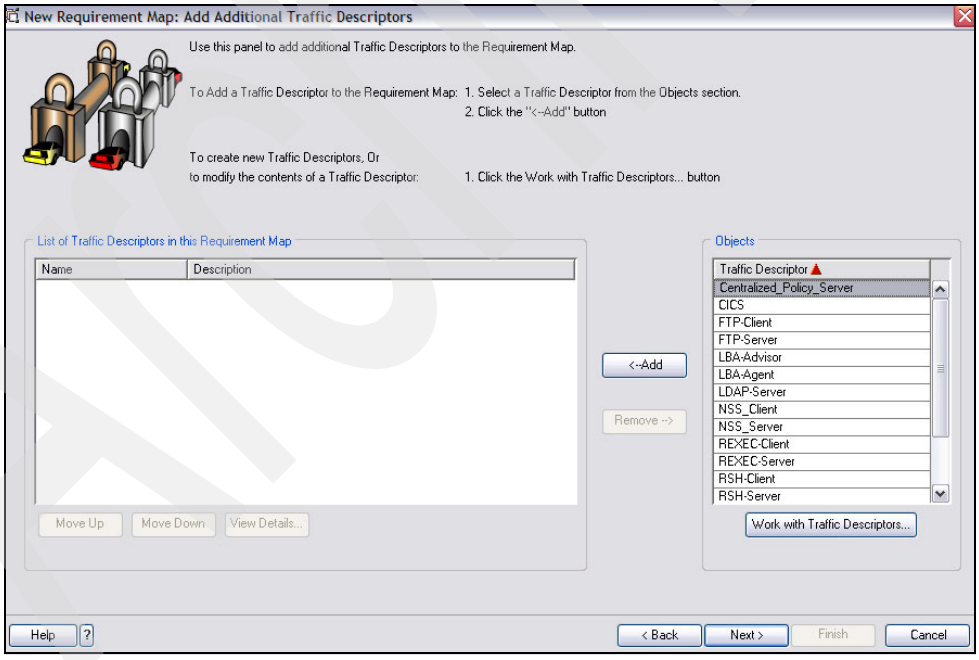


Figure 6-11 Traffic Descriptors

Select the **Add** button to move Centralized_Policy_Server to the Traffic Descriptor in the center of the panel. In our case, we pressed the **View Details** button to examine the traffic descriptor already made available by the IBM Configuration Assistant; see Figure 6-12 on page 278.

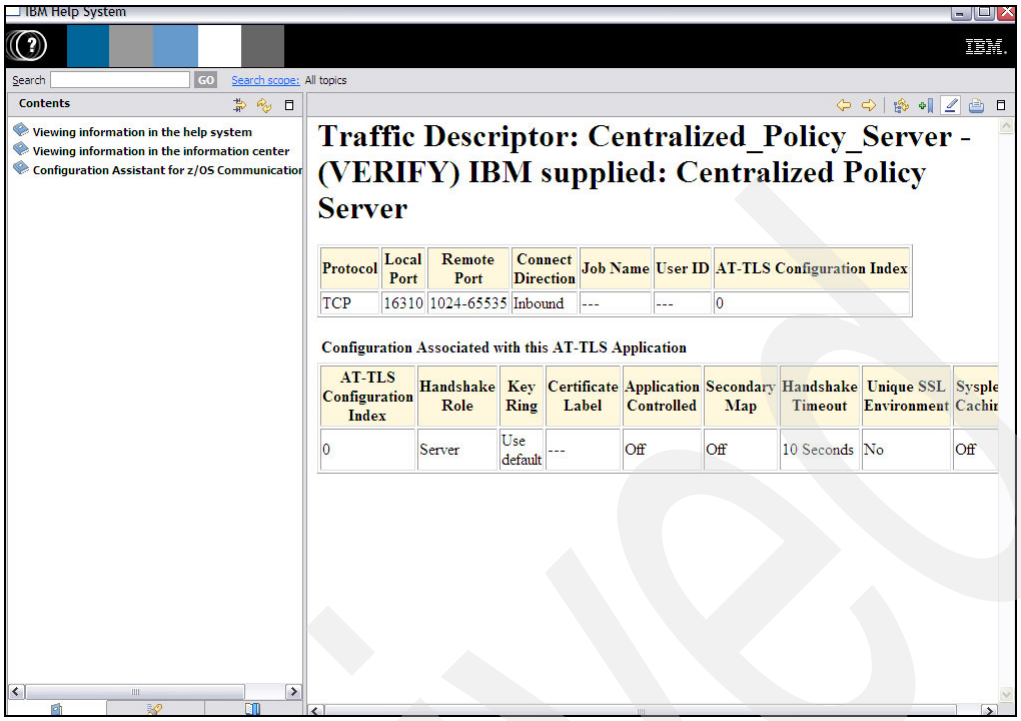


Figure 6-12 Details of Centralized Policy server Traffic Descriptor

We assigned a security level to the Traffic Descriptor, as shown in Figure 6-13.

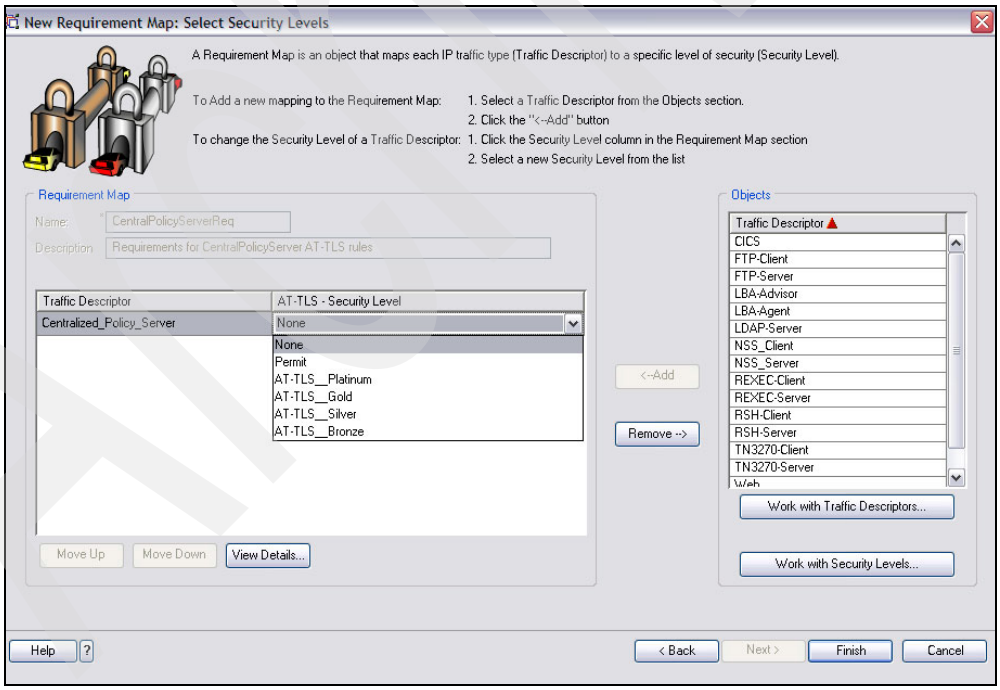


Figure 6-13 Select Silver Security level

We selected **AT-TLS Silver** as the security level, then pressed **Finish.** On the subsequent panel we pressed **Proceed** to complete our definition; see Figure 6-14 on page 279.

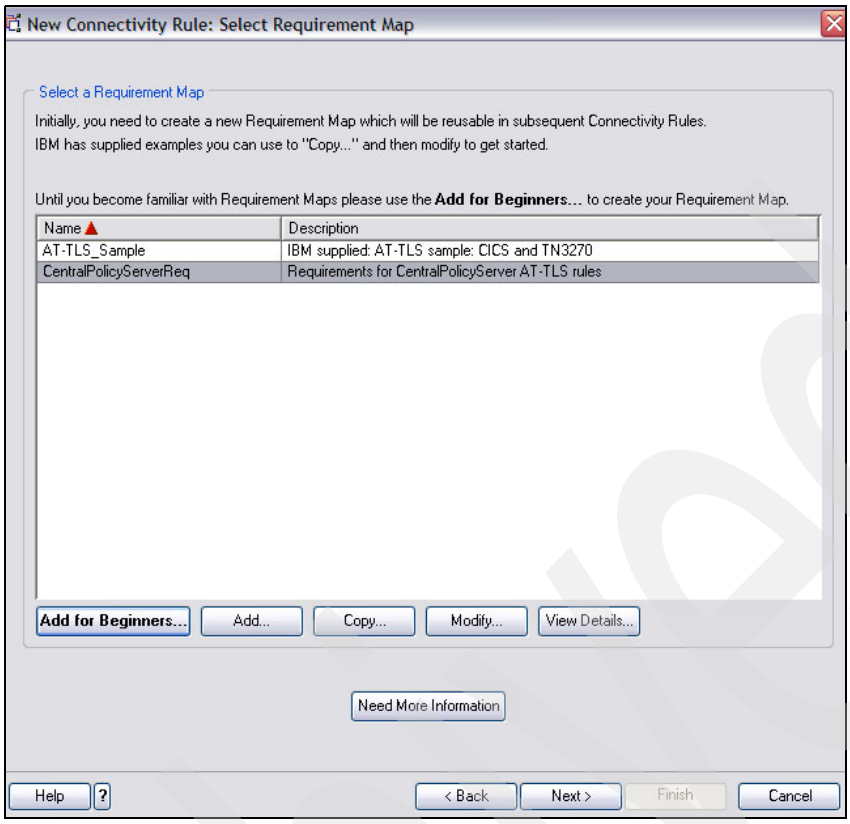


Figure 6-14 View of new connectivity rule for central policy server

We selected **Next** from this panel, which took us to the Finish panel. We could have selected Advanced settings from this panel in order to change trace and timer settings, but we elected not to. Instead, we pressed **Finish** and were presented with the panel shown in Figure 6-15 on page 280.

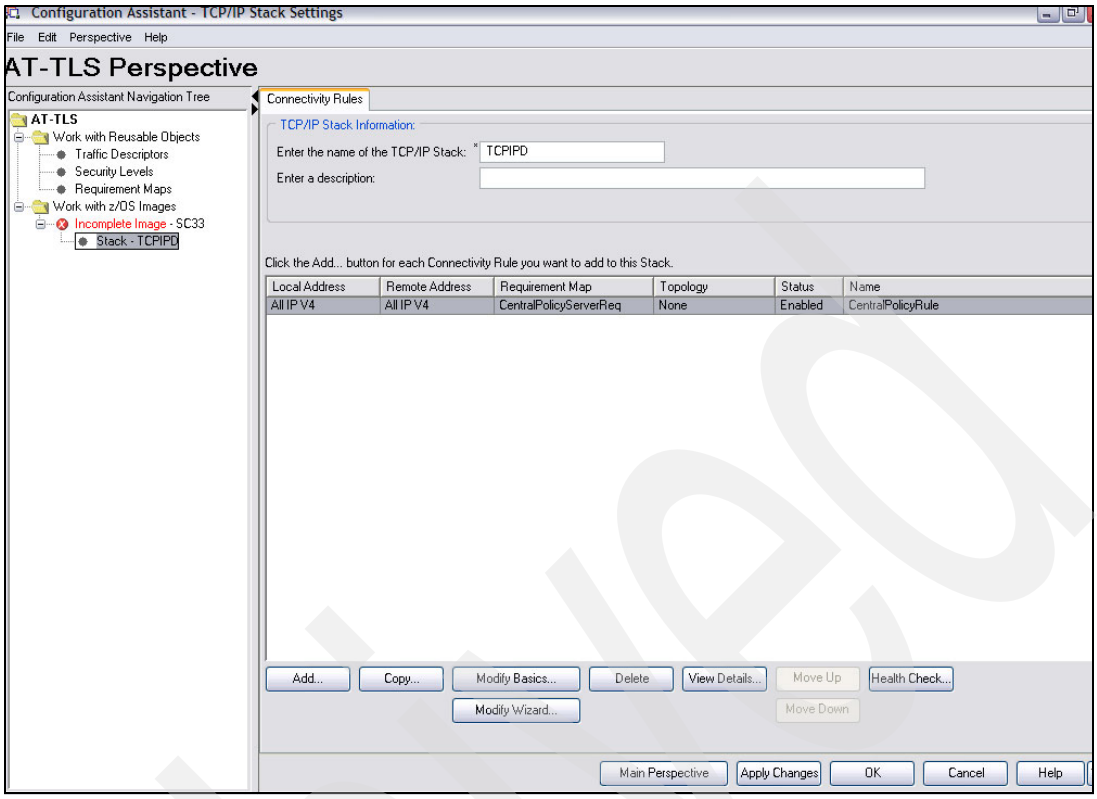


Figure 6-15 Completed connectivity rule from AT-TLS perspective

You can see that the new policy is enabled and no longer `Incomplete`. On this panel, we ran **Health Check...** and viewed the results. Our policy had no errors, so we were confident that it would work at the mainframe central policy server site.

Note that our MVS image displayed as `Incomplete`. At this point we can **`Apply Changes`** to alter this status, or we can highlight the MVS image to obtain the message in Figure 6-16:

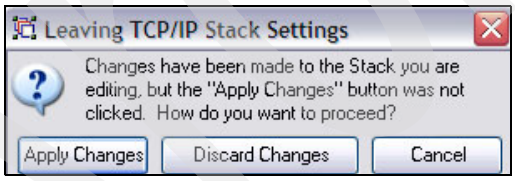


Figure 6-16 Applying changes to the MVS image

We selected **Apply Changes**, then moved to the key ring shown in Figure 6-17 on page 281.

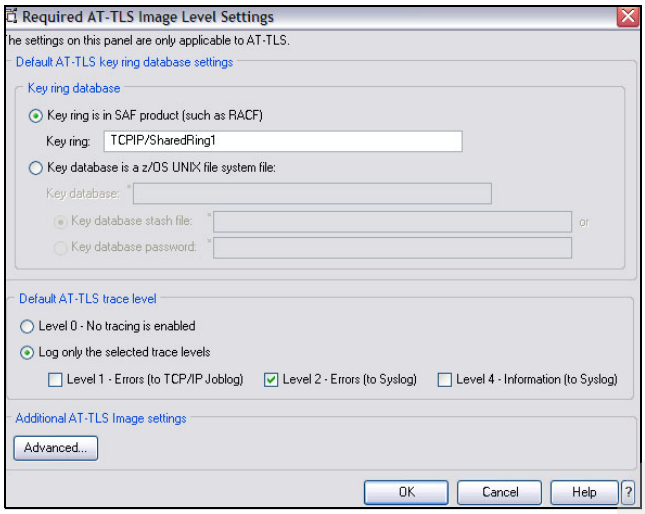


Figure 6-17 Defining key ring owner and name

We filled in the name of the server's key ring stored in the RACF repository. Note how we prefaced it with the name of the key ring's owner, TCPIP. This is a good practice although not always necessary; it is required only if the user of the key ring is not the owner of the key ring. When we pressed **OK** the `Incomplete` status for the MVS image disappeared, signalling that the MVS image definition was now complete.

We then FTPed the policy to the mainframe target, shown in Figure 6-18.

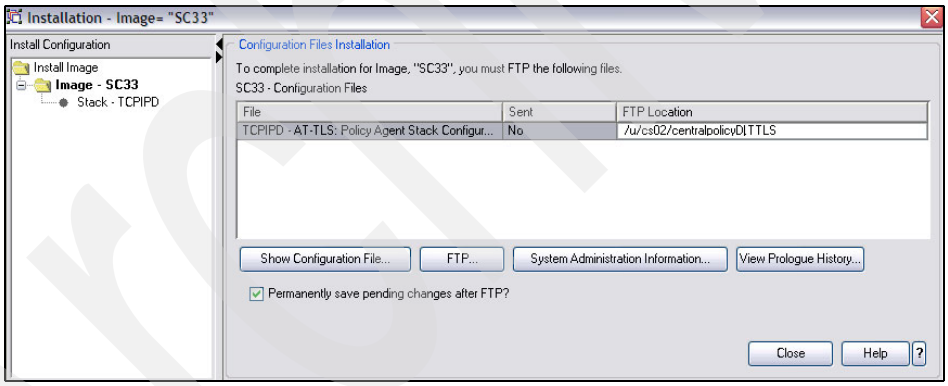


Figure 6-18 FTP the TTLS policy for central policy services to the mainframe

On this panel we identified the location to which we wanted to send the new policy file and pressed **FTP**. We filled in the target host IP address on the subsequent panel, shown in Figure 6-19 on page 282, and pressed **Send**.

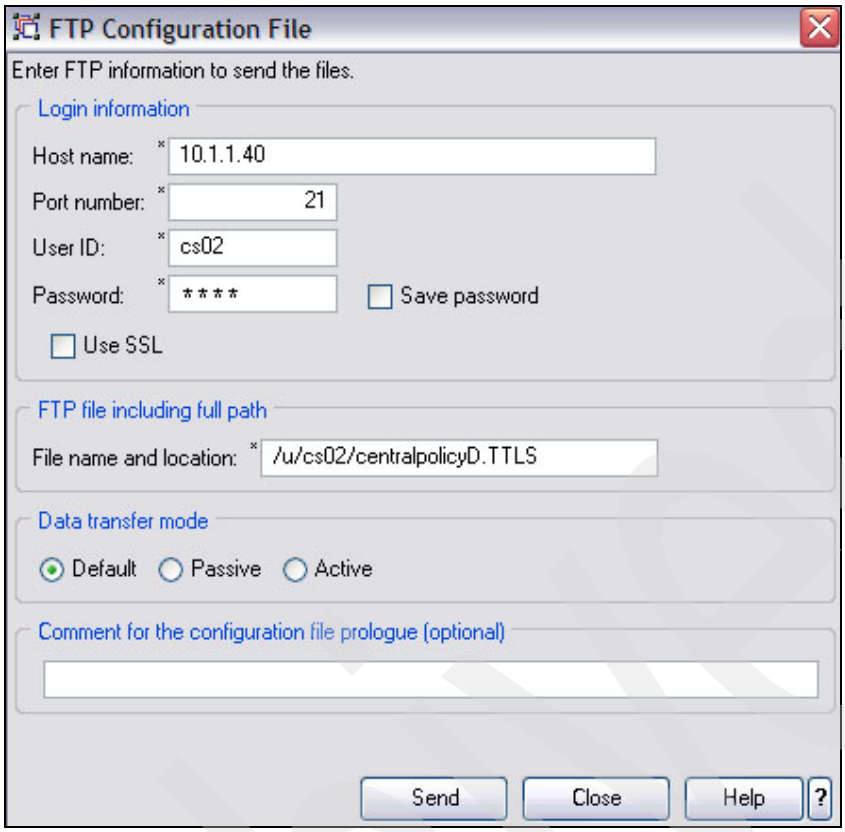


Figure 6-19 Information for FTP

After the successful FTP of the TTLS policy file, messages indicated that the backing store file was sent to the mainframe as well (earlier we had identified the mainframe as the repository for the backing store file by using the **File** -> **Preferences** pull-down).

After testing and verification, as explained in 6.4, “Activating and verifying the policy services environment” on page 288, we could import or merge the new AT-TLS policy for the centralized policy server into the existing AT-TLS policy file.

We assume that you have already permitted the user ID associated with the Policy Agent procedure to the EZB.INITSTACK.sysname.tcpname SERVAUTH class profile. If you have not, set up Policy Agent as described in “Defining the security product authorization for PAGENT” on page 231 and execute the following RACF command:

```
PERMIT EZB.INITSTACK.SC33.TCPIPD CLASS(SERVAUTH) ID(PAGENT) ACCESS(READ)
```

6.3.3 Configuring the policy client

Now that we had configured the centralized policy server and built the AT-TLS policy file for the SSL connection between client and server, we could build the PAGENT configuration files for the centralized policy client. The TCPIPA stack at LPAR A23 (system name SC30) was to be our client.

We required at least two configuration files and two configuration statements for the centralized policy client:

- The main PAGENT configuration file - which contains the ServerConnection statement
- The TcpImage configuration file - which contains the PolicyServer statement

Both of these files are edited manually without the help of the IBM Configuration Assistant. In Example 6-9 we show you the configuration of the ServerConnection statement in the main PAGENT configuration file for the policy client. We copied the sample pagent configuration file from /usr/lpp/tcpip/samples/pagent.conf and modified it for our purposes.

Example 6-9 Main configuration file (pagent30.conf) for policy client: ServerConnection

```
#  ************************************************************
#  ******* Being used as a Central Policy CLIENT (TCPIPD=Server) ***
#  ************************************************************
# LogLevel Statement
# LogLevel 511
  LogLevel 15

# TcpImage and PEPInstance Statements (synonyms)
TcpImage TCPIPA /etc/pagent30_TCPIPA.conf  FLUSH PURGE 600 1
# ServerConnection Statement: MUST BE IN CLIENT'S MAIN CONFIGURATION FILE
ServerConnection a
      {
         ServerHost                  10.1.1.40 b
         ServerPort                  16310 c
         ServerConnectWait           60
         ServerConnectRetries        3
         ServerSSL d
         {
            ServerSSLKeyring         TCPIP/SharedRing1 e
            ServerSSLV3CipherSuites  TLS_RSA_WITH_3DES_EDE_CBC_SHA      f
            ServerSSLV3CipherSuites  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA  f
            ServerSSLV3CipherSuites  TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA  f
            ServerSSLV3CipherSuites  TLS_DHE_RSA_WITH_AES_256_CBC_SHA   f
            ServerSSLV3CipherSuites  TLS_DHE_DSS_WITH_AES_256_CBC_SHA   f
            ServerSSLV3CipherSuites  TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA   f
            ServerSSLV3CipherSuites  TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA   f
         }
      }
```

Our example shows some of the main PAGENT configuration file statements. The main configuration file points to a TcpImage file called /etc/pagent30_TCPIPA.conf 1. It also contains the ServerConnection statement a that identifies this PAGENT as a client. Note the location of the various “begin” and “end” brackets that enclose the blocks of code required to establish the TCP connection to the server.

a `ServerConnection` identifies the location of the Central Policy Server, the connection timers, and the security parameters of the TLS/SSL connection to the server. The client uses the TLS/SSL connection to retrieve the policies it desires from the Central Policy Server.

b `ServerHost` identifies the IP address of the Central Policy Server. The client connects via TCP to this address. Instead of an IP address, you may choose to point to a host name so that the resolver can locate the IP address of the server.

c `ServerPort` identifies the target TLS/SSL port that the client connects to. This is the listening port at the Central Policy Server.

d `ServerSSL` indicates the start of the block of definitions that define the security parameters to be used over the TLS/SSL connection.

e `ServerSSLKeyring` provides the name of the TLS/SSL keyring at the client side of the connection. It must be prefaced with the user ID that owns the SSL keyring if the PAGENT user ID is not the owner of the key ring. This is the key ring that contains the CA certificate

used to authenticate the server SITE certificate; it may also contain the client certificate if SSL Client Authentication has been enabled for the connection. (Note that if the server certificate is a self-signed certificate, then this is the key ring that contains the copy of the server self-signed certificate.)

f `ServerSSLV3CipherSuites` indicate the names of the eligible CipherSuites that can be used to negotiate the TLS/SSL connection between the client and the server. They are listed in the order of preference.

In Example 6-10 you see how we have customized the client's TcpImage file identified by **1** in Example 6-9 on page 283. We copied the sample image file from /usr/lpp/tcpip/samples/pagent.conf, omitting all statements except for PolicyServer.

Example 6-10 pagent30_TCPIPA.conf at LPAR A23 (SC30)

```
# #  ******* Being used as a Central Policy Client (TCPIPD=Server) ***
#  IMAGE FILE FOR TCPIPA ********************************************
#############******  BELOW:   For CSV1R9 Redbook ********************
# PolicyServer Statement:  MUST BE CONFIGURED IN IMAGE FILE OF CLIENT
#
#   If a ServerConnection statement is not configured in the main config
#   then this statement is ignored.

      PolicyServer g
      {
         Userid             PACLIENT h
         AuthBy             Password   CLIENTPA i
         ClientName         SC30_TCPIPA j
         PolicyType         QOS                    k
         {
            FLUSH
            PURGE
         }
      }
```

g `PolicyServer` begins the block of code used to identify the authentication parameters of the policy client and the policy types that the client wants to retrieve from the Central Policy Server.

h `Userid` represents the user ID that you assigned to the policy client in "RACF user ID and password for the policy client" on page 265. The user ID does not need to be unique. Multiple policy clients may share the same credentials: User ID and AuthBy value.

i `AuthBy` represents the authentication method to be used after the server accepts the user ID presented by the policy client. Authentication may take place with a `Password` or with a `Passticket`. We recommend that you first implement the Policy Server and Policy Client relationship with the simple `Password` method. After you have the implementation functions as you wish, then you can change to an AuthBy Passticket user ID that you assigned to the policy client in"RACF user ID and password for the policy client" on page 265. In our example, we used the Password authentication method; we named the password assigned to the policy client user ID in "RACF user ID and password for the policy client" on page 265. The password `CLIENTPA` is stored in the clear in this TcpImage file; if you need more security surrounding this password, you can switch to Passticket authentication, which uses PTKTDATA class profiles on the client and server to generate a one-time session key with every connection. Refer to *z/OS Communications Server: IP Configuration Guide*,

SC31-8775, for instructions about defining the PTKTDATA class profiles enabling Passticket authentication for the policy client.

j `ClientName` is passed by the client during connection processing. The policy server uses this name to find the list of policies that this client is allowed to retrieve and to determine what type of security authorization is valid for this client. Unlike the Userid value, the ClientName must be unique for each client. If you omit this parameter, the policy client generates this name by combining the system's host name and the TCP stack name. If the system host name is MVS30 and the TCP stack name is TCPIP, the policy ClientName becomes MVS30_TCPIP.

k `PolicyType` identifies the policies that the client wants to obtain from the server. In our example we are retrieving only QoS policies. The complete list of valid values is: IDS, IPSec, QoS, Routing, and TTLS.

Note: If you have older PAGENT configuration files with Traffic Regulation (TR) policies defined under QoS, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for information about considerations for TR and IDS policies.

6.3.4 Correlating the definitions at the policy server and policy client

You now have several sets of definitions for the Policy Server and for the Policy Client in RACF and in PAGENT. In Figure 6-20 on page 286 you see how a subset of the definitions relate to each other.

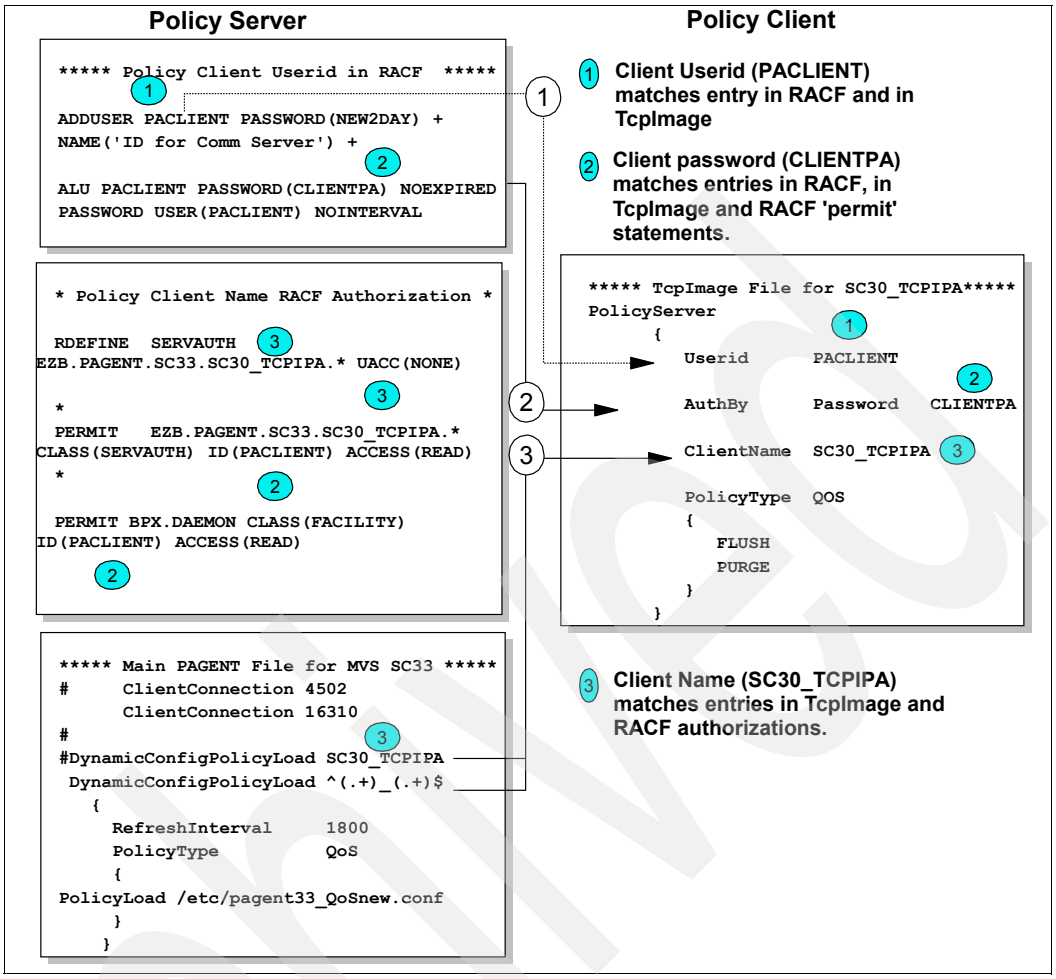


Figure 6-20 Correlation of Client Userid, Client Password, Client Name at Server and Client

Figure 6-21 shows the definitions for the SSL processing.

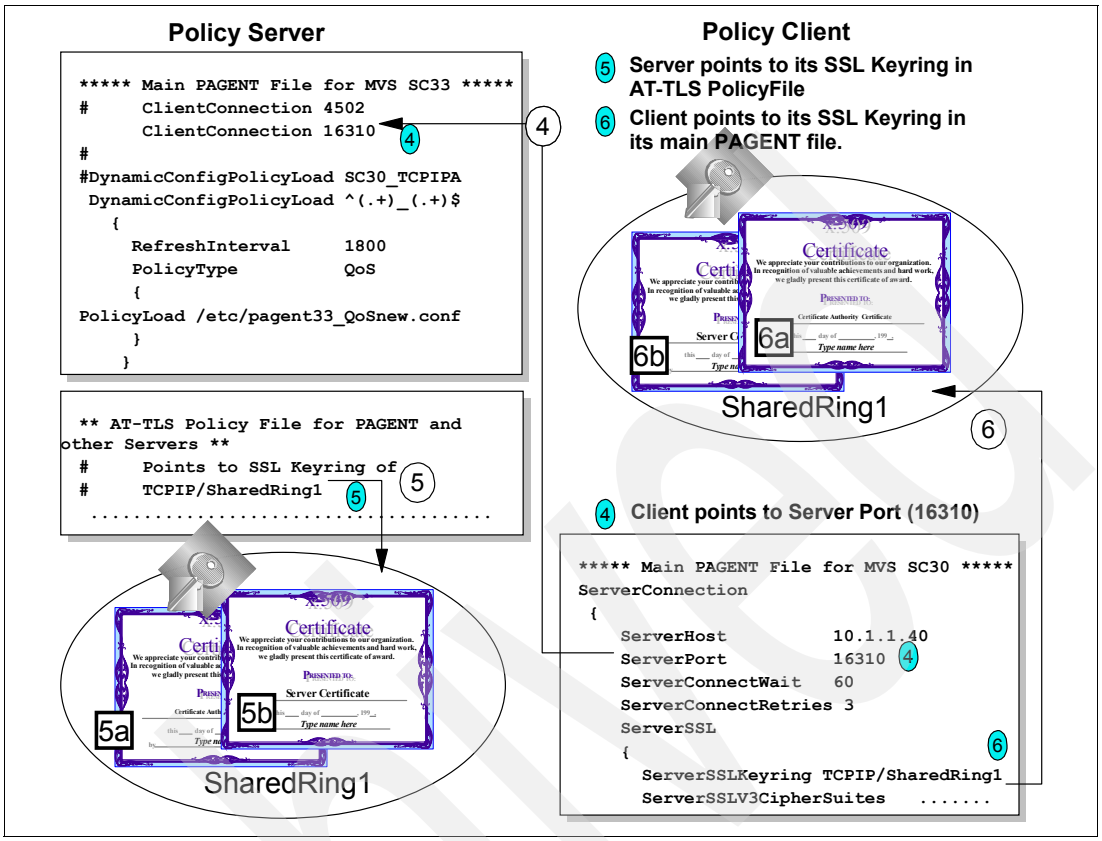


Figure 6-21 Shared key ring for the SSL connection between server and client

When the client establishes a connection to the server, it requests a connection to the server's listening port (16310) at IP address 10.1.1.40, as labeled with as *(*4) in Figure 6-21. The client requests an SSL connection implicitly; that is, there is no need for an AT-TLS policy at the client side in order to establish a secure connection with the Central Policy Server. The coding labeled as (6) identifies the client's key ring.

Note: Our implementation of the key ring at this installation uses a Shared Keyring. Thus, the server key ring and the client key ring are the same, shared with a shared RACF database across the LPARs.

The pertinent certificate for our client is the Certificate Authority Certificate (6a), because it has signed the server certificate that will provide the server authentication. However, note that because this is a Shared Keyring, the Server Site certificate (6b) is also housed in the same ring.

The server has been built with an AT-TLS policy (5) that points to a server key ring: TCPIP/SharedRing1. Inside that key ring we see that the pertinent certificate for the server is the Server Site Certificate (5b). The Site Certificate is presented to the client for server authentication. Again note that this is a shared key ring. As a result, the key ring also contains the Certificate Authority Certificate (5a) that the client uses for authentication of the server certificate.

6.4 Activating and verifying the policy services environment

We first activated the policy agent at TCPIPD, the server image, and received the messages shown in the Example 6-11.

Example 6-11 Initialization of Central Policy Server

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : QOS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING
EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER a
```

We received message EZZ8452I **a** indicating that this PAGENT is operating as a Policy Server.

The **netstat conn** command output labeled **b** in Example 6-12 reveals that PAGENT is bound to the Policy Server port that we had identified in our configuration file: 16310.

Example 6-12 PAGENT bound to the policy server listening port at TCPIPD

```
PAGENT   00005560 Listen
  Local Socket:   ::..16310  b
  Foreign Socket: ::..0
```

In Example 6-13, we started PAGENT on TCPIPA as a Central Policy Client and saw the messages that indicated that TCPIPA had retrieved the QoS policy from the server at address 10.1.1.40.

Example 6-13 Initialization of PAGENT at policy client

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8781I PAGENT CONNECTED TO POLICY SERVER FOR TCPIPA : PRIMARY AT  10.1.1.40
EZZ8790I PAGENT REMOTE POLICY PROCESSING COMPLETE FOR TCPIPA : QOS
```

Note: If TCPIPA had been unable to establish the connection to the Policy Server at TCPIPD, we would have seen the following message:

```
EZZ8780I PAGENT CANNOT CONNECT TO POLICY SERVER FOR TCPIPA : PRIMARY AT 10.1.1.40
```

The output from a **netstat conn** command at the server, shown in Example 6-14, shows that the SSL or TLS connections between Policy Server and Policy Client are indeed established.

Example 6-14 Display of policy server and policy client connection

```
EZZ8790I PAGENT REMOTE POLICY PROCESSING COMPLETE FOR TCPIPA : QOS
PAGENT   00001145 ESTBLSH a
```

```
  LOCAL SOCKET:   10.1.2.11..1915
  FOREIGN SOCKET: 10.1.1.40..16310
PAGENT   00001144 ESTBLSH b
  LOCAL SOCKET:   10.1.2.11..1914
  FOREIGN SOCKET: 10.1.1.40..16310
PAGENT   0000703E LISTEN
  LOCAL SOCKET:   ::..16310
  FOREIGN SOCKET: ::..0
PAGENT   0000B01F ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.11..1914
```

Connection ID 1144 (b) represents the connection for the data transfer back and forth between the client and the server. Connection ID 1145 (a) represents the connection used for one-way signalling between the server and the client.

We then verified from the MVS console at the server (MVS system ID of SC33) that the connections established were indeed based on an AT-TLS policy. We issued the command:

```
D TCPIP,TCPIPD,N,TTLS
```

The output from this command is shown in Example 6-15.

Example 6-15 TLS connections between the policy server and the policy client

```
D TCPIP,TCPIPD,N,TTLS
 EZD0101I NETSTAT CS V1R9 TCPIPD 536
 TTLSGRPACTION                          GROUP ID          CONNS
 GACT1~CENTRALIZED_POLICY_SERVER        00000007              2
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

We displayed the connection at the server, filtering on the address space name of PAGENT, and obtained the results shown in Example 6-16.

Example 6-16 D TCPIP,TCPIPD,N,CONN,CLIENT=PAGENT output

```
D TCPIP,TCPIPD,N,CONN,CLIENT=PAGENT
EZD0101I NETSTAT CS V1R9 TCPIPD 918
USER ID  CONN     STATE
PAGENT   00000018 LISTEN
  LOCAL SOCKET:   ::..16310
  FOREIGN SOCKET: ::..0
PAGENT   00000033 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.4.11..1028
PAGENT   00000035 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.4.11..1029
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

We looked at the basic TTLS information by issuing the **`netstat TTLS`** command with the connection identifier shown in Example 6-17.

Example 6-17 TTLS basic information about the policy server and client SSL/TLS connection

```
D TCPIP,TCPIPD,N,TTLS,CONN=33
EZD0101I NETSTAT CS V1R9 TCPIPD 927
CONNID: 00000033
```

```
  JOBNAME:      PAGENT
  LOCALSOCKET:  ::FFFF:10.1.1.40..16310
  REMOTESOCKET: ::FFFF:10.1.4.11..1028
  SECLEVEL:     TLS VERSION 1   a
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA b
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: CENTRALPOLICYRULE~1  c
  TTLSGRPACTION:  GACT1~CENTRALIZED_POLICY_SERVER
  TTLSENVACTION:  EACT1~CENTRALIZED_POLICY_SERVER
  TTLSCONNACTION: CACT1~CENTRALIZED_POLICY_SERVER
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

This example showed that in a, we negotiated for TLS V1. We saw the cipher suite selected for the exchange of data in b. Finally, we saw the name of the policy rule that applied to the connection between server and client (c). However, this example did not show us the detail about the policy. Therefore, we reissued the same **netstat TTLS** command, but we added the **DETAIL** parameter, as shown in Example 6-18.

Example 6-18 Detail of connection between policy server and policy client

```
D TCPIP,TCPIPD,N,TTLS,CONN=33,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPD 929
CONNID: 00000033
  JOBNAME:      PAGENT
  LOCALSOCKET:  ::FFFF:10.1.1.40..16310
  REMOTESOCKET: ::FFFF:10.1.4.11..1028
  SECLEVEL:     TLS VERSION 1
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: CENTRALPOLICYRULE~1
  PRIORITY:       255
  LOCALADDR:      0.0.0.0/0
  LOCALPORT:      16310
  REMOTEADDR:     0.0.0.0/0
  REMOTEPORTFROM: 1024              REMOTEPORTTO: 65535
  DIRECTION:      INBOUND
  TTLSGRPACTION:  GACT1~CENTRALIZED_POLICY_SERVER
    GROUPID:                    00000002
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
  TTLSENVACTION:  EACT1~CENTRALIZED_POLICY_SERVER
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1  d
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED e
  TTLSCONNACTION: CACT1~CENTRALIZED_POLICY_SERVER
    HANDSHAKEROLE:              SERVER
     HANDSHAKEROLE:              SERVER
```

```
     V3CIPHERSUITES:            09 TLS_RSA_WITH_DES_CBC_SHA
                                0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

In Example 6-18 on page 290 we saw the name of the key ring **d** and that client authentication was required **e**. Remember that the client was not using certificates for its authentication. The client was authenticating with user ID and password (or passticket) and additional RACF controls have been put in place for this authentication. If we display the TTLS policy content with the **pasearch -t** command at the server, the output results are nearly the same, as you see in the excerpt from the command output in Example 6-19.

Example 6-19 Excerpts from command output: pasearch -t

```
CS05 @ SC33:/u/cs05>pasearch -t

TCP/IP pasearch CS V1R9                   Image Name: TCPIPD
  Date:                 11/29/2007        Time:  18:10:18
  TTLS Instance Id:     1196375120

policyRule:             CentralPolicyRule~1
  Rule Type:            TTLS
.....................
TTLS Action:                  eAct1~Centralized_Policy_Server
.....................
TTLSKeyringParms:
   Keyring:                   TCPIP/SharedRing1    d
.....................
ClientAuthType:           Required e
   ResetCipherTimer:        0
  EnvironmentUserInstance:  0
.....................
```

In Example 6-19, we saw the name of the key ring **d** and that client authentication was required **e**.

Next we went to the UNIX shell on the server system and issued the **pasearch** command with the **-C** option; see Example 6-20.

Example 6-20 Display of local stacks and policy clients at policy server node

```
CS06 @ SC33:/u/cs06>pasearch -C

TCP/IP pasearch CS V1R9
  Date:                 11/26/2007        Time:  18:31:02

  Policy Agent image names with policies configured:
    TCPIPD
    SC30_TCPIPA
```

The example output in Example 6-20 shows that PAGENT in MVS SC33 was managing the policies for the TCPIPD local stack and for the policy client named SC30_TCPIPA.

Our next verification step was to view the names of the policy objects that PAGENT at the server node SC33 is managing for the policy client named SC30_TCPIPA. We executed the following command:

```
pasearch -c -p SC30_TCPIPA
```

The resulting output, shown in Example 6-21, revealed that the server was managing a QoS policy object on behalf of a Client (1) with the Image Name of SC30_TCPIPA (2) and with the Client Userid of PACLIENT (3). The QoS policy object that applied to this client is /etc/pagent33_QoSnew.conf (4).

Example 6-21 Verifying at the server that a specific client has retrieved policy

```
TCP/IP pasearch CS V1R9                  Image Name: SC30_TCPIPA 2
  Date:                 11/26/2007       Time:  18:35:29
  PAPI Version:         6                DLL Version:  6

Qos Policy Object:
  ConfigLocation:       Client 1         LDAPServer:        False
  ImageFileName:        /etc/pagent33_QoSnew.conf 4
  ClientUserid:         PACLIENT 3
  PolicyClientAddr      ::ffff:10.1.4.11
  PolicyClientPort:     1035
  ConnectTime:          Mon Nov 26 17:40:07 2007
  ApplyFlush:           True             PolicyFlush:       True
  ApplyPurge:           True             PurgePolicies:     True
  AtomicParse:          False            DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   True             ModifyOnIDChange:  True
  Configured:           True             UpdateInterval:    1800
  PerfColEnabled:       False
  InstanceId:           1196116808
  LastPolicyChanged:    Mon Nov 26 17:40:08 2007
```

The **pasearch -c** command at the client showed that the policy location is Remote. Example 6-22 showed that we executed the command naming the TCPIPA stack as our reference. We were running in a CINET environment and would otherwise retrieve the information about all stacks.

Example 6-22 Display of all policies at TCPIPA

```
CS05 @ SC30:/u/cs05>pasearch -c -p TCPIPA

TCP/IP pasearch CS V1R9                  Image Name: TCPIPA
  Date:                 11/27/2007       Time:  16:09:20
  PAPI Version:         6                DLL Version:  6

Qos Policy Object:
  ConfigLocation:       Remote a         LDAPServer:        False
  ClientName:           SC30_TCPIPA b
  ClientUserid:         PACLIENT c
  PolicyServerAddr      10.1.1.40 d
  PolicyServerPort:     16310 e          PolicyServSysname: SC33 f
  ConnectTime:          Tue Nov 27 12:59:51 2007
  ApplyFlush:           True             PolicyFlush:       True
  ApplyPurge:           True             PurgePolicies:     True
  AtomicParse:          False            DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   False            ModifyOnIDChange:  True
  Configured:           True             UpdateInterval:    1800
  PerfColEnabled:       False
  InstanceId:           1196186392
  LastPolicyChanged:    Tue Nov 27 12:59:52 2007


Ids Policy Object:
  ConfigLocation:       Local g          LDAPServer:        False
  CommonFileName:
```

```
  ImageFileName:
  ApplyFlush:           True              PolicyFlush:       True
  ApplyPurge:           True              PurgePolicies:     True
  AtomicParse:          False             DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   False             ModifyOnIDChange:  False
  Configured:           False             UpdateInterval:    600
..................
```

In Example 6-22 on page 292, note that the QoS policy had been retrieved from a Remote (a) location, whereas the IDS policy object was Local (g). There were many other policies available to TCPIPA, but only the QoS policy had been retrieved from the server. This meant that the TcpImage file for TCPIPA shown in Example 6-10 on page 284 had since been altered to point to various other local policy files, and that this file was not being used solely for the purposes of a policy client.

At some point an installation might decide to move many of these policies (like the IDS policy) to a central policy server and not have them retrieved locally at all.

The lines labeled as b, c, d, e, and f display the QoS Policy Objects being used to communicate with the central policy server.

6.5 Diagnosing the centralized policy services environment

If you are setting up a central policy server and client for the first time, you might want to raise the PAGENT loglevel to a higher level than you normally would use. This is especially important at the client site, because this is where you can identify whether you are actually sending your client ID and your password correctly to the server.

Any error messages that appear on the MVS system consoles need to be investigated. First you need to look up the message in the appropriate IP Messages manual. Then you need to look in the syslog daemon error log for further clues. Most installations will leave the logging level in PAGENT at loglevel of 15. This usually suffices for solving coding errors or connection errors. However, you may temporarily need to raise that logging level to 511 to obtain more granular messages.

Note: You can edit the pagent.conf file to change the loglevel. Or simply issue the **MODIFY PAGENT,LOGLEVEL,LEVEL=n** command from the MVS console. Remember to reset the loglevel to 15 after you finish diagnosing your problem or collecting data.

For example, you might find that your connections between server and client are not being established. If you look at the log file, you might find error messages indicating that the RACF authorizations are questionable.

On the other hand, if they are accurate, you would find messages like those shown in Example 6-23 on page 293.

Example 6-23 RACF authorization checking

```
INFO   :007: ......plfm_check_client_permission: Checking authorization for
'EZB.PAGENT.SC33.SC30_TCPIPA.QOS', client user name = 'PACLIENT'
LOG    :007: ......plfm_check_client_permission: After EZACDRAU call: Permission Granted
LOG    :007: .....checkClientPerm: Permission Mask = 1
```

To give another example, you might receive the message shown in Example 6-24 at the client or the server when the intended policy is not loaded.

Example 6-24 Message to indicate problems with the policy definition and load

```
EZZ8438I PAGENT POLICY DEFINITIONS CONTAIN ERRORS FOR TCPIPA : QOS
```

An examination of the syslog daemon log reveals the following information that was collected with PAGENT loglevel of 15.

Example 6-25 SYSLOGD error messages with PAGENT loglevel of 15

```
EVENT  :011: .....bind_policy_load: client PEP 'SC30_TCPIPA' doesn't match any
DynamicConfigPolicyLoad statement, using defaults for all disciplines

...
SYSERR :009: ...plfm_get_file_time: cannot open '/etc/pagent_remote.qos', errno
EDC5129I No such file or directory.

OBJERR :009: ...plfm_get_file_time: Can not get FILE handle for information:
'/etc/pagent_remote.qos'
```

The log entries in Example 6-25 tell you two things:

- The DCPL (DynamicConfigPolicyLoad) statement as coded at SC33 in the **pagent33.conf** file does not match the client image name of SC30_TCPIPA.
- The default QoS file of pagent_remote.qos cannot be found where indicated in the DCPL statement.

Upon further investigation of the DCPL statement, we discovered that the pattern we used for matching did not match SC30_TCPIPA. We enabled “hex on” in ISPF and determined that we were using the wrong code page for PAGENT; we were using US Code Page 037 instead of the required US Code Page 1047. Furthermore, we did not have the default QoS file (pagent_remote.qos) in the /etc directory. So, we corrected our code page and the file at the policy server. At the next connection between client and server, the appropriate QoS file was found and sent to the client over one of the two SSL/TLS connections.

You could also use debug level 128 on the client side (**MODIFY pagent,DEBUG,LEVEL=128**) to diagnose connection problems. Debug output goes to the log file; look for OBJERR or SYSERR messages to help identify the problem.

We cannot describe all diagnostic techniques here, but you can find many more in *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782. The chapter on Policy Agent includes tables describing the most common configuration errors and their solutions.

6.6 References

For additional information, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781
- *Enterprise Systems Architecture/390 Reference Summary*, SA22-7209

7

Quality of Service

Quality of Service (QoS) refers to a set of networking technologies intended to ensure satisfactory end-to-end application performance in the presence of network congestion, essentially by giving time-sensitive applications (such as interactive transaction processing) priority in the network over less time-sensitive applications (such as print or file transfer).

This QoS discussion could have been placed in the IBM Redbook publication *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341, because performance issues are frequently perceived as availability issues.

However, we chose to include the QoS discussion in this book, *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 4: Policy-based Network Security*, SG24-7342, because QoS is a key part of policy-based networking and is implemented in z/OS through the Policy Agent (PAGENT), discussed in Chapter 5, “Policy Agent” on page 221.

This following topics are discussed in this chapter.

Section	Topic
7.1, “Quality of Service definition” on page 296	Basic concepts of QoS
7.2, “Why QoS is important” on page 302	Key characteristics of QoS and why it may be important in your environment
7.4, “Using the IBM Configuration Assistant for z/OS Communication Server” on page 302	Configuration examples and problem determination suggestions

7.1 Quality of Service definition

Quality of Service (QoS) is not a new concept—just ask any SNA expert. In a TCP/IP context, QoS is nothing new, either. The original 1981 standard for the IP protocol, RFC 791, contained a field called Type of Service (TOS). Within this field were 8 bits representing “abstract parameters of the quality of service” for each packet.

For about the next 20 years, however, QoS was largely ignored, probably for two reasons:

- The first reason is because IP networks were popular because of their simplicity (otherwise we might still be running SNA networks)
- Secondly, networking hardware has steadily added bandwidth. Why go through the trouble of implementing QoS-based networking when you can just upgrade to a faster network and everybody is happy anyway?

With the advent of video streaming, though, QoS has become of interest again. IP routers and hosts are starting to check the ToS field of IP packets and handle according to QoS rules. And, as might be expected, the complexity we were avoiding in SNA networks is creeping into IP networks. In the 1990s, ATM networks defined a QoS architecture. This architecture helped form the basis for the modern usage of the TOS field in IP packets.

The QoS concept of a *contract* between a network user and the network, guaranteeing certain network throughput (and delay and delay variability), was implemented in TCP/IP as Integrated Services. Integrated Services is supported by the Resource Reservation Protocol (RSVP), which is used to allow an application to request (or *signal*) the network to reserve a certain amount of bandwidth with particular QoS criteria.

RSVP is defined in Internet Engineering Task Force (IETF) Internet standard RFC 2205 and can be used to provide something similar to a dedicated circuit over an IP network. Integrated Services and RSVP can provide an essential capability to support certain network applications such as high-quality, interactive, voice, or video. However, due to its complexity and the fact that adequate performance can be achieved for most applications more simply by just using prioritization, RSVP has not been widely implemented.

Short of Integrated Services, QoS may be thought of as “network prioritization done right.” Historically, organizations individually configured each router in the network to inspect and prioritize each message. As the complexity of networks and applications have increased, however, it has become increasingly difficult to individually configure each network component yet still ensure that the overall network implementation matches the desired business policies.

Consequently, organizations are now developing *enterprise-wide* QoS policies (encompassing the network and advanced servers such as zSeries® mainframes). The Differentiated Services form of QoS involves associating individual packets or flows with a particular *class of service* (not to be confused with SNA class of service) and having each node along the network path handle packets in a cooperative manner, according to a common set of rules, resulting in end-to-end service classes. Additionally, policy-based networking has emerged as a standards-based approach for defining QoS policies in one place and applying them uniformly across the entire IT environment.

7.1.1 Differentiated Services

Differentiated Services (DiffServ) was developed to allow a network to support multiple service classes without the need to maintain the state of each traffic flow along the path or to perform signaling between nodes (illustrated in Figure 7-1 on page 297). It can, therefore, scale to support the traffic seen in today’s global networks.

The network domain manager or administrator defines aggregate traffic service classes (for example, premium, gold, silver, and bronze). DiffServ is, therefore, less complex than Integrated Services. It is less network-intensive and is appropriate for networks of networks even where portions of the network are outside the control of the network domain manager.

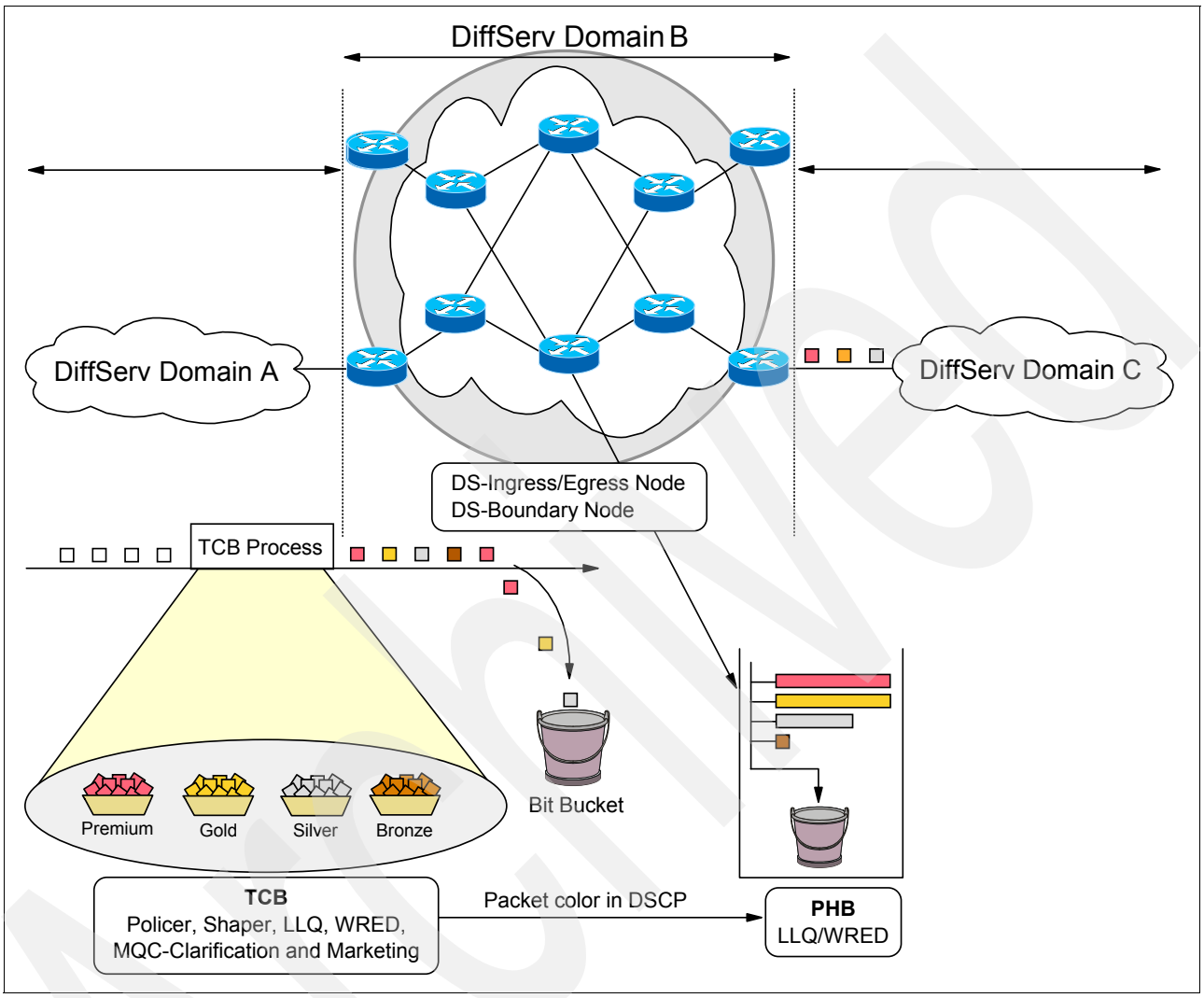


Figure 7-1 DiffServ end-to-end architecture

DiffServ is primarily described in IETF RFC 2474. But, RFC 2475, RFC 3246 and others are all part of the extended set of standards that continue to be developed in this area. DiffServ is meant to handle traffic aggregates. This means that traffic is classified according to the application requirements relative to other application traffic. Each node then handles the traffic using internal mechanisms to control bandwidth, delay, jitter, and packet loss. Through the use of standard per-hop behaviors (PHBs, RFCs 2597 and 3260), packets receive the proper handling and the result is end-to-end QoS.

For true end-to-end QoS, each administrative domain must implement cooperative policies and PHBs. Packets entering a DiffServ domain can be metered, marked, shaped, or policed to implement traffic policies as defined by the administrative authority. This is handled by the DiffServ traffic conditioner block (TCB) function.

DiffServ boundary nodes typically perform traffic conditioning. A traffic conditioner typically classifies the incoming packets into predefined aggregate classes, meters them to determine

compliance to traffic parameters, marks them appropriately by writing or re-writing the DSCP (Differentiated Services Code Point, a set of bit within the IP ToS), and finally shapes the traffic as it leaves the node.

The DS field

To distinguish the data packets from different applications in DS-capable network devices, the IP packets are modified in a subset of the IP TOS. A small bit pattern, called the *DS field*, in each IP packet is used to mark the packets that receive a particular forwarding treatment at each network node. The DS field uses part of the space of the former TOS octet in the IPv4 IP header and the traffic class octet in the IPv6 header. All network traffic inside of a domain receives a service that depends on the traffic class that is specified in the DS field.

The DS field uses 6 bits to determine the Differentiated Services Code Point (DSCP) as defined in RFC 2474 and RFC 2475. This code point will be used by each node in the net to select the PHB. A 2-bit currently unused (CU) field is reserved. The values of the CU bits are ignored by DS-compliant nodes when PHB is used for received packets. Figure 7-2 shows the structure of the defined DS field.

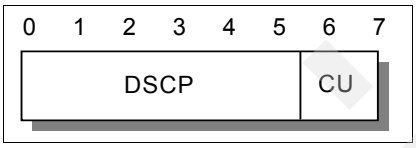


Figure 7-2 DS field

In the event that some nodes in a network recognize only the IP precedence bits, standard DSCP PHBs are constructed in such a way that they remain compatible with IP precedence. For example, the DSCP values can be used such that the values for IP precedence relate to the classes, as shown in Table 7-1.

Table 7-1 Relationship between IP precedence and DSCP

RFC 791 precedence		**RFC 2474, RFC 2475 DiffServ**	
Network Control	111 (7)	Preserved	111000
Internetwork Control	110 (6)	Preserved	110000
CRITIC/ECP	101 (5)	Express Forwarding	101xxx
Flash Override	100 (4)	Class 4	100xxx
Flash	011 (3)	Class 3	011xxx
Immediate	010 (2)	Class 2	010xxx
Priority	001 (1)	Class 1	001xxx
Routine	000 (0)	Best Effort	000000

7.1.2 QoS with z/OS Communications Server

In the z/OS Communications Server environment, support for Integrated Services is provided by the RSVP Agent. The RSVP Agent queries the Policy Agent for relevant information and communicates with the network to request the desired QoS on behalf of the application.

Differentiated Services is supported by the PAGENT. PAGENT gets policy definitions from a local configuration file or a Lightweight Directory Access Protocol (LDAP) server. PAGENT then installs the policies in the z/OS Communications Server stacks as desired.

Figure 7-3 shows the relationship between the various z/OS QoS components. Tasks or daemons such as PAGENT and RSVPD work together and with the TCP/IP protocol stack to classify and mark packets for QoS. Data collection points are also available for performance management.

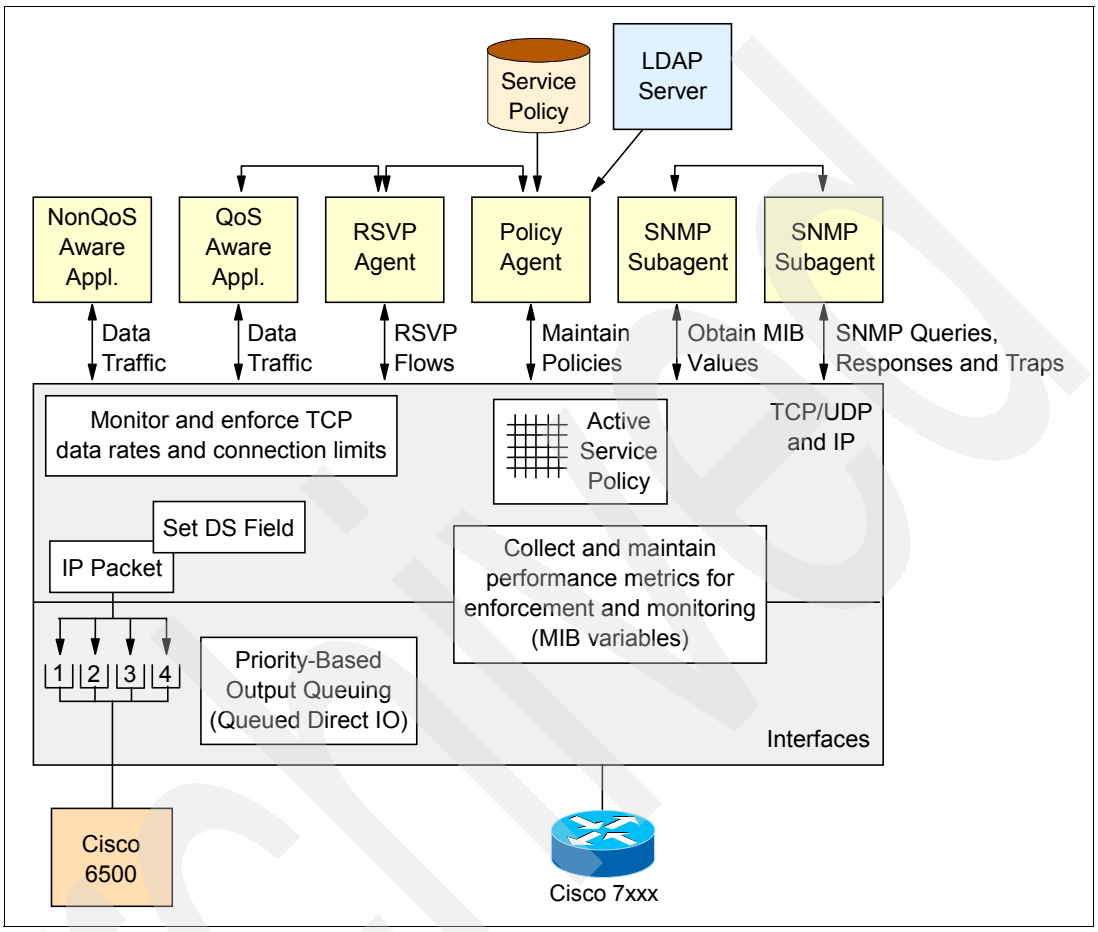


Figure 7-3 z/OS CS Quality of Service components

Note: Without a cooperative framework of host-based components and QoS mechanisms within the network (*and* the necessary interorganzational coordination), it is impossible to establish and implement end-to-end service levels. It could well happen that you set up the policies and go through the effort to set the DS field in messages, only to have the network overwrite your settings with its own.

7.1.3 PAGENT QoS policies

We suggest, when you first implement QoS policies, that you start with a small number of critical applications or traffic types. Then, as you develop more knowledge of the traffic patterns and interactions, continue to apply a set of service classes to applications or traffic streams as needed.

The PAGENT supports the following QoS policies:

- Differentiated Services (DS) policies
- Integrated Services (RSVP) policies
- Sysplex Distributor (SD) policies

Note: Sysplex Distributor policies are discussed in *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341.

Policy conditions consist of a variety of selection criteria that act as traffic filters. Traffic can be filtered based on source/destination IP addresses, source/destination ports, protocol, inbound/outbound interfaces, application name, application-specific data, or application priority. Only packets that match the filter criteria are selected to receive the accompanying action. Policy rules can refer to several policy actions, but only one policy action is executed per policy scope. A given policy action may be referred to by several policy rules.

Differentiated Services policies

Policies to be implemented can be configured via the Policy Agent configuration file, in an LDAP server, or both. After being read, the policies are combined into a single list. Policy rules and actions map subsets of outbound traffic to various QoS classes, and can be used to create end-to-end Differentiated Services.

Setting DSCP using the Policy Agent

PAGENT policies are defined by rules and actions. The rules consist of a variety of selection criteria to provide a *match condition*. Matching the *rule* then forces the *action*. One particularly important action is the setting of the DS field. Outbound traffic can be marked with the desired Differentiated Services Control Point (DSCP) value. This marking will then be interrogated by the network and the appropriate per-hop behavior (PHB) applied as the packet traverses the network.

Integrated Services (RSVP) policies

Given the narrow applicability of Integrated Services and RSVP, they are not covered in this book.

7.1.4 Configuring QoS in the z/OS Communications Server

The two components responsible for QoS within the z/OS Communications Server are the Policy Agent and the RSVP Agent. In this section, we provide an overview of the configuration steps necessary to use the z/OS CS Policy Agent for QoS. PAGENT runs in the z/OS environment and reads policy definitions from a local configuration file or a central repository that uses the LDAP.

Note: There are two graphical user interfaces (GUI) for configuring the QoS policies:

- zQoS Manager for configuring QoS GUI - this GUI was used in older versions of Communications Server and uses the LDAP server.
- IBM Configuration Assistant for z/OS Communication Server GUI - this is the new GUI which generates a flat configuration file and does not need the LDAP server.

In this chapter, we explain how to use the IBM Configuration Assistant for z/OS Communication Server.

After the policies are updated, PAGENT installs policies in one or more z/OS CS stacks, replacing existing policies or updating them as necessary.

Note: The IBM Configuration Assistant for z/OS Communication Server GUI is intended to replace the zQoS Manager for configuring QoS.

Policies

Policies consist of several related objects. The main object is the *policy rule.* A policy rule object refers to one or more *policy conditions*, *policy actions*, or *policy time period condition objects*, and also contains information about how these objects are to be used. Policy time period objects are used to determine when a given policy rule is active. Active policy objects are related in a way that is analogous to an IF statement in a program. For example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a policy rule are TRUE, then the policy actions associated with the policy rule are executed.

Differentiated Services rule

The most common QoS deployment uses rules to map outbound traffic from particular applications into subclasses. Example 7-1 illustrates this type of policy. The goal of this priority control policy is to map a subset of the traffic outbound from an FTP server.

Example 7-1 Sample Priority_Control rule for FTP

```
policyRule Priority_Control~9
{
   PolicyRulePriority        64920
   DestinationAddressRange   192.168.1.254 1
   SourcePortRange           21 1
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~3
}

policyRule Priority_Control~10
{
   PolicyRulePriority        64910
   DestinationAddressRange   192.168.1.254 1
   SourcePortRange           20 1
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~3
}
PolicyAction action~3
{
   PolicyScope               DataTraffic
   OutgoingTOS               01000000 2
}
```

This policy is identified as a Differentiated Services policy by the PolicyScope DataTraffic attribute on the PolicyAction statement, as well as the use of several DS-only attributes.

The following statements apply to Example 7-1:

1 The policy rules selects traffic originated by port 21 for TCP (FTP outbound data connection uses port 20) from the IP address 192.168.1.254.

2 The policy action specifies that the TOS byte be set to '01000000' for traffic that conforms to this policy.

7.1.5 For additional information

For additional information about these topics, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776

7.2 Why QoS is important

Over the past decade, the amount of available network bandwidth has increased almost exponentially while bandwidth costs have declined almost as dramatically. Yet, still, bandwidth is not free (nor is it equally available in all locations). Consequently, organizations must strive to provide required application performance in the face of constrained network capacity. The best way to do so is to understand the service levels required for each type of traffic in the network and prioritize that traffic accordingly. QoS, along with policy-based networking, provides the facilities to do that prioritization consistently and end-to-end, across the entire IT environment.

7.3 Install the Policy Agent

The Policy Agent reads the configuration files that contain the QoS policy configuration statements, checks them for errors, and installs them into the TCP/IP stack. Setting up the PAGENT is described in Chapter 5, “Policy Agent” on page 221.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an APF-authorized library.

After setting up the PAGENT, you need to define the QoSConfig statement to specify the path of the policy file that contains stack-specific QoS policy statements to PAGENT. Example 7-2 shows the QoS statements in the TCP/IP stack configuration file used by Policy Agent.

Example 7-2 The pagent /etc/sc30.tcpipa_image.conf file with QoS configured

```
# This is a file that contains statements for TCP/IP stack TCPIPA in z/OS image SC30
# This file is used by Policy Agent.
#
# QoSConfig Statements
QoSConfig //'TCPIP.SC30.POLICIES(QOSA)'
```

7.4 Using the IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server is discussed in the following sections:

- IBM Configuration Assistant for z/OS Communication Server
- Download and installation
- Using the GUI

IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server enables centralized configuration of Quality of Service policies for z/OS. The IBM Configuration Assistant for z/OS Communication Server helps a network administrator produce a flat QoS policy file.

The IBM Configuration Assistant for z/OS Communication Server is a tool for network administrators. Therefore, before you begin, read the chapter on policy-based networking in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. Also, be familiar with your particular environment so that you can make decisions about what events are to be detected under what circumstances and the appropriate actions to take.

Download and installation

The download and installation instructions are written for Windows. The information and executable in the following sections are also located at:

`http://www-1.ibm.com/support/search.wss?rs=852&tc=SSSN3L&rankprofile=8&dc=D410&dtm/`

Using the GUI

This section is intended to help the network administrator manage and understand the GUI provided. The first-level directories are:

- z/OS host information
- QoS policy rules

Open the IBM Configuration Assistant for z/OS Communication Server.

Note: IBM Configuration Assistant for z/OS Communication Server Help is available via the Help button. If detailed information is needed for a particular field, click the **?** button and then click the desired field.

In the Welcome panel, select **Quality of Service (QoS)**, as shown in Figure 7-4 on page 304. Click **OK**.

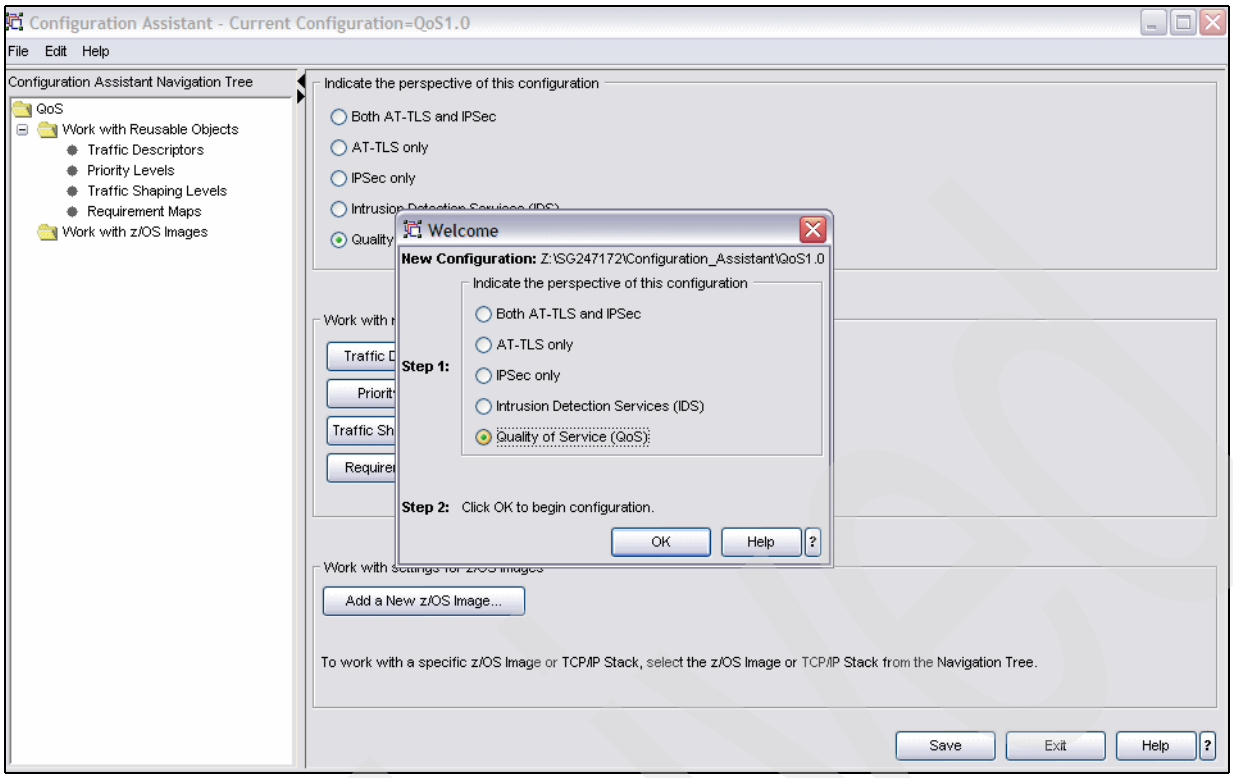


Figure 7-4 Welcome panel: Selecting Quality of Service policy

7.4.1 z/OS host information

This section provides information that will be included in PAGENT's configuration file on the z/OS host. The information that you provide will depend upon your policies. The factors to consider are:

- Should policies be applied to Sysplex Distributor?
- Are you using OSA Express cards in QDIO mode?
- Are you running with multiple TCP/IP stacks on this LPAR? And if so, to which instances of TCP/IP should your policies be applied?

Adding z/OS image to the policy

To add z/OS image to the policy:

1. From the Configuration Assistant Navigation Tree, right-click **QoS** → **Work with z/OS Images** and click **Add new z/OS Image** from the menu, as shown in Figure 7-5 on page 305.

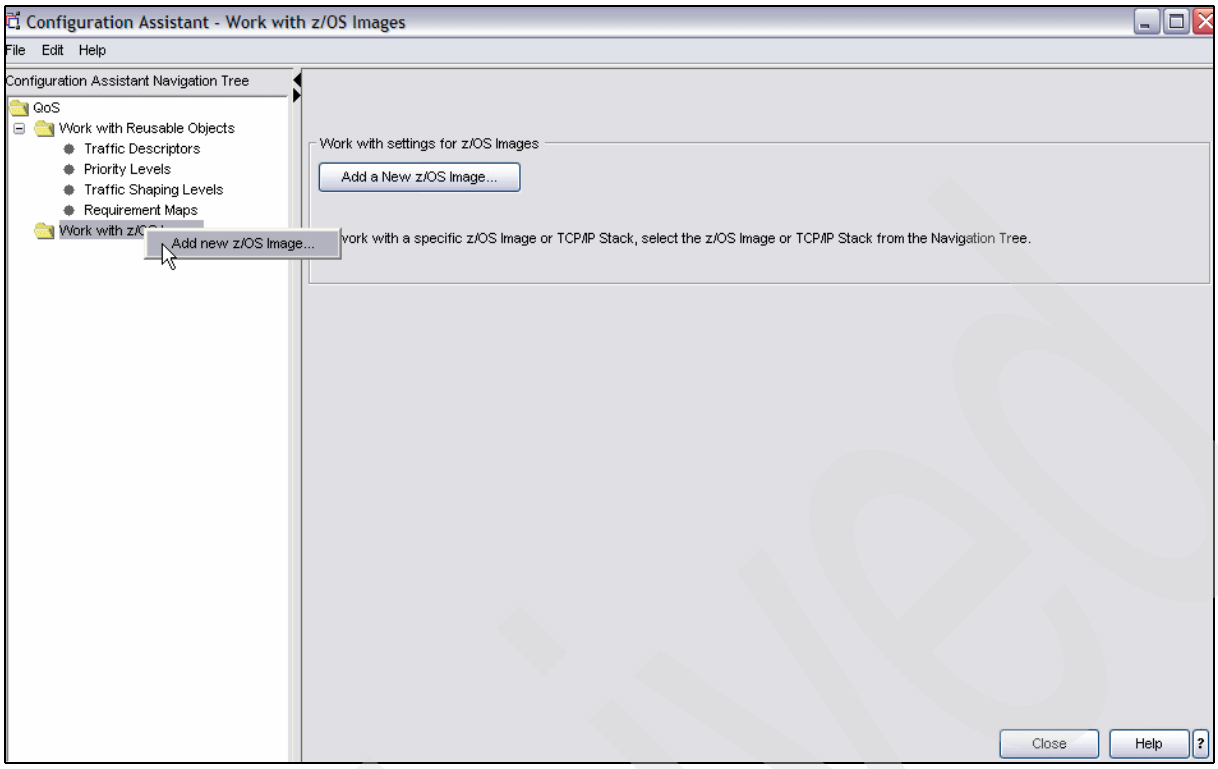


Figure 7-5 Work with z/OS Images panel: Adding a new z/OS image to the QoS policy

2. In the New z/OS Image: Information panel, enter the z/OS image name and description. (In our case, we added SC30, as shown in Figure 7-6.) Click **OK**.

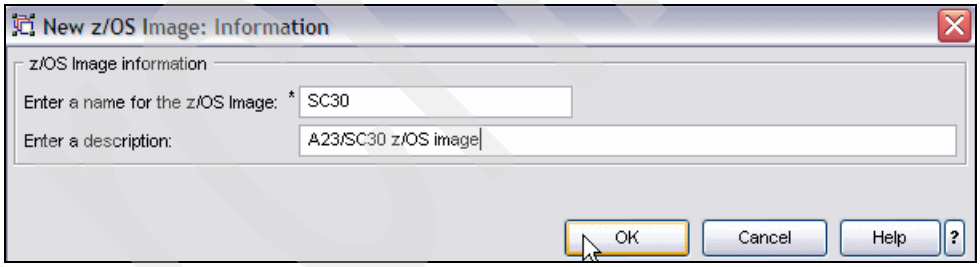


Figure 7-6 New z/OS Image: Information panel: Defining SC30 z/OS image

3. In the Proceed to the next step? panel, click **Yes**; see Figure 7-7.

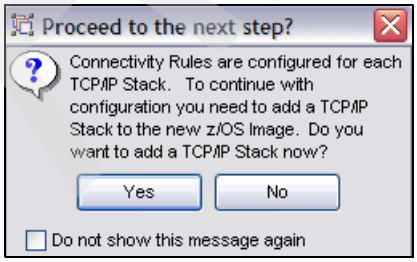


Figure 7-7 Proceed to the next step? panel

Adding TCP/IP stack to the policy

To add TCP/IP stack to the policy:

1. In the New TCP/IP Stack Wizard: Welcome panel, click **Next**.
2. In the New TCP/IP Stack: Name panel, enter the name of the TCP/IP stack and a description, as shown in Figure 7-8. Click **Next**.

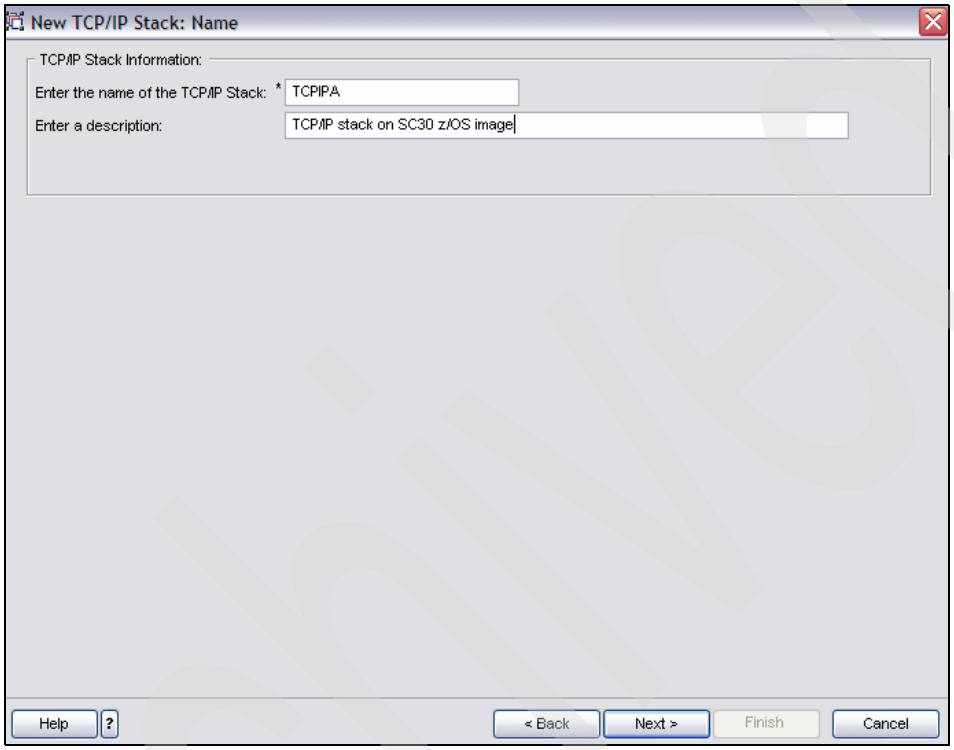


Figure 7-8 New TCP/IP Stack: Name - adding TCPIPA stack

3. In the New TCP/IP Stack Wizard: Sysplex Distributor panel, indicate if the stack will use QoS for Sysplex Distributor. In our configuration, we chose not to use QoS for Sysplex Distributor. Select **This Stack will NOT use QoS for Sysplex Distributor**, as shown in Figure 7-9. Click **Next** and then click **Finish**.

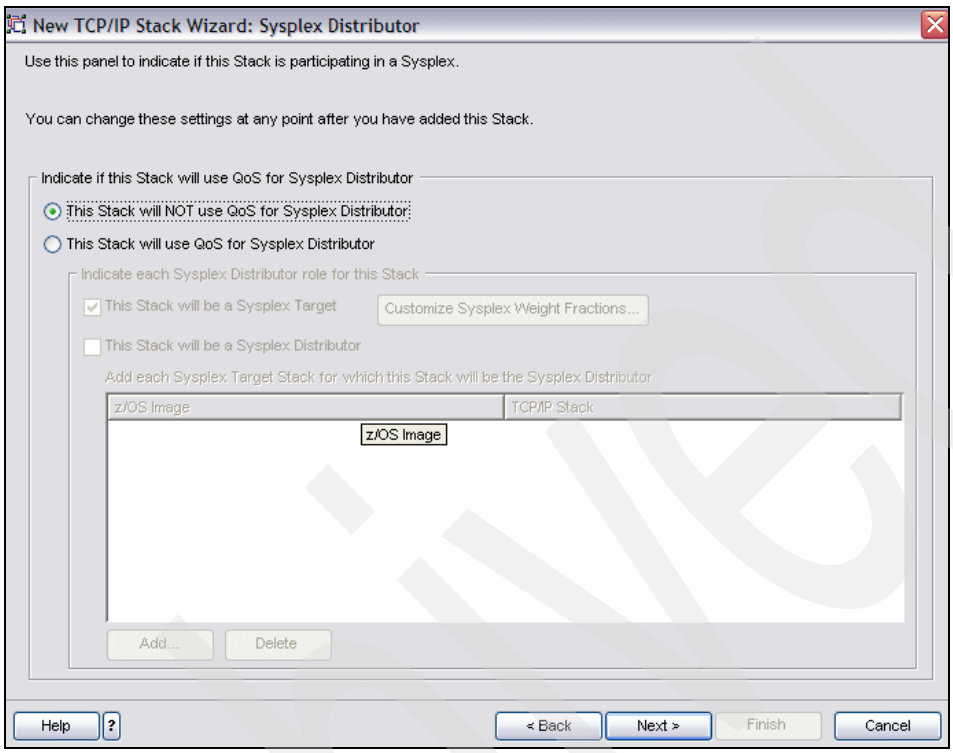


Figure 7-9 New TCP/IP Stack Wizard: Sysplex Distributor panel

4. In the Proceed to the next step? panel, click **No**.

QoS performance logging

Performance logging is used to collect QoS performance monitoring data. The performance data can be collected on a policy rule or action or on both rules and actions. Policy Rules map to the GUI's connectivity rules and traffic descriptors. Policy actions map to the GUI's priority levels and traffic shaping levels. The collected data can also be logged to a specified performance log file for offline collection and monitoring by a user application, or can be accessed in near real time using the Policy API (PAPI).

From the Configuration Assistant Navigation Tree, select **QoS** -> **Work with z/OS Images** -> **Image - *image_name*** -> **Stack - *stack_name***.

In the Configuration Assistant - TCP/IP Stack Settings panel, select the **QoS Performance Logging** tab and select **Enable QoS Performance Logging** check box. In addition, enter a file name for the performance log file, as shown in Figure 7-10 on page 308. Click **Apply Changes**.

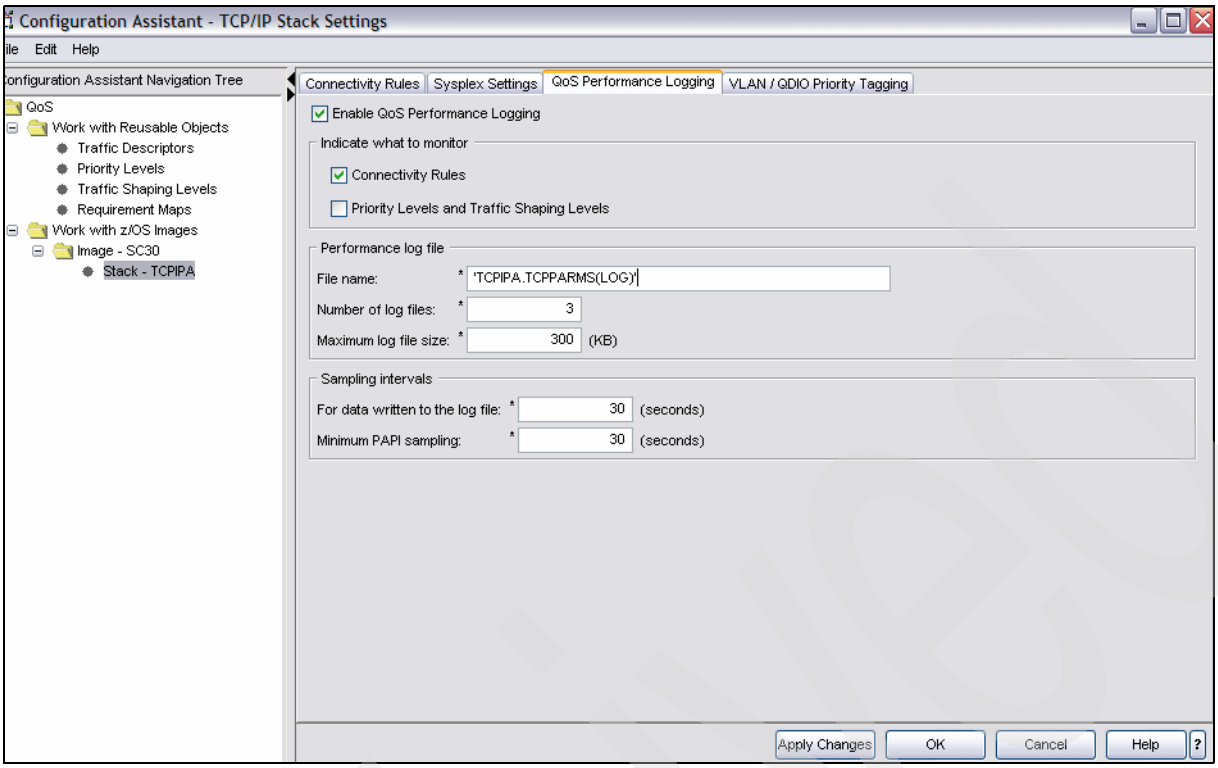


Figure 7-10 QoS Performance Logging tab: Enabling performance logging

VLAN/QDIO priority tagging

This panel is used to set specific parameters for Virtual LAN (VLAN) and device priority tagging.

- **VLAN priority tagging**: This function allows you to correlate the TOS byte in outgoing IP packets to a priority in LAN frames according to IEEE 802.1Q specifications. Specifically, it sets the packet priorities for switches.
- **Device priority tagging**: This function allows you to correlate the TOS byte in outgoing IP packets to a priority for interfaces that use OSA-Express configured in QDIO mode.

1. From the Configuration Assistant Navigation Tree, select **QoS** -> **Work with z/OS Images** -> **Image - *image_name*** -> **Stack - *stack_name***.
2. In the Configuration Assistant - TCP/IP Stack Settings panel, select the **VLAN/QDIO Priority Tagging** tab and select the **Enable priority tagging** check box.

   There are two options for the priority tagging:

   – Enabling the priority logging for all interfaces
   – Enabling the priority logging for specific interfaces

   We chose the first option, so verify that **Enable for all interfaces** is selected. (If you want to specify specific interfaces, select **Enable for only the following interfaces** and click **Add**.)
3. In the VLAN Tagging Settings for Single Interface panel, enter the local interface and then click **OK**. This can be an IPv4 address or an interface name. If an interface name is specified, it must match a name specified on one of the following statements in the TCP/IP profile:

   – LINK statement for an IPv4 interface
   – INTERFACE statement for an IPv6 interfaces

4. In the Configuration Assistant - TCP/IP Stack Settings panel, click **OK**, as shown in Figure 7-11. Click **Apply Changes**.

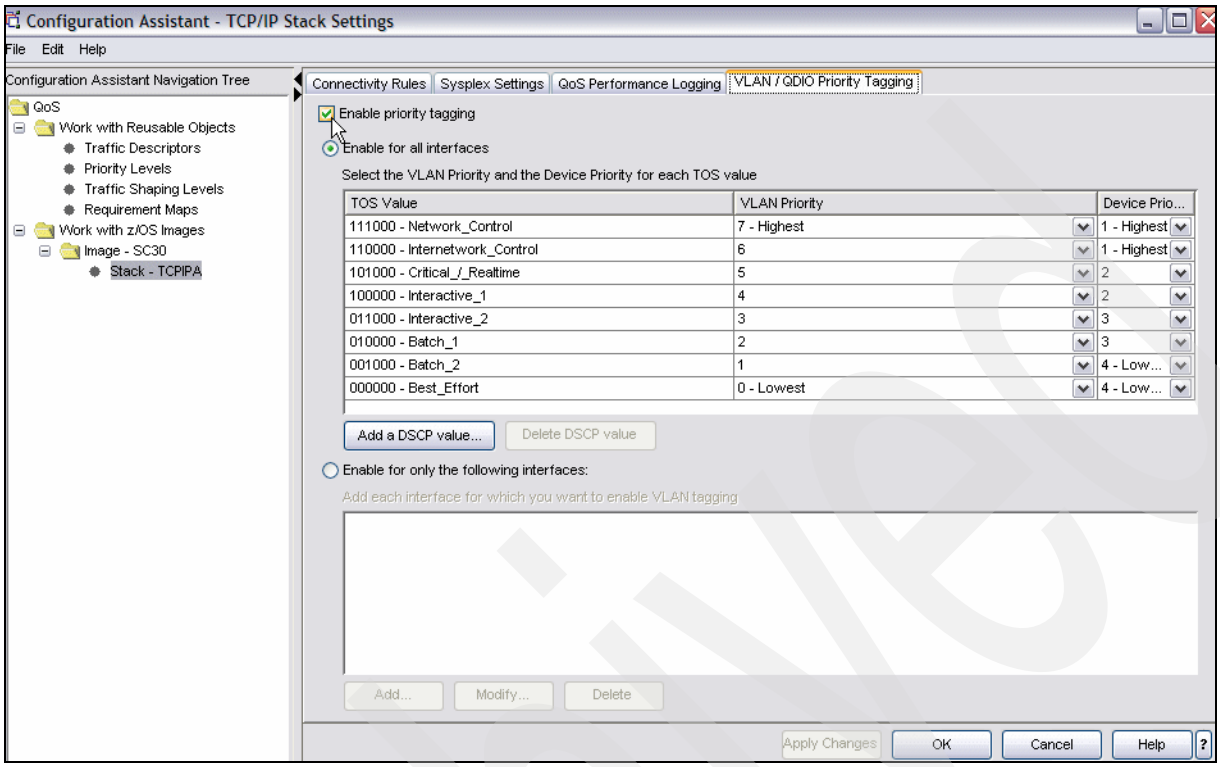


Figure 7-11 TCP/IP Stack Settings panel: After enabling priority tagging for all interfaces

7.4.2 QoS policy rules

Specifying the QoS policy rules is the most critical task and typically will be done iteratively until the final policy rules are accepted:

- You are required to define the data endpoints and to use one requirement map in at least one policy rule.
- You may optionally specify that rules apply only during validity periods.

Use this section to specify QoS policy rules. When you have finished specifying QoS policy rules, right-click **QoS** -> **Work with z/OS Images** -> **Image - *image_name*** -> **Stack - *stack_name*** and select **Install Configuration Files** from the menu.

Creating QoS policy rules

The Connectivity Rules tab under Configuration Assistant - TCP/IP Stack Settings panel (shown in Figure 7-12) enables you to create a policy rule by linking together policy requirement map objects with data endpoints and validity periods (time periods when the policy condition will be active). You also have the option of identifying whether this particular rule will be used by the Sysplex Distributor for load distribution. However, before you get to this stage, you must create the QoS requirement maps and define their validity ranges.

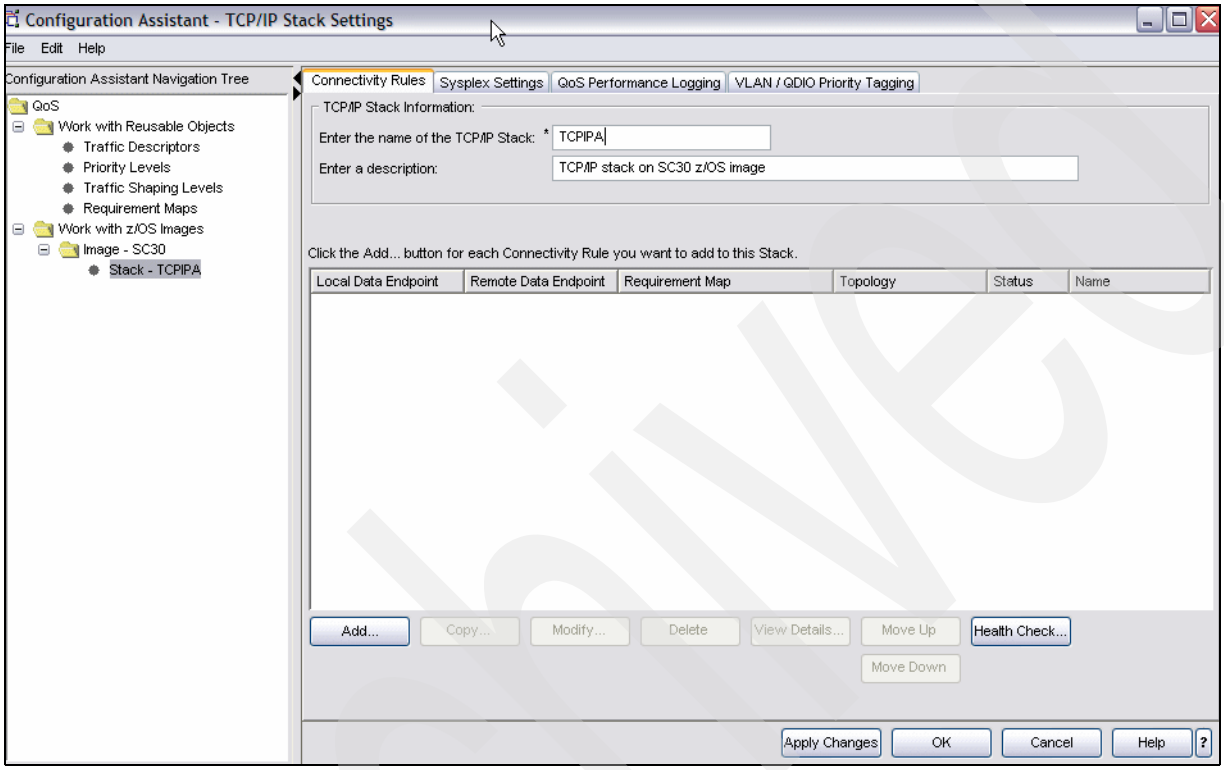


Figure 7-12 Connectivity rules panel

QoS requirement maps

Like the Connectivity Rules panel, the Configuration Assistant - Requirement Maps panel (shown in Figure 7-13) links together sets of information. In this case, these are all actual conditions that you will want PAGENT to check for.

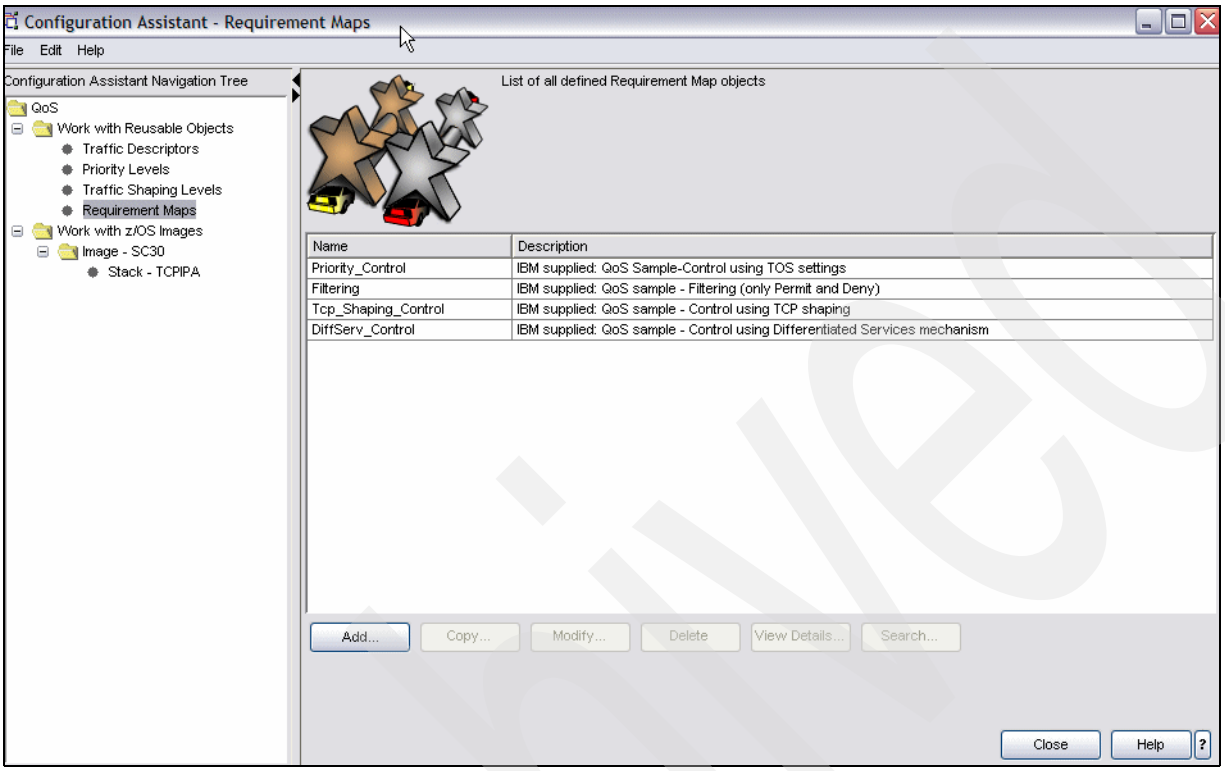


Figure 7-13 Configuration Assistant - Requirement Maps panel

Creating a requirement map object

When adding a new requirement map, each row of the Requirement Map table is a mapping between a traffic descriptor and an action. For QoS, the actions are a priority level and a traffic shaping level. Each mapping characterizes a level of service (QoS) for a type of traffic. Here we define what should happen to a packet if it matches a particular traffic descriptor.

- **Priority levels** - are objects which characterize different ways to prioritize network traffic. Network traffic is prioritized by specifying a specific value for the IPv4 Type-of-Service (TOS), the IPv6 Traffic Class value, or Differentiated Services Code Point (DSCP) field to be set in the outgoing IP packets.
- **Traffic shaping levels** - are objects which characterize different ways to control the levels of traffic. Traffic shaping levels can control TCP connections, TCP throughput, and Token Bucket traffic policing.

To add a requirement map object:

1. From the Configuration Assistant Navigation Tree, select **QoS** -> **Work with Reusable Objects** -> **Requirement Maps**.
2. In the Configuration Assistant - Requirement Maps panel, click **Add**.
3. In the Requirement Map panel, enter a name and description. Select a traffic descriptor from the Objects list and click **Add**. For example, we chose **FTP-Client** and **FTP-Server**, as shown in Figure 7-14 on page 312. We left the default priority level and traffic shaping level.

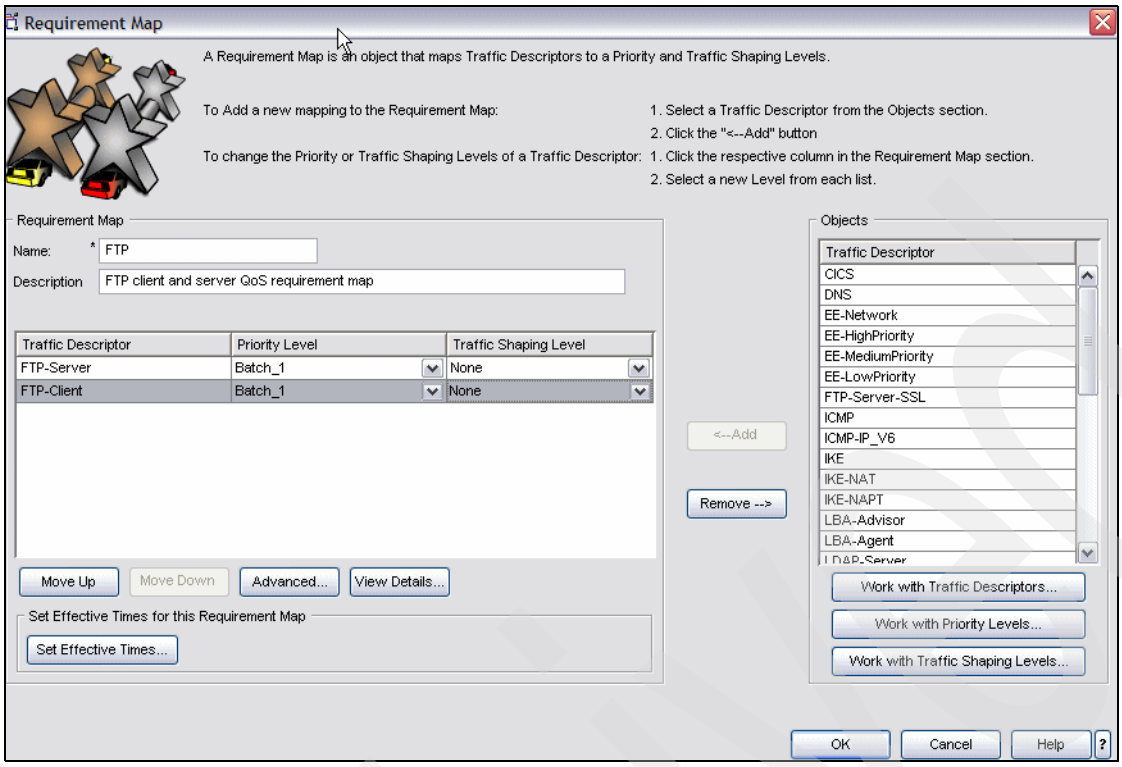


Figure 7-14 Requirement Map panel: adding a requirement map for FTP

4. In the Requirement Map panel, you can click **Set Effective Times** to set the periods of time on which the requirement map should be active. After you finish setting the effective times, then in the Effective Times panel, click **OK**.
5. In the Requirement Map panel, click **OK**.

Note: Be aware that these requirement map objects are reusable and as such may be included in multiple QoS connectivity rules.

QoS connectivity rules

Now that you have created your requirement map objects, the next step is to add connectivity rules. This is done through the Connectivity Rules panel. Here we define a rule for data endpoints and a specific requirement map object. We chose to use one of the built-in objects.

To add a connectivity rule:

1. From the Configuration Assistant Navigation Tree, select **QoS** -> **Work with z/OS Images** -> **Image -** ***image_name*** -> **Stack -** ***stack_name***.
2. From the Configuration Assistant - TCP/IP Stack Settings panel, verify that you are working on **Connectivity Rules** tab and click **Add**.
3. In the Connectivity Rules: Data Endpoints panel, identify the data endpoints for the rule. For example, we defined the rule from **All IPv4 addresses** to 192.168.1.254, as shown in Figure 7-15 on page 313. Click **Next**.

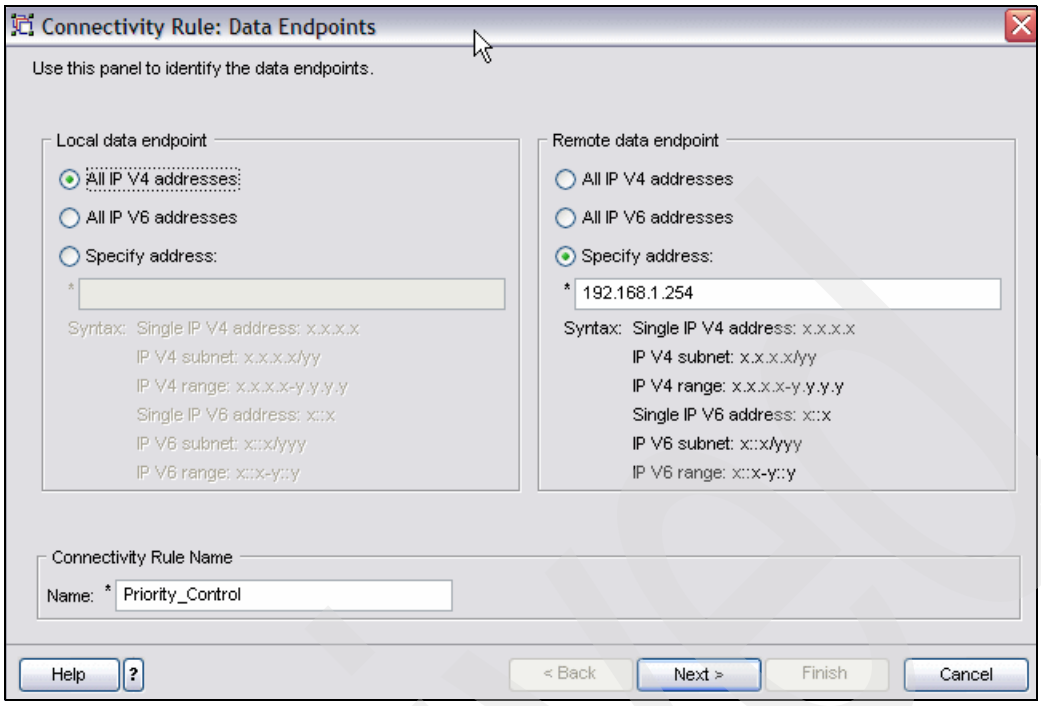


Figure 7-15 Connectivity Rule: Data Endpoints panel - identifying the data endpoints

4. In the Requirement Map: Select Requirement Map panel, select the requirement map that you want to use in this rule. We chose one of the built-in requirement map objects, **Priority_Control**, as shown in Figure 7-16.

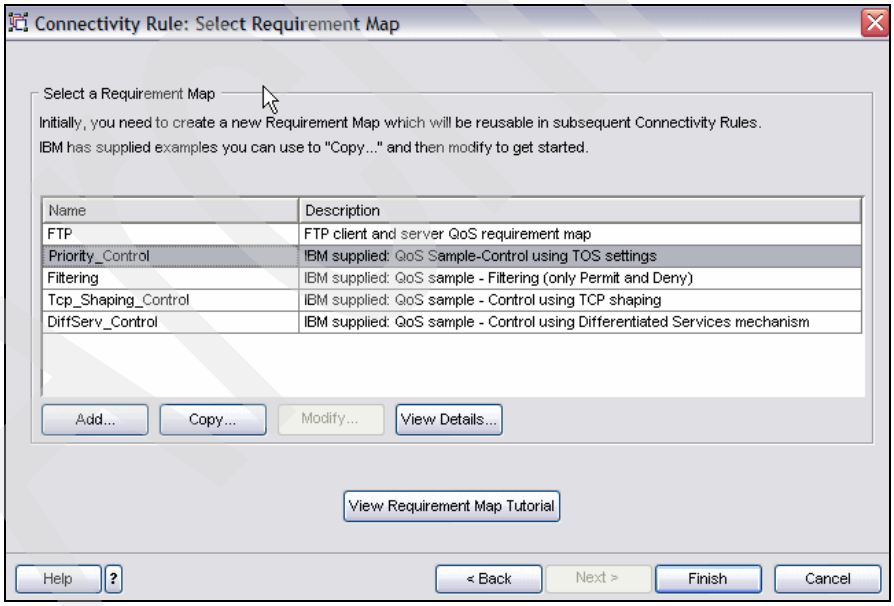


Figure 7-16 Connectivity Rule: Select Requirement Map - stack does not participate in sysplex

5. If the TCP/IP stack that you are configuring now is not participating in a sysplex, then click **Finish**, as shown in Figure 7-16.

   If the stack is configured to participate in a sysplex, click **Next**, as shown in Figure 7-17 on page 314.

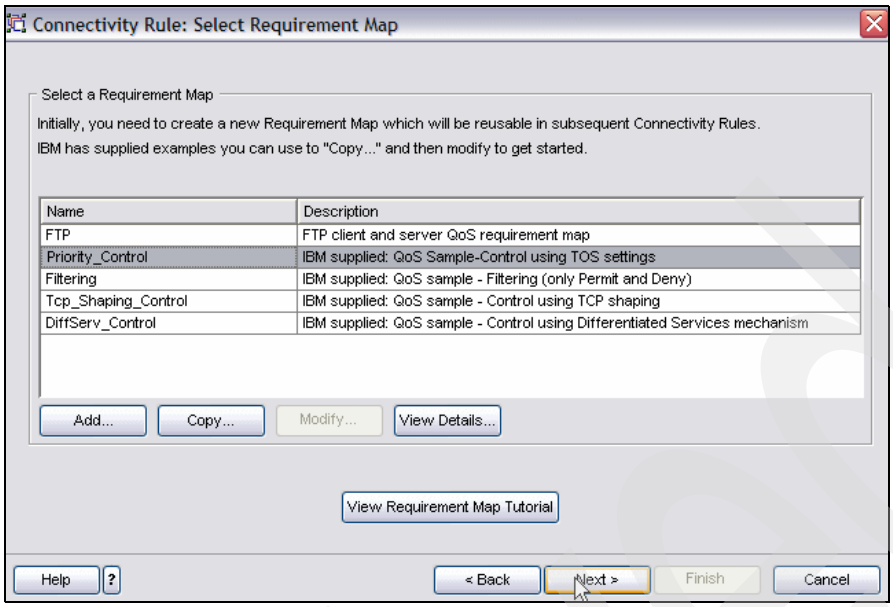


Figure 7-17 Connectivity Rule: Select Requirement Map - stack participates in a sysplex

6. This step is relevant only if the stack is participating in a sysplex.
   a. In the Connectivity Rule: QoS Sysplex Target Role panel, indicate if the rule is for a QoS sysplex target by selecting the **This rule is for a QoS sysplex target** check box. In addition, indicate if this rule should be a primary or secondary target of the Sysplex Distributor. If you select **This Stack will be a primary target for applications specified in this Connectivity Rule:**, click the **Add** button to add each local interface that should be a primary target. Click **Next**.
   b. In the Connectivity Rule: Additional Settings panel, you may click **Advanced** to modify the QoS sysplex distributor settings. After you finish modifying, click **OK**.
   c. In the Connectivity Rule: Additional Settings panel, click **Finish**.

Note: The actual setup of Sysplex Distributor for high availability and workload balancing is discussed in detail in Chapter 6, “Internal application workload balancing,” of *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7341. In that book, however, a simple configuration flat file is used to define Sysplex Distributor policies to PAGENT.

If you need to create more complex or dynamic Sysplex Distributor policies, use the IBM Configuration Assistant for z/OS Communication Server. Some of the following panels illustrate the use of IBM Configuration Assistant for z/OS Communication Server Manager for Sysplex Distributor policies.

Connectivity rule priorities

A connectivity rule object refers to one requirement object, which refers to one or more traffic descriptors, priority levels, and traffic shaping levels. Validity periods determine when each requirement map object is active. Active requirement map objects are analogous to an IF statement in a program; for example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a requirement map on a connectivity rule are TRUE, then the traffic shaping levels associated with the requirement map object are executed. Only one connectivity rule and associated actions can be applied to a particular packet. The prioritization of the policy can be seen when you add a connectivity rule and use the **Move Up** and **Move Down** buttons, as shown in Figure 7-18.

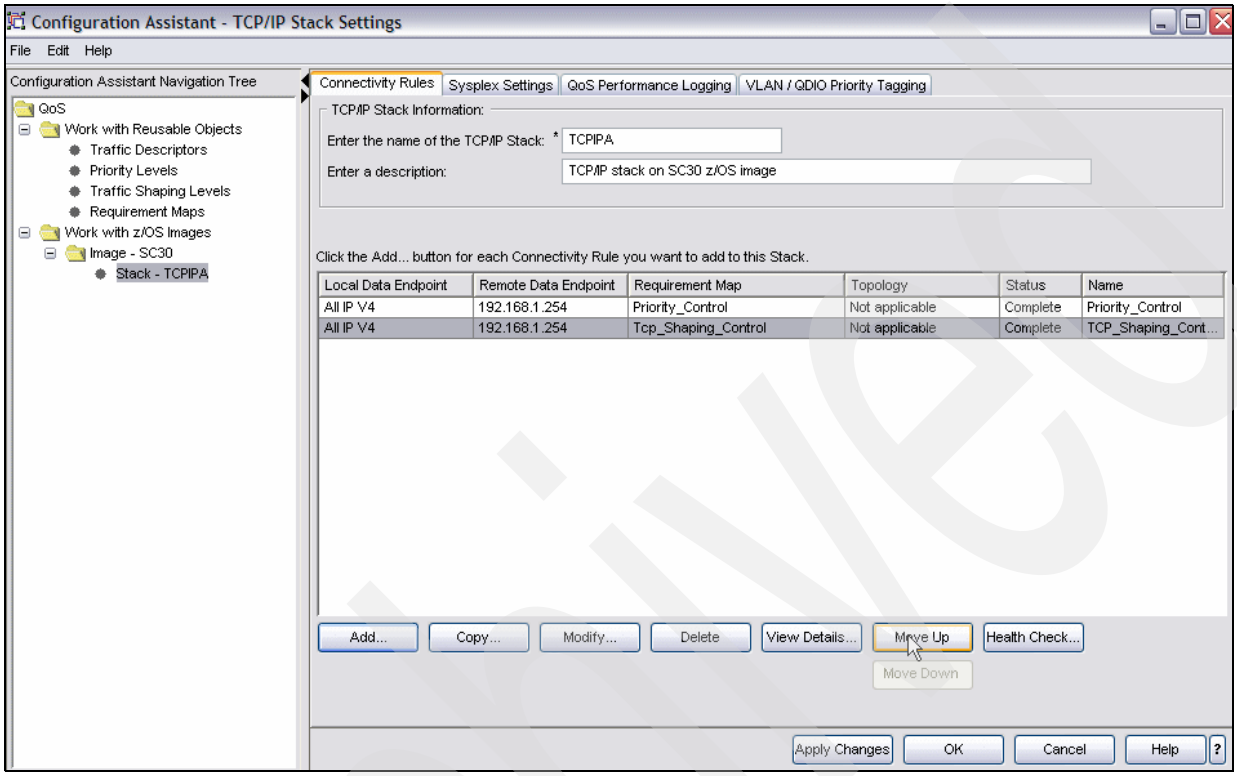


Figure 7-18 Configuration Assistant - TCP/IP Stack Settings panel: Move Up button

The first connectivity with a true condition will be executed. Thus, the prioritization of connectivity rules must be evaluated prior to implementation. You can easily prioritize a connectivity rule by clicking a connectivity rule in the right pane and clicking the **Move Up** or **Move Down** buttons.

FTP the QoS policy to the z/OS image

When you finish defining the QoS policy, FTP it to the z/OS image.

1. From the Configuration Assistant Navigation Tree, right-click **QoS** -> **Work with z/OS Images** -> **Image - *image_name*** -> **Stack - *stack_name*** and select **Install Configuration Files**, as shown in Figure 7-19 on page 316.

> **Tip:** Click the **Health Check...** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The Health Check reports on errors in your policy and we found it helpful.

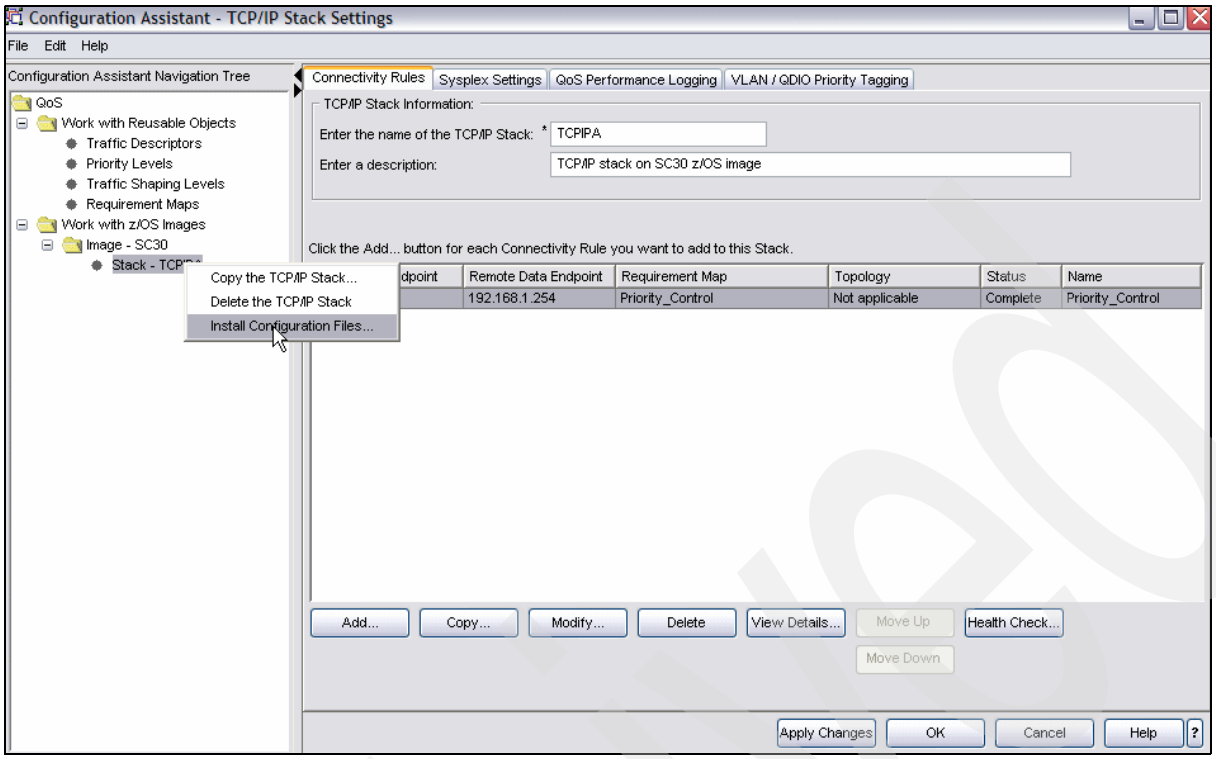


Figure 7-19 Configuration Assistant - TCP/IP Stack Settings panel: Installing the configuration files

2. If you are asked to apply changes in the Do you want to Apply Changes? panel, then click **Apply Changes**.
3. In the Installation - Stack = “stack_name” panel, select the configuration file from the list, as shown in Figure 7-20. Click **FTP**.

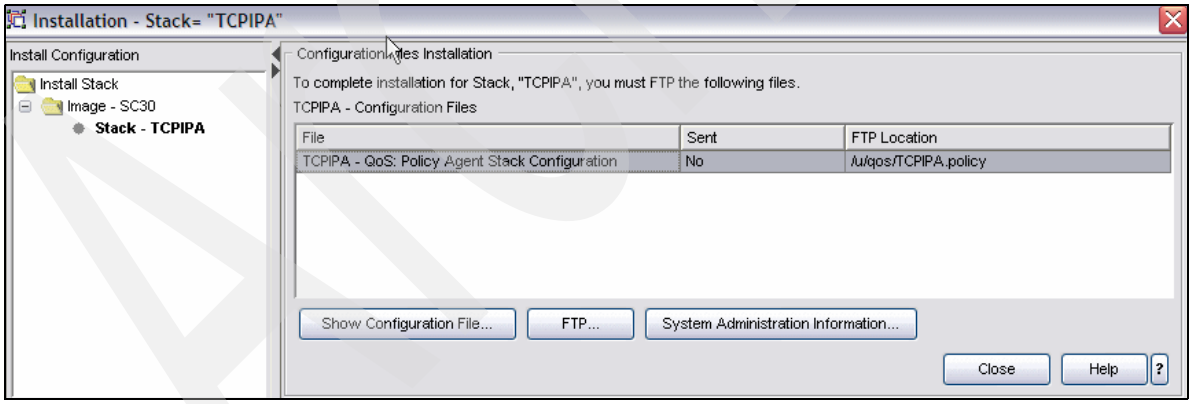


Figure 7-20 Installation - Stack - TCPIPA panel: installing the QoS configuration files

4. In the FTP Configuration File panel, enter the host name, user ID, and its password. In addition, verify that the port number is 21 and enter the file name and location for the QoS configuration file, as shown in Figure 7-21. The name and location should be equal to the name and location that were defined in the QoSConfig statement at the TCP/IP configuration file for Policy Agent. Click **Send**.

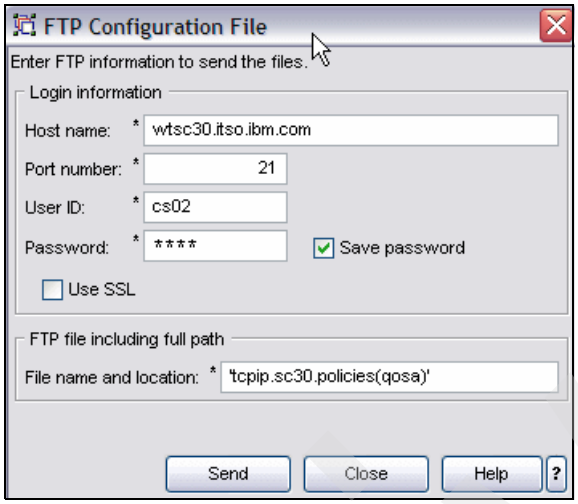


Figure 7-21 FTP Configuration File panel: installing the QoS configuration file

5. From the console, refresh the Policy Agent policies, by issuing the F PAGENT,REFRESH command. You should see the following messages on the log:

```
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : QOS
```

7.4.3 Problem determination

You can see the effect of defined QoS connectivity rules in the following ways:

- Check the log file of the Policy Agent to see if you got any error messages. (In our configuration, the log file is /tmp/pagent.sc30.log.)
- Use the Network SLAPM2 Subagent to display service policy and mapped application information, as well as to manage and display Network SLAPM2 performance monitoring.
- Use the SLA Subagent to display service policy and mapped application information, as well as to manage and display SLA performance monitoring.
- Use the z/OS UNIX **pasearch**, z/OS UNIX **netstat**, and TSO NETSTAT commands as follows:
  - The NETSTAT SLAP (**netstat -j**) command shows performance metrics for active QoS policy rules.
  - The NETSTAT ALL or **netstat -A** command has additional information for each active connection that shows the QoS policy rule name if the connection maps to a QoS policy.
  - Issue **pasearch -q** to see all QoS policies that are active in Policy Agent.

Available management tools

There are also tools available that can help you manage your network QoS configuration and parameters including the z/OS Communications Server SNMP SLA Subagent and network device MIB variables and tools.

z/OS Communications Server SNMP SLA Subagent

The z/OS CS SLA Subagent allows network administrators to retrieve data and determine if the current set of SLA policy definitions is performing as needed or if adjustments need to be made. The SLA Subagent supports the Service Level Agreement Performance Monitor (SLAPM) MIB. Refer to RFC 2758 for more information about the SLAPM MIB.

Network MIB variables and tools

If you are using Cisco networking gear, the following MIBs may be helpful.

Cisco QoS MIB

The Cisco Class-Based QoS MIB provides you with the same statistics that the **`show policy interface`** command provides.

To measure packet loss through the network with regard to different classes of service, use Class-Based Queuing over Security Management Information Base (CBQoSMIB (available in 12.1(5) T), SAA/IPM, or QPM.

See also:

- CISCO-CLASS-BASED-QOS-MIB.my
- CISCO-CLASS-BASED-QOS-MIB-CAPABILITY.my

Cisco QoS Device Manager

Cisco QoS Device Manager (QDM) is a Web-based network management application that provides an easy-to-use graphical user interface for configuring and monitoring advanced IP-based QoS functionality in Cisco Systems routers.

QDM is intended for users who are configuring QoS functionality in their network for the first time. It is an easy-to-use management application to help you configure and monitor QoS features in the most critical router devices in your network. Using QDM, you can quickly and easily configure QoS functionality and immediately observe the effect that this QoS configuration has on the pattern of network traffic through the network.

Cisco QoS Policy Manager

QoS Policy Manager (QPM) is a QoS policy system that makes it easy to define traffic policies and automate multiple service levels across any network topology. The product enables network-wide, content-based Differentiated Services; centralized policy control for voice/video/data networks; automated QoS configuration and deployment; and campus-to-WAN policy control.

By automating the process of translating application performance requirements into QoS policy, QPM helps ensure reliable performance for Internet business applications and voice traffic that contends with noncritical traffic. Using QPM, a network administrator can quickly construct rules-based QoS policies that identify and partition application traffic into multiple levels of service.

8

IP filtering

IP filtering provides a means of *permitting* or *denying* IP messages (packets) into and out of the z/OS Communications Server environment at a very early stage in message handling (and therefore, very efficiently). By “very early”, we mean at a point when the packet is just about to enter the protocol stack and before any applications have been executed or much “work” performed.

Note: One thing that can be confusing is that z/OS Communications Server IP filtering support has been packaged together with IP Security (IPSec) and is referred to as integrated IP Security. That is because there is a close affinity between IPSec and IP filtering in the z/OS Communications Server; although you can implement IP filtering without IPSec, you cannot implement IPSec without IP filtering.

In order to configure IP filtering, you have to indicate that you are configuring IPSec in the IBM Configuration Assistant for z/OS Communications Server.

The following topics are discussed in this chapter.

Section	Topic
8.1, “Defining IP filtering” on page 320	Basic concepts of IP filtering
8.2, “Why IP filtering is important” on page 322	Key characteristics of IP filtering and why it might be important in your environment
8.3, “How IP filtering is implemented” on page 323	Selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

8.1 Defining IP filtering

IP filtering enables a z/OS system to classify any IP packet that comes across a network interface and take specific action according to a predefined set of rules. An administrator can configure IP filtering to deny or allow any given network packet into or out of a z/OS system with an IP filtering policy. IP filtering provides:

- Packet filtering and logging
- Filtering rules that determine whether IPSec encryption and authentication are required

The **ipsec** command is used to manage and monitor the IP filtering and VPN policies.

Note: z/OS Communications Server provides IPv6 support for integrated IP Security: IPv6 support for IP filtering, IP security/Virtual Private Network (IPSec/VPN), and Internet Key Exchange (IKE) dynamic key management, no longer requiring the Integrated Security Services Firewall Technologies. IP filtering, IPSec, and Application Transparent Transport Layer Security (AT-TLS) are all under Policy Agent control.

This support provides easier configuration, greater scalability, improved performance, and enhanced serviceability over the Firewall Technologies. Therefore, integrated IP security is the recommended way to implement packet filtering.

Filter rules can be defined to match inbound and outbound packets based on:

- Packet information
- Network attributes
- Time of day

Possible actions that can be taken include:

- Permit (with or without manual or dynamic IPSec)
- Deny
- Log (in combination with Permit or Deny)

Note: When an IP packet is blocked, the source of the packet is not informed.

8.1.1 Basic concepts

When a packet arrives over a network interface into the z/OS environment, the IP filtering function running under the TCP/IP stack searches the Security Policy Database (SPD), matching the TCP/IP header against the specified filters. Filters are rules defined to either deny or permit packets. IP filtering matches a filter rule to data traffic based on any combination of IP source or destination address (or masked address), protocol, source or destination port address, direction of flow, or time.

In order to create the IP filtering policy, we have to know the resources available in the network, the resources available in a z/OS image, and how they relate to others hosts. Figure 8-1 on page 321 illustrates the basic concept of IP filtering.

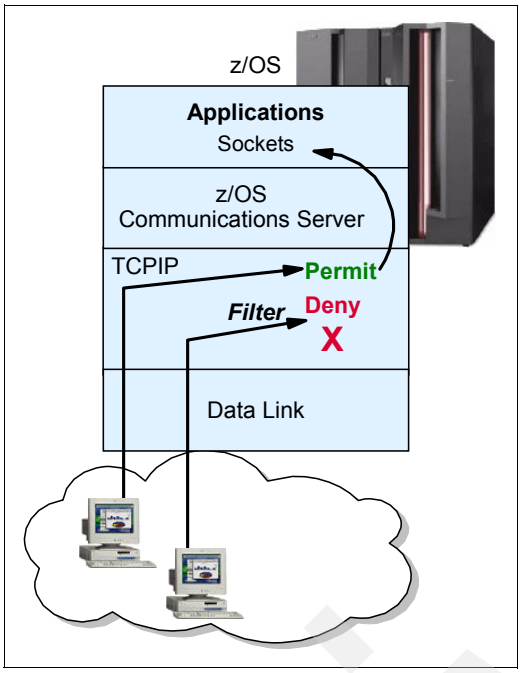


Figure 8-1 IP filtering at the z/OS data endpoint

The resources available in the z/OS image are:

- TCP/IP address space and stack (can be more than one stack)
- The network interfaces and their respective IP addresses
- The servers or clients, which are the address spaces running programs that will either access or be accessed by others hosts (such as TN3270 server, File Transfer Protocol (FTP) server and client, and IBM DB2®) and what interfaces they will be using
- The direction of the information flow and if routing might be needed
- Authentication and encryption requirements

The network resources that should be mapped outside the z/OS image are:

- Clients and servers that need to connect to the z/OS image, their IP addresses, and the services required
- Networks and subnets

The relationship between the TCP/IP components in a z/OS image and the network resources is translated in the IP filtering implementation. We can call this relationship an IP filtering *policy*. This policy contains all the rules that permit or deny the access to a z/OS image.

Note: The IP filtering function available on the z/OS Communications Server should generally only be used to control and protect the resources owned by the z/OS image. In other words, the best use of a z/OS image is not to have it function as a firewall router (also called a secure gateway). There are specialized network devices that are more suitable and cost-effective as routing firewalls.

Security Policy Database

The Security Policy Database provides two types of filter policies: the *default IP filter policy* and the *IP security filter policy*, as explained here.

- The default IP filter policy is intended to allow limited access while the IP security filter policy is being loaded. It can be reverted to in an attack situation with an operator command. The default IP filter policy is defined in the TCP/IP profile, and the default is to deny all traffic. It provides a basic filtering function only (permit rules only and no VPN support).
- The IP security filter policy is intended to be the primary source of filter rules, and the default is to deny all traffic. It is defined in a Policy Agent IPSec configuration file and can be generated by the IBM Configuration Assistant for z/OS Communication Server.

The **ipsec** command is used to switch between the default and IP security filter policies.

The IPSECURITY option on the IPCONFIG statement

The IPSECURITY option enables use of the integrated IP security functions. Figure 8-2 shows a functional view of IP filter policy on z/OS.

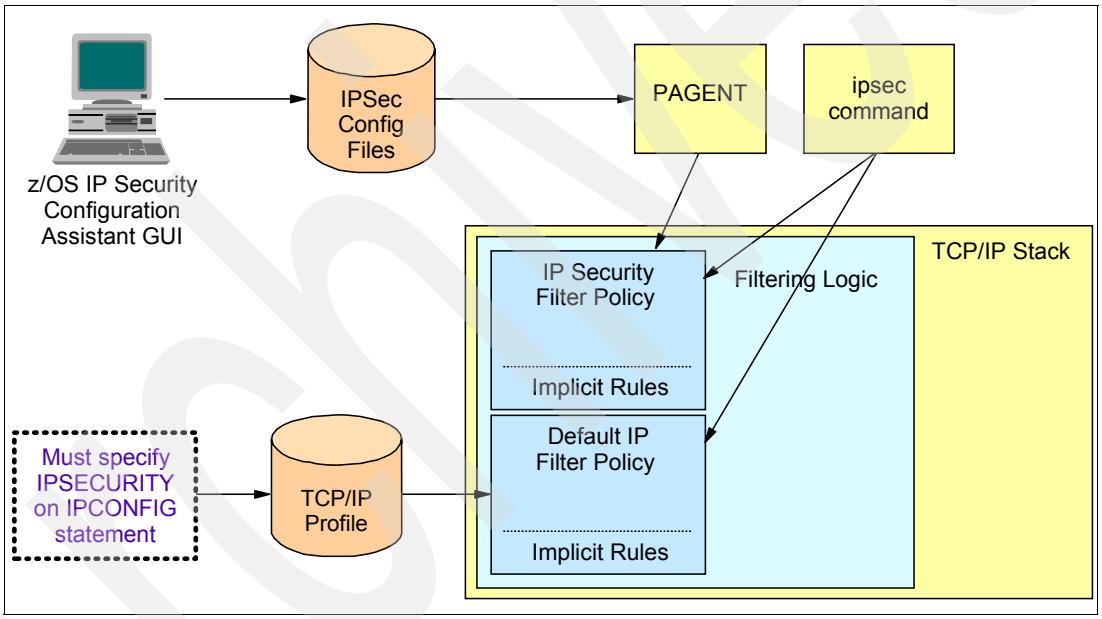


Figure 8-2 IP filter policy

8.1.2 For additional information

For additional information about these topics, consult the following books:

- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782

8.2 Why IP filtering is important

Depending on your security policies, IP filtering capabilities can provide either the primary means of protecting your z/OS environment from network-based attacks, or a powerful additional line of defense (when used in conjunction with layers of external firewalls and access control lists).

IP filtering is required in order to identify the IPSec behavior to apply to all traffic.

8.3 How IP filtering is implemented

IP filtering is implemented through the z/OS Communications Server PAGENT function. PAGENT is discussed in Chapter 5, “Policy Agent” on page 221.

8.3.1 z/OS IP filtering implementation

There are two parts to implementing the IP filtering policy:

- The *default policy* is specified using the IPSEC statement in the TCP/IP profile data set.
- The *filter policy* is specified using the PAGENT policy configuration files. Those are flat files that can be created in a z/OS UNIX file or in a sequential data set under z/OS.

The TCP/IP profile has to be configured to activate IP filtering. That is done with the IPSECURITY option defined in the IPCONFIG statement[1]. If IPSECURITY is not specified, then the IP filtering function will not be available even if it is configured either in the PAGENT filter policy configuration file or in the IPSEC statement in the TCP/IP profile. Figure 8-3 shows the structure of an IP filtering implementation.

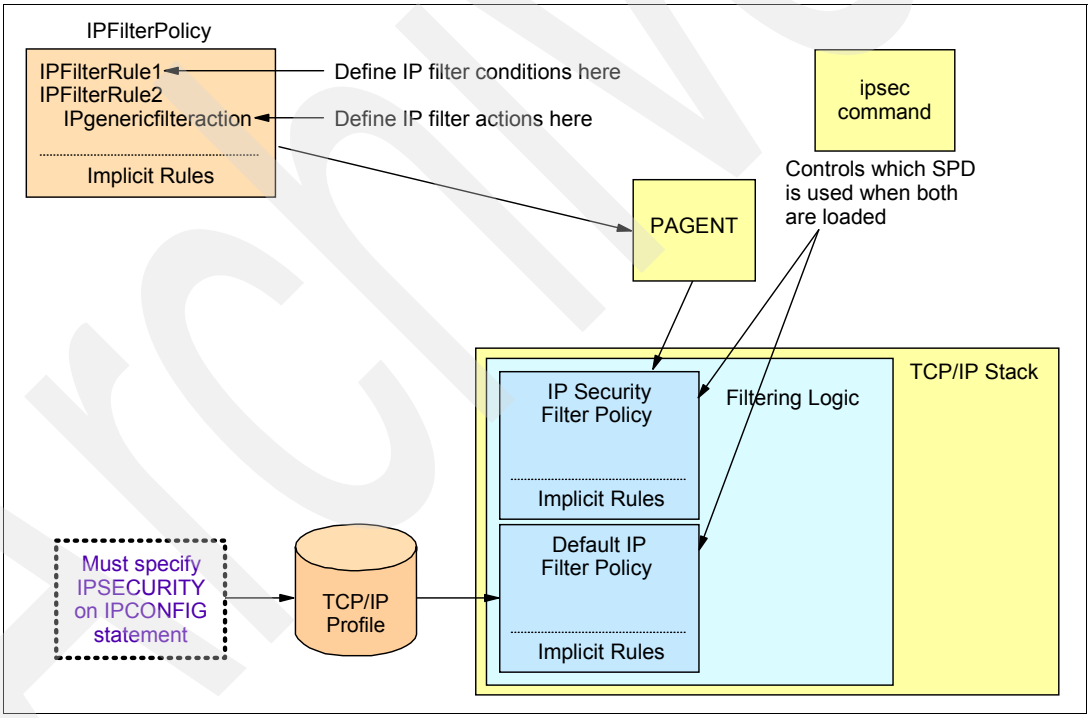


Figure 8-3 IP security filtering structure

As illustrated in Figure 8-3:

- PAGENT loads the filter policies into the TCP/IP stack at startup or when you make changes to it and refresh the policy by using the **`modify`** console command.
- The default policy is always loaded by the stack when the configuration profile is first processed. The initial profile does not have any IP filter rules, so only the implicit rules will

[1] Define IPSECURITY in the IPCONFIG6 statement to enable IP filtering for IPv6.

be loaded. Any rule defined for the log options can be changed by the **vary tcpip obeyfile** console command.

- The **ipsec** command is used to manage and monitor the IP security filtering.

Important: The implicit rules are always created using either the default or the filter policy. Just by using the IPSECURITY option in the IPCONFIG statement, the implicit rules are created and deny all the inbound and outbound TCP/IP traffic in that z/OS image. This means that if you code IPSECURITY without either some IPSEC statements or filter policies, you will have a completely inaccessible stack.

Therefore, consider defining a default policy that has an IPSECRULE to allow one administrative IP address to connect to the TCP/IP stack. In this way, if PAGENT fails to start, you will still have a way to access the stack using TN3270.

The major differences between the default policy and the filter policy are:

- In the default policy, there are only permit rules. The implicit rules implement the deny all functions. In the pagent policy, you have the option to create deny rules.
- The default policy only permits local packets; all routed traffic is denied. If you want to apply filter policies to messages that are being routed between a z/OS image and other z/OS images, then the filter policy must be used.
- There is no option to group similar resources in the default policy. That capability is only available in the filter policy.

The default policy is always used in the absence of a filter policy. If a filter policy is defined, both are loaded into the TCP/IP stack and the filter policy is used unless you specify with the **ipsec** command that the default policy should be used. Using the **ipsec** command, you can switch between the default and the filter policy whenever necessary.

The IPSECURITY option is activated only at the TCP/IP startup. If you want to remove the IP filtering function, you must restart the stack without it.

Example 8-1 shows our IP filtering definitions in the TCP/IP profile for our z/OS test system image A23.

Example 8-1 IPSec configuration statements on profile for z/OS image A23

```
IPCONFIG DATAGRAMFWD
   IPSECURITY       1

 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 40 SECCLASS 1 2
 DEVICE OSA20A0  MPCIPA
 LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 40 SECCLASS 1
 DEVICE OSA20C0  MPCIPA
 LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 41 SECCLASS 1
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 41 SECCLASS 1

IPSEC          3
   LOGENABLE   4
   LOGIMPLICIT 5
ENDIPSEC
```

1 The IPSECURITY option is specified on the IPCONFIG statement, indicating that we want IP Security activated on the stack. This option automatically installs the default security policy

that contains only the implicit rules, meaning that all the IP traffic is blocked. If PAGENT is running and there is an IP filtering policy defined, it is installed and activated.

Important: If the IPCONFIG IPSECURITY statement is coded in the TCP/IP profile, the default IP filter policy is to effectively deny all network traffic, with the exception of some selected Internet Control Message Protocol (ICMP) messages that are necessary for the internal stack function.

2 The SECCLASS option on the LINK statements enables you to either uniquely identify an interface, or to group interfaces with similar security requirements, based on site policy. Then you can configure a single IP filter rule that matches all of the IP traffic from interfaces that share a common security class without explicitly identifying any attributes of the IP packets. In our scenario, we have only one security class for the network outside the z/OS image; the security class 1.

Each non-virtual interface on a z/OS system is assigned a security class. The security class of an interface is determined by the SECCLASS parameter that is coded on either the LINK statement or the DYNAMICXCF parameter of the IPCONFIG statement in the TCP/IP profile. The value of SECCLASS is a number in the range 1 - 255. If SECCLASS is not specified for an interface, the interface is assigned the default security class of 255.

Each IP packet entering or leaving the system inherits the security class of the interface that it traverses:

- For inbound traffic, this is the interface on which the packet arrived.
- For outbound traffic, this is the interface over which the packet is sent.

Security classes can only be assigned to physical interfaces, not VIPA devices. Networks connected to the same network interface (for example, Alpha network and Beta network in Figure 8-4) cannot be distinguished into different security classes.

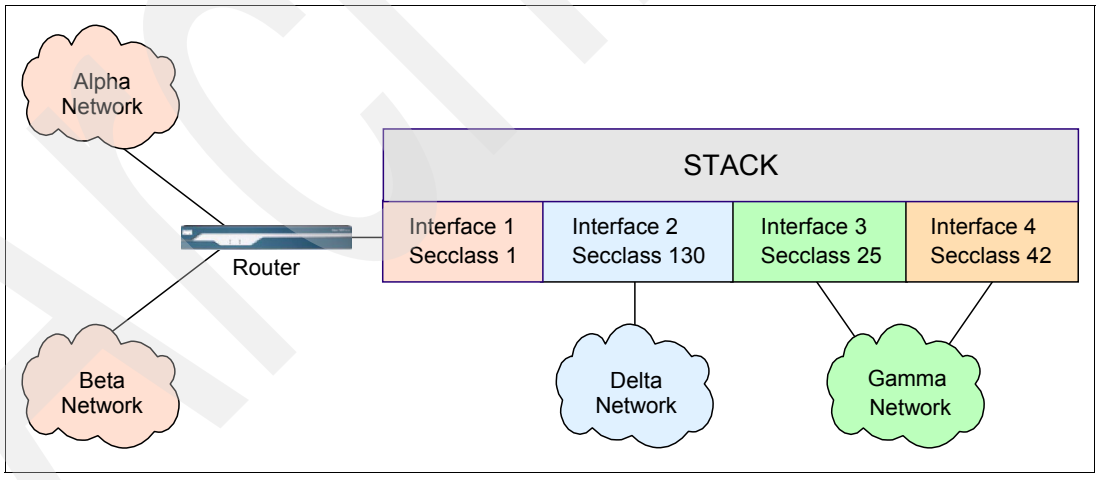


Figure 8-4 Interface security class example

Note: Although it is possible to configure different security classes for different interfaces into the *same* network (for example, Gamma network in Figure 8-4), all interfaces into the same network should be configured with the same security class to avoid unnecessarily complicating security policies.

Security classes can be used in conjunction with IP address information to create filter rules to block packets having spoofed source IP addresses. For example, if a packet enters the stack from the Delta network (in Figure 8-4 on page 325), but its source IP address is not from the address space of the Delta network, then the packet is probably spoofed and should be denied.

3 The IPSEC statement is used to define the default policy. The default policy is activated and applied if the filter policy is not defined. Additional rules have to be defined in order to gain access to services running on this TCP/IP stack. In our scenario, our default policy contains only the implicit rules, and all of our real policy rules are defined in the filter policy configuration. Filter rules can be defined in the IPSEC statement by using the IPSECRULE statement. In our example, we do not have any rules defined on the IPSEC statement.

4 The LOGENABLE option activates packet filter logging. All the log messages are sent to the syslogd by the Traffic Regulation Management Daemon (TRMD). You can disable it by using the LOGDISABLE option. In each of the filter rules, there is an option to generate (or not) a log record when the rule is applied to a packet.

5 The LOGIMPLICIT specifies that we want to log all packets that get blocked by the implicit rules in the default policy.

In our implementation, we have all the configuration files stored in a partitioned data set (PDS). This PDS is shared by all of the components: PAGENT, TRMD, SYSLOGD, and TCP/IP address spaces. Figure 8-5 shows the members flow to customize IP filtering from IPSec and PAGENT.

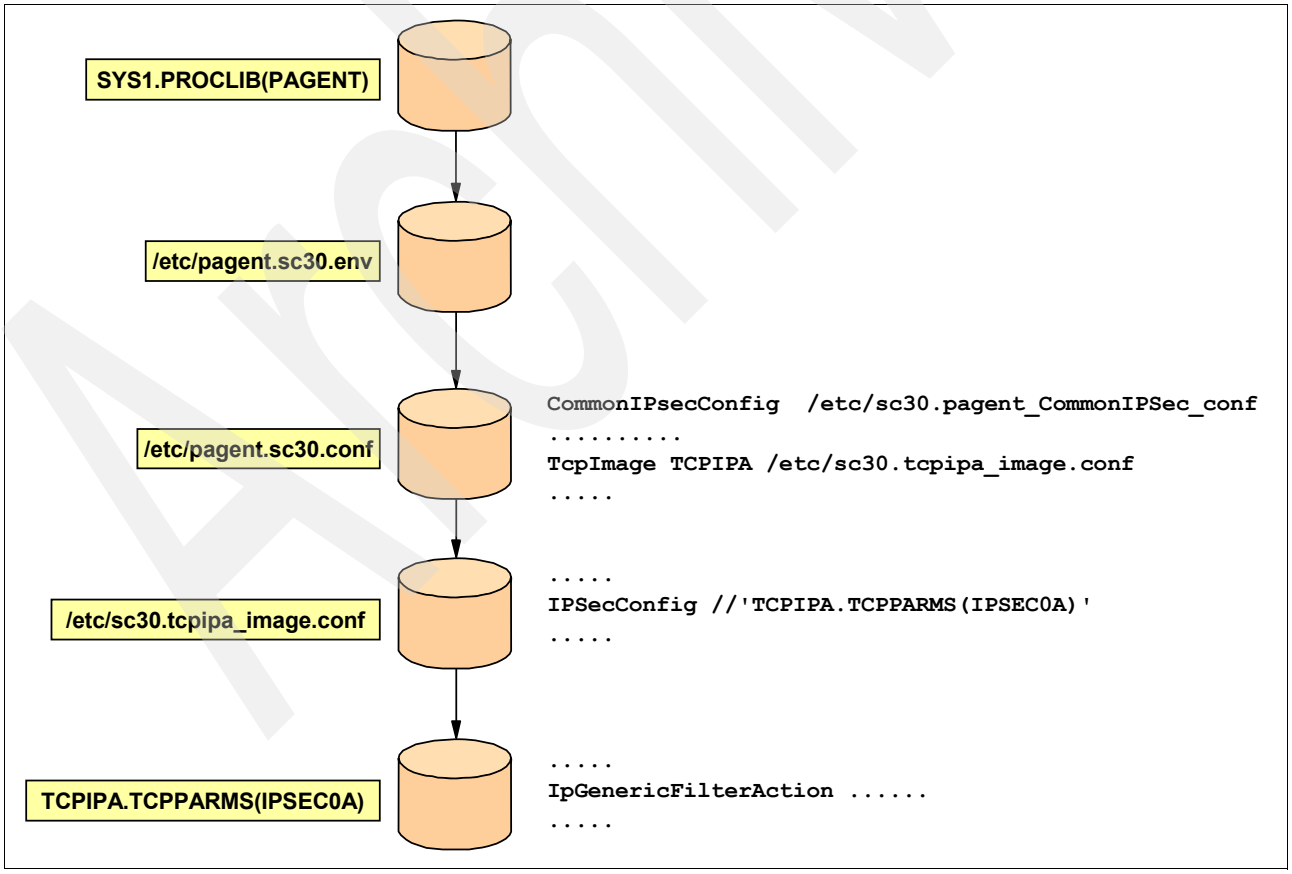


Figure 8-5 Members flow

The IpSecConfig statement in the pagent configuration file is:

```
IpsecConfig //'TCPIPA.TCPPARMS(IPSECOA)'
```

The IpsecConfig statement specifies the path of an IPSec policy file that contains common IPSec policy statements. If no path name is specified, then the common IPSec policy file specified on the CommonIpSecConfig statement is used.

You can manually create the IP security policy configuration files by coding all of the required statements in a z/OS UNIX file or MVS data set. There are a large number of powerful configuration options provided by IP security policy statements that permit advanced users to carefully fine-tune the IP security policy. However, IBM also provides a configuration graphical user interface (GUI) that you can use to generate the Policy Agent and IKE daemon configuration files. The IBM Configuration Assistant for z/OS Communication Server is a standalone application that runs under the Windows operating system and requires no network connectivity or setup.

The IBM Configuration Assistant for z/OS Communication Server GUI tool can be downloaded from:

http://www-306.ibm.com/software/network/commserver/zos/support/

For a complete explanation of the IP filtering statements and options, see:

- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

8.3.2 FTP and Telnet IP filtering scenario

In the following IP filtering implementation scenario (see Figure 8-6 on page 328), we use the IBM Configuration Assistant for z/OS Communication Server to define IP filter rules for LPAR A23 as follows:

- Permit Telnet traffic between remote workstation (10.12.4.224) and LPAR A23 (10.10.1.230)
- Permit FTP traffic between IP address 10.10.1.241 (LPAR A24) and IP address 10.10.1.230 (LPAR A23)
- Permit *basic services* traffic between all data endpoints and A23

Important: When the IPSECURITY option is activated, the implicit rules are in effect. Therefore, all inbound and outbound traffic traversing the TCP/IP stack is denied.

In our scenario, we allow only specific point-to-point connections; however, there is also some basic services traffic that must be permitted. These include:

- resolver and DNS (if implemented in the z/OS image)
  - Outbound: srcport=any, destport=53, proto=TCP or UDP
  - Inbound: srcport=53, destport=any, proto=TCP or UDP
- omproute (dynamic routing)
  - RIP: Inbound and outbound: srcport=520, destport=520, proto=UDP; for RIPv2 also need IGMP
  - OSPF: Inbound and outbound: IP protocol 89 (OSPF) and IGMP

We also recommend permitting ping traffic. This allows you to verify connectivity between data endpoints.

- Deny all other application traffic for LPAR A23

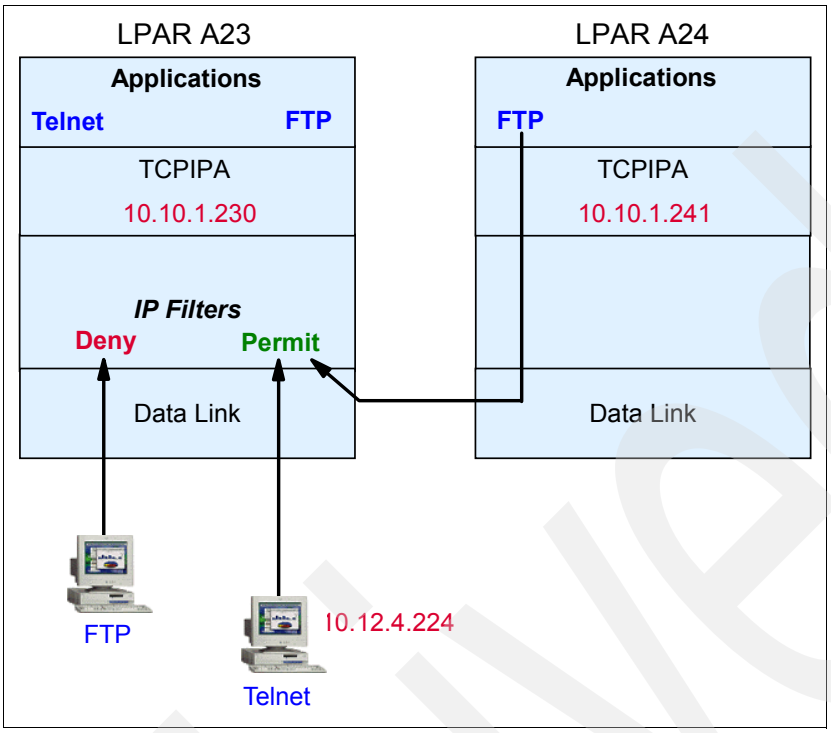


Figure 8-6 IP filtering scenario

Implementation steps

The steps are as follows:

1. Start the IBM Configuration Assistant for z/OS Communication Server.

   This presents you with the main configuration window (shown in Figure 8-7 on page 329), from where you can create and implement policies. Select **IPSec only** (recall that z/OS Communications Server IP filtering support is packaged with the IPSec support), then click **Add New z/OS Image**.

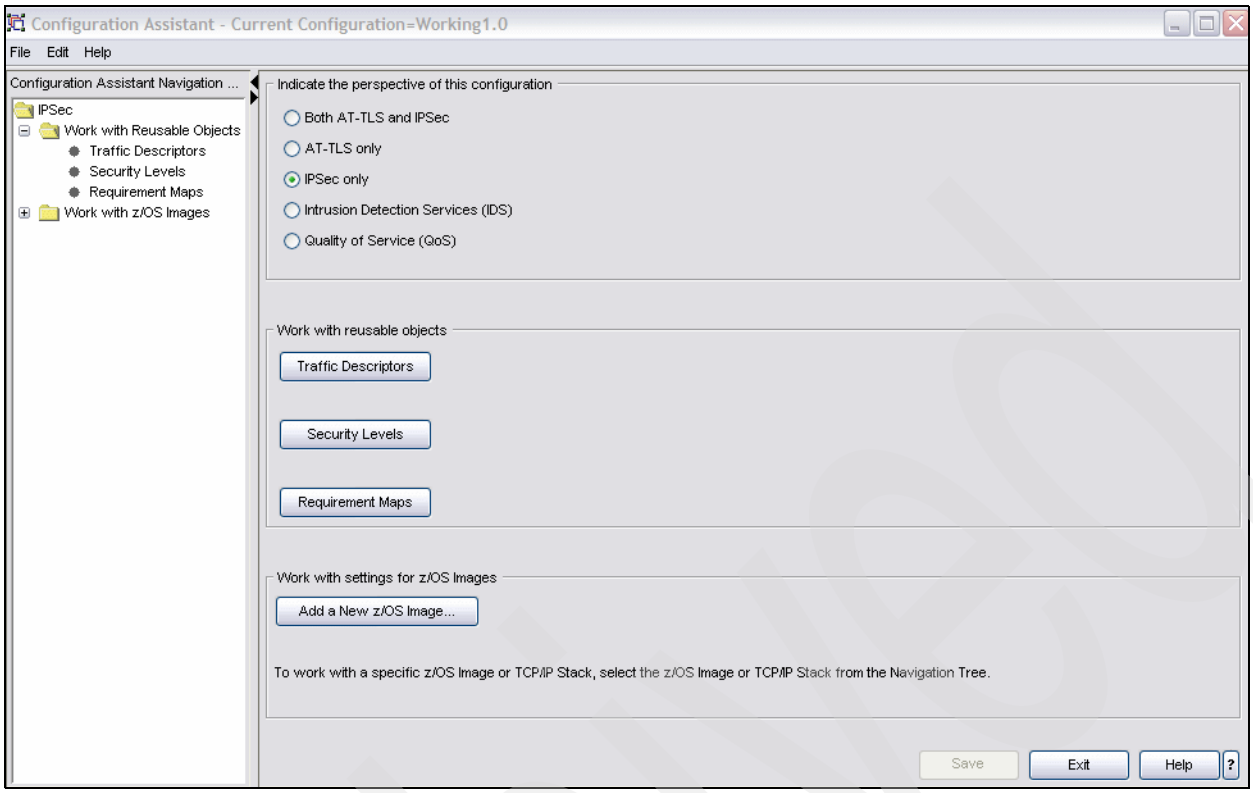


Figure 8-7 Configuration Assistant start panel

2. Figure 8-8 on page 330 shows that our LPAR name is A23 and we are not using IPSec VPNs with dynamic tunnels, because we are only applying IP filtering in this scenario. Click **Next**.

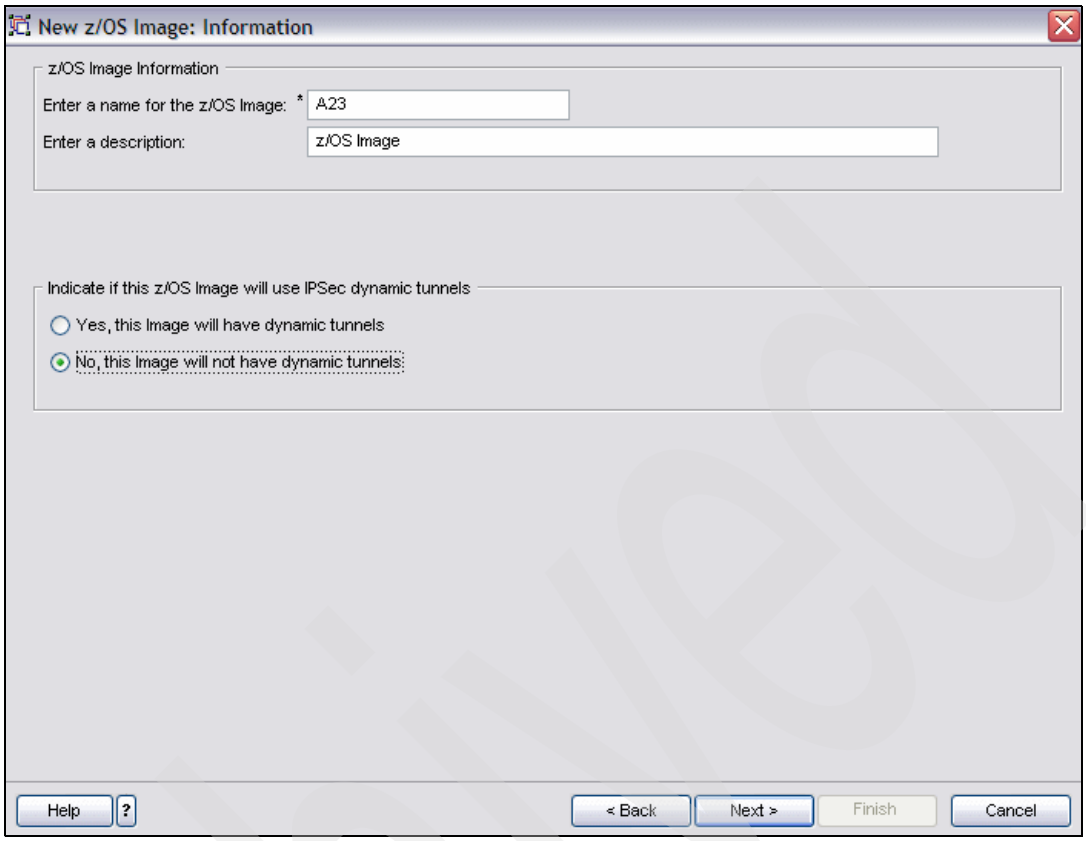


Figure 8-8 z/OS image information

3. The window shown in Figure 8-9 opens. Click **Finish**.

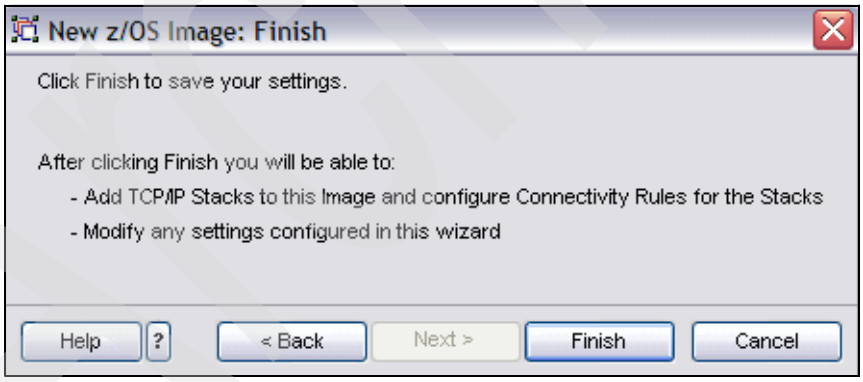


Figure 8-9 Save new image

4. You are asked to add a TCP/IP Stack, as shown in Figure 8-10. Click **Yes**.

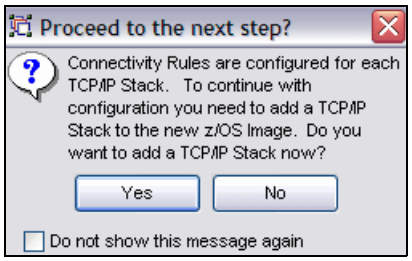


Figure 8-10 Proceed to configuring TCP/IP stack

5. The Welcome wizard window is displayed, as shown in Figure 8-11. Click **Next**.

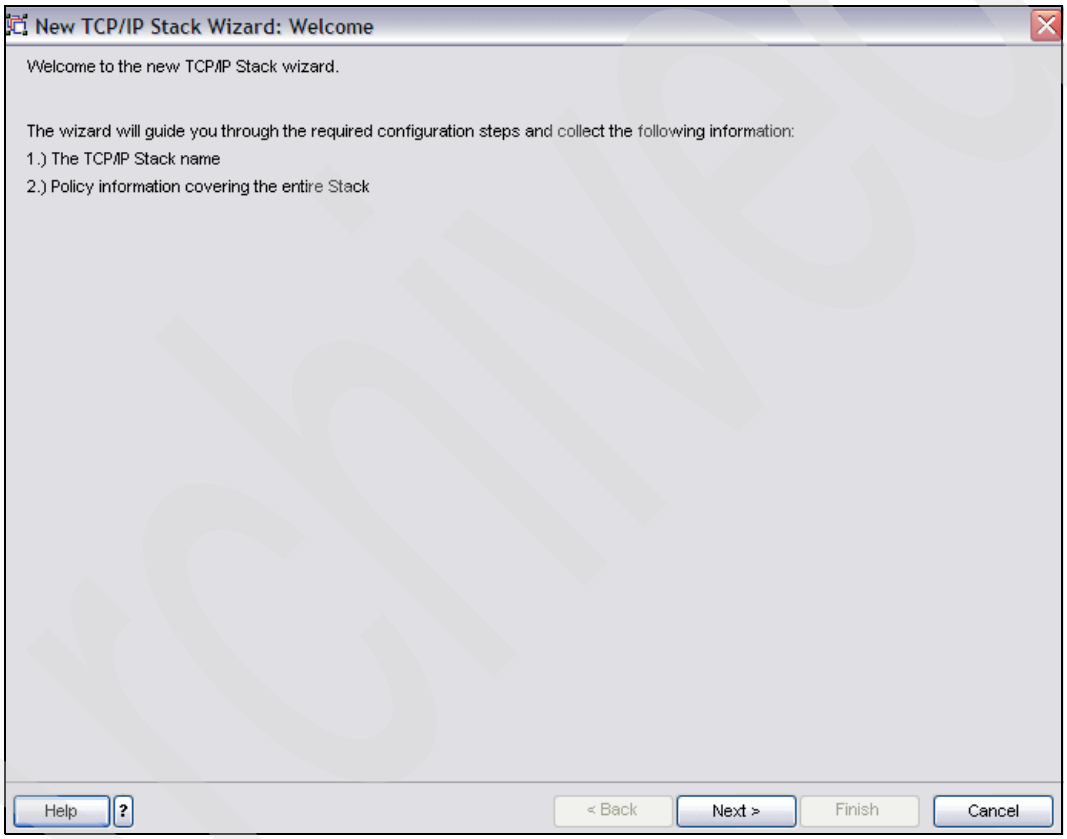


Figure 8-11 New Stack Wizard

6. The window shown in Figure 8-12 on page 332 opens.

   Our stack name is TCPIPA and, as with the z/OS image, we are not using IPSec VPNs with dynamic tunnels, because we are only applying IP filtering in this scenario.

   Click **Next**.

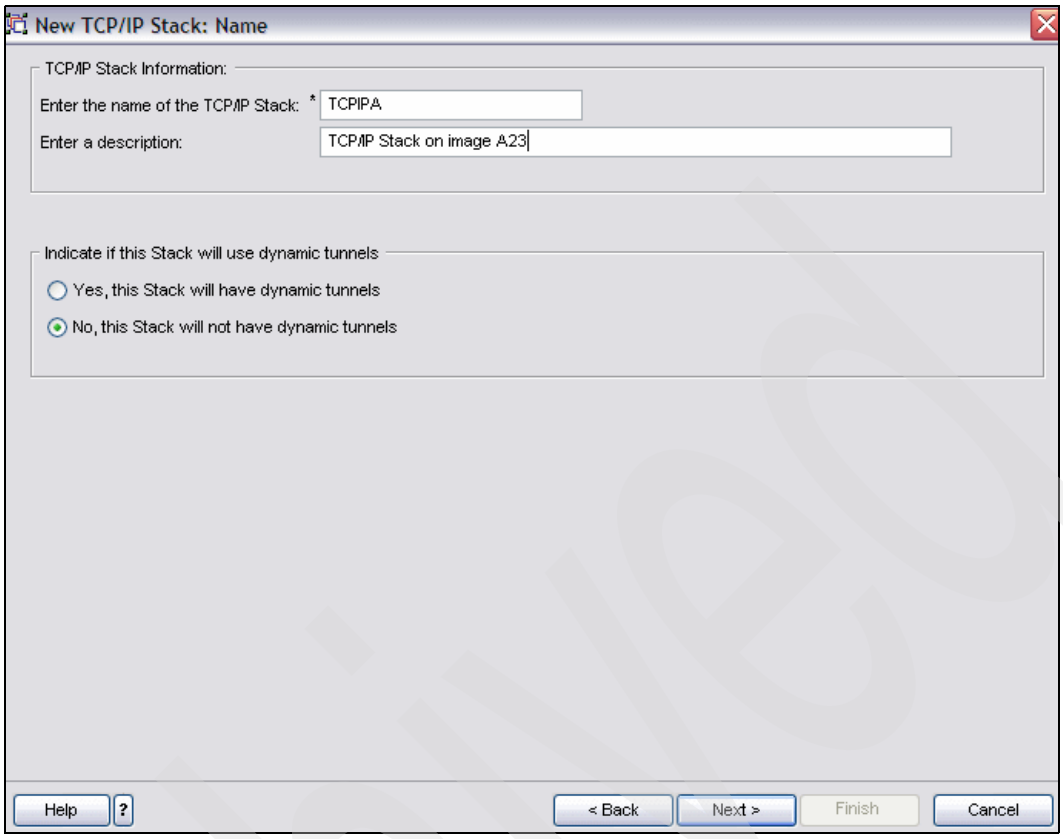


Figure 8-12 IP stack information

7. In Figure 8-13 on page 333, configure the stack-level settings. We chose to:
   - Enable Filter Logging Policy.
   - De-encapsulate and then filter IPSec payloads rather than filter IPSec headers.
   - Allow IP V6 link activation.

   Click **Next**.

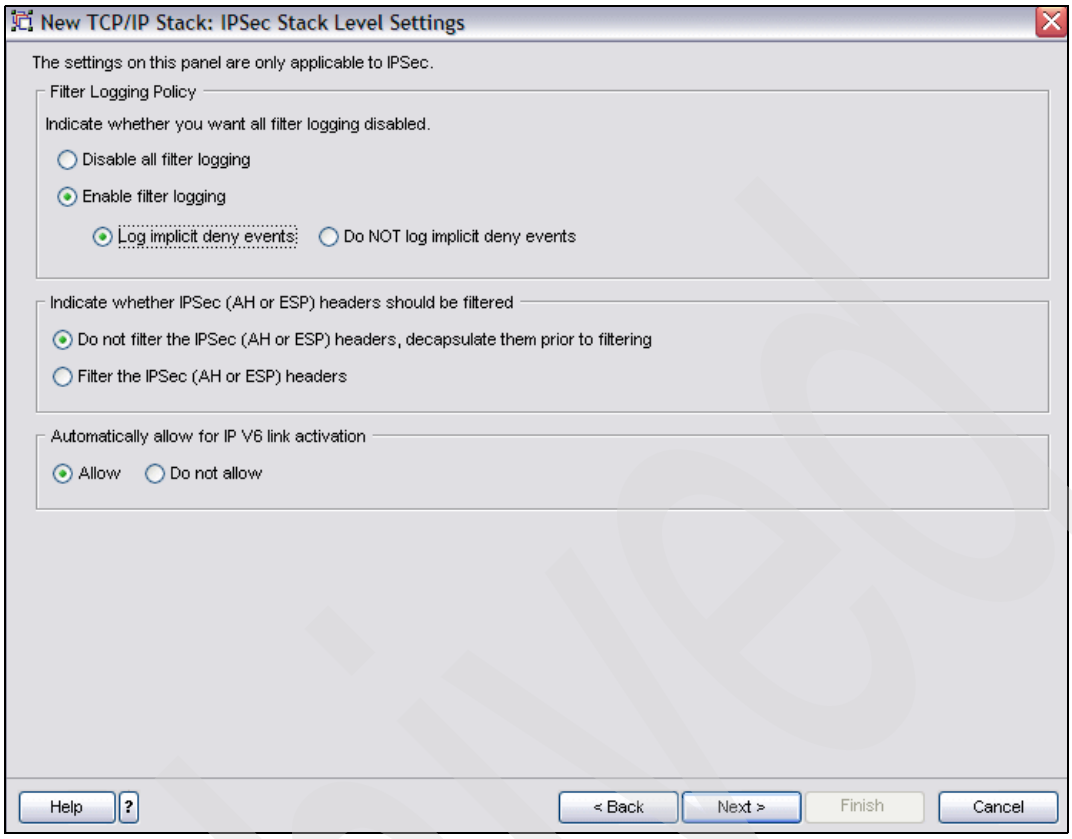


Figure 8-13 Filter Logging Policy

8. You are asked to save your Filter settings (Figure 8-14). Click **Finish**.

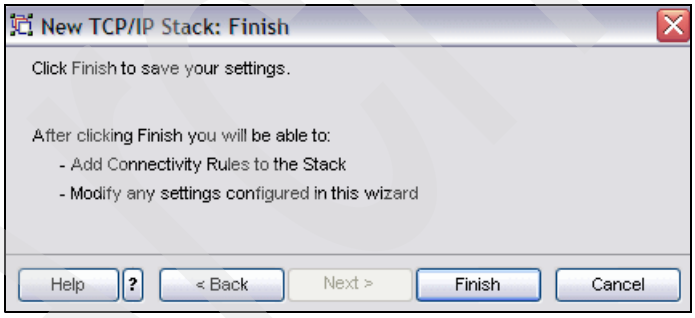


Figure 8-14 Save Filter settings

9. You are asked to configure the connectivity rules for the new stack (see Figure 8-15). Click **Yes**.

Figure 8-15 Proceed to configuring connectivity rules

10. Indicate your connectivity rule type in the Connectivity Rule: Welcome window shown in Figure 8-16.

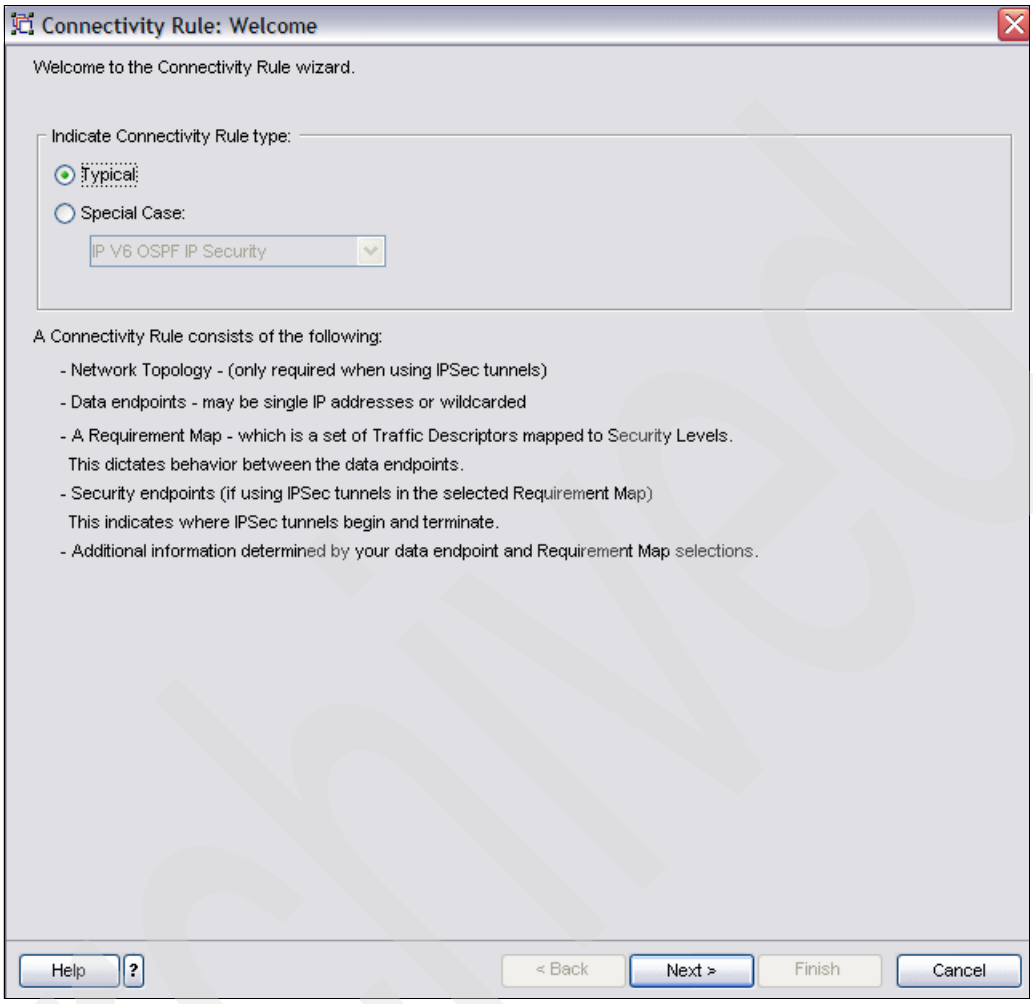


Figure 8-16 Connectivity Rule Welcome window

11. Select **No topology information is required** and click **Next** (see Figure 8-17).

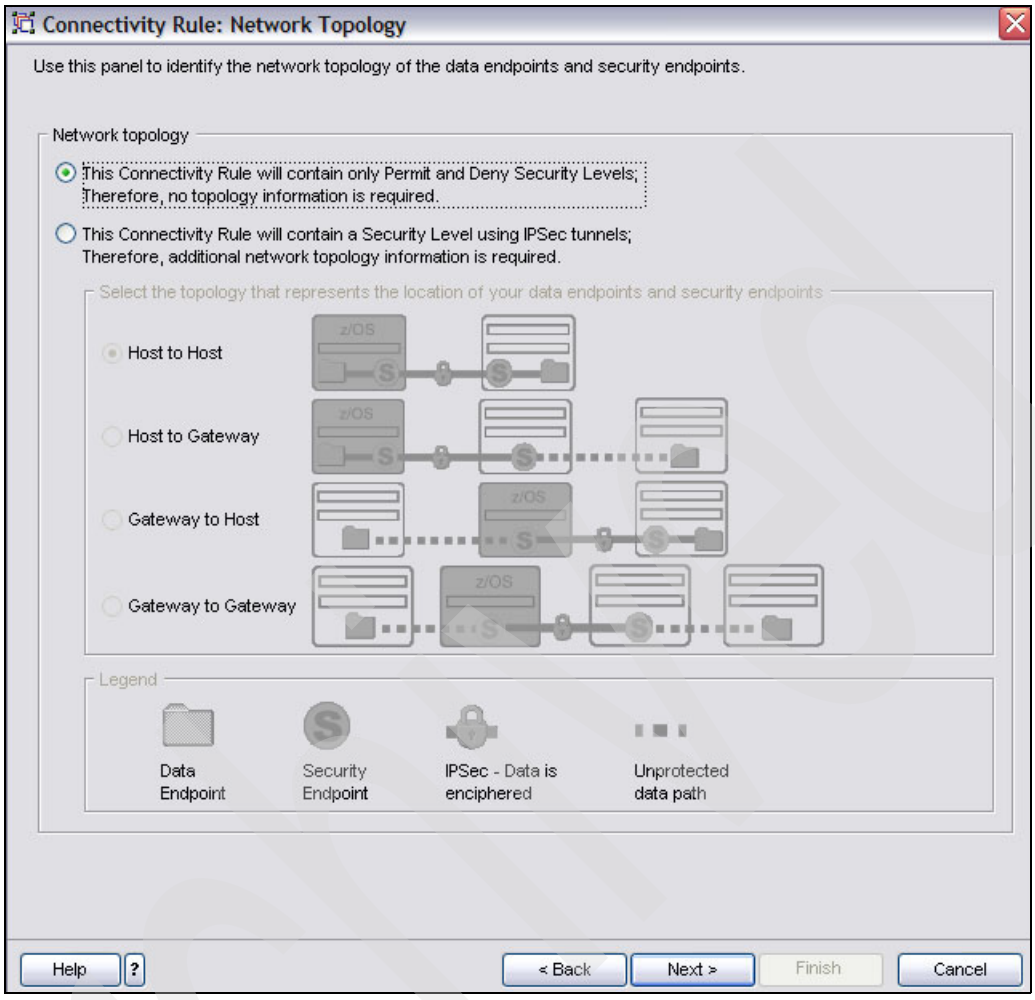


Figure 8-17 Network topology

12. In the panel shown in Figure 8-18, specify the IP addresses of the data endpoints. This can be a single IP address, an IP address range, an IP subnet, or all IPv4 or all IPv6 addresses.

    The name of the Connectivity Rule must be a string from 1 to 25 characters. The configuration assistant supplies a zero (0) in this field, which you can change to something more meaningful.

    Important: The *Source data endpoint* must be the IP address of the stack that you are protecting.

    Click **Next**.

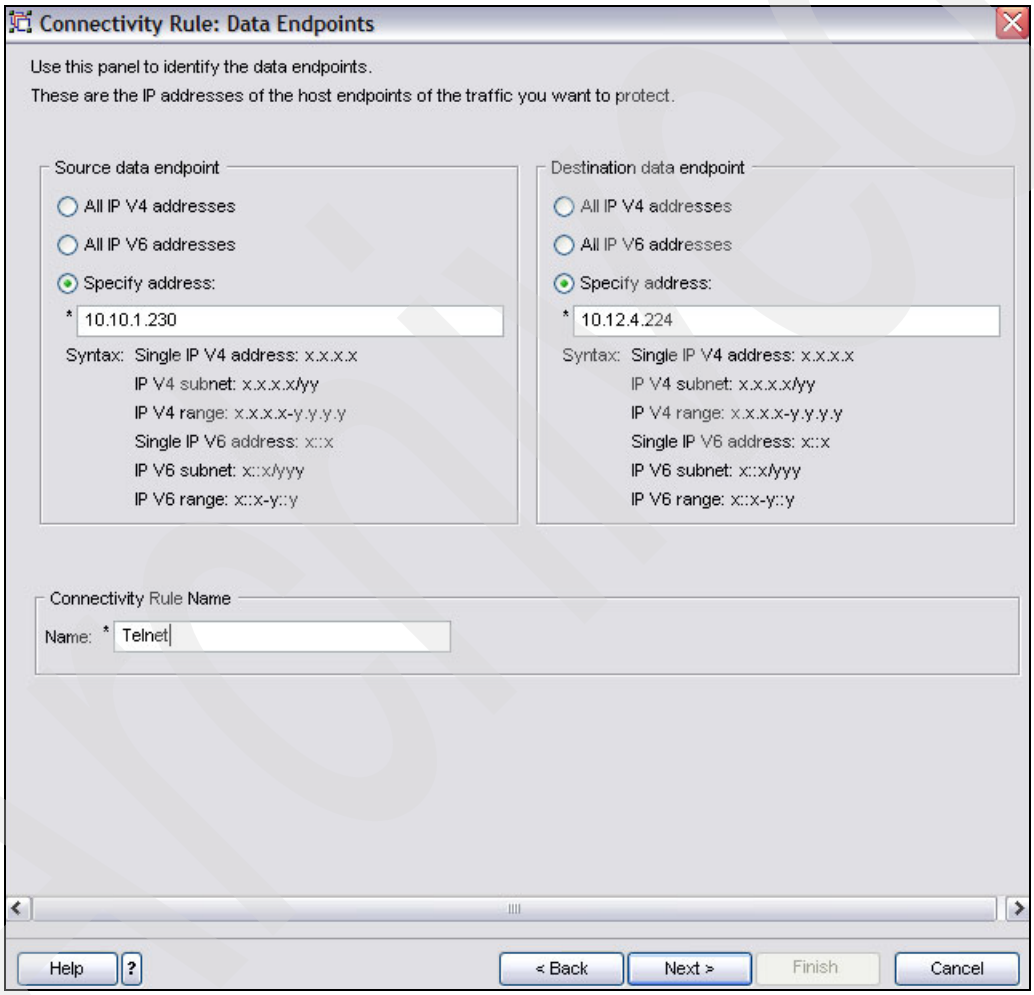


Figure 8-18 Connectivity rules

13. The Connectivity Rule: Select Requirement Map window in Figure 8-19 opens. There are IBM-provided maps; however, in our example, we want to configure a new rule, therefore we click **Add for Beginners**.

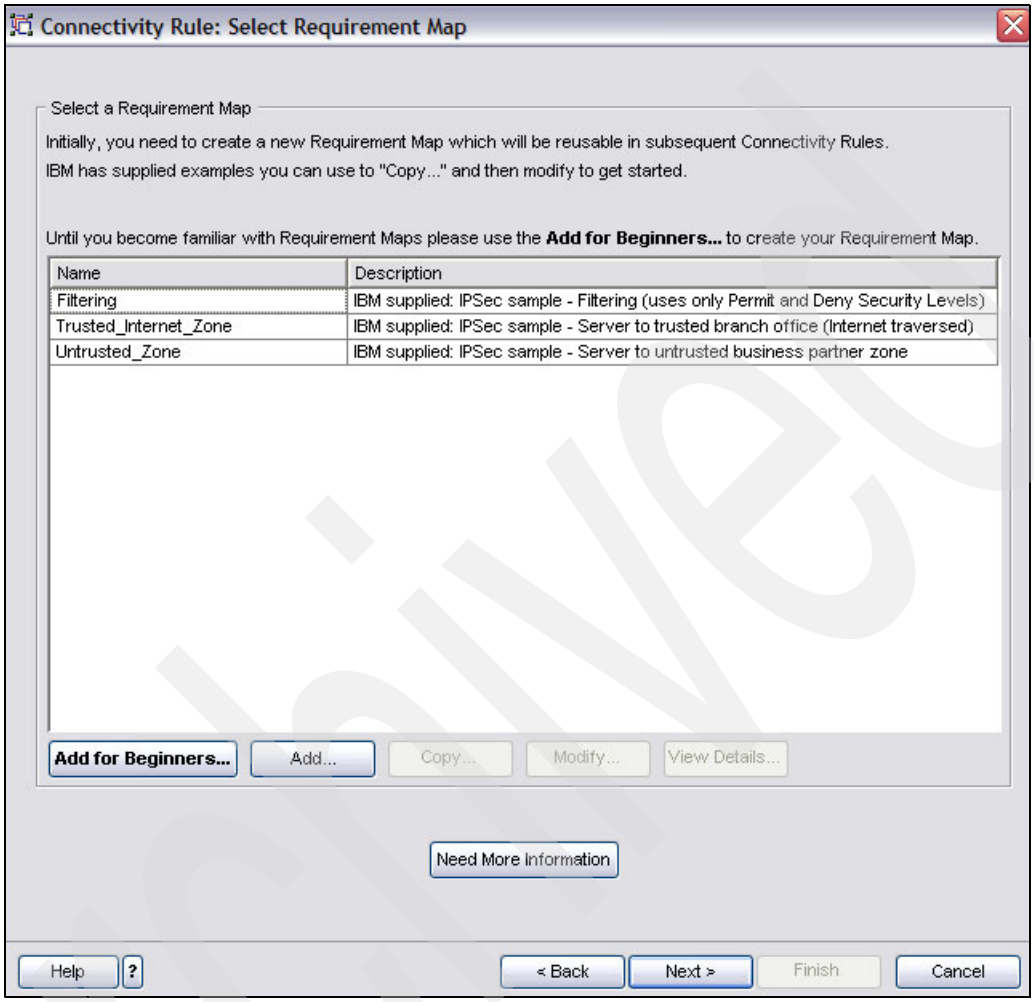


Figure 8-19 Map for data endpoints

14. This opens the Requirement Map Introduction window shown in Figure 8-20. Click **Next**.

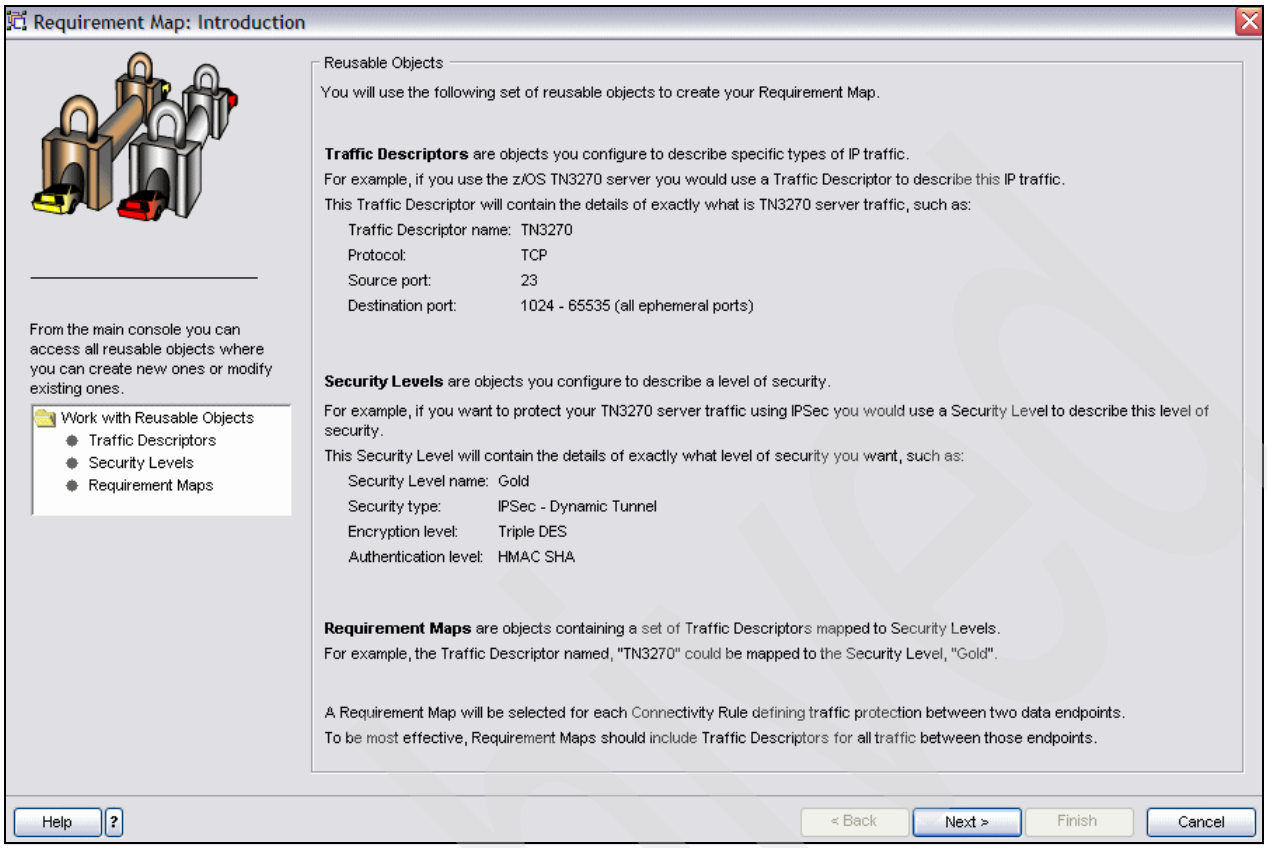


Figure 8-20 Requirement Map Introduction window

15. Figure 8-21 shows a listing of likely traffic types. Select **TN3270 Server traffic** to allow Telnet connectivity between remote workstation (10.12.4.224) and LPAR A23 (10.10.1.230). Click **Next**.

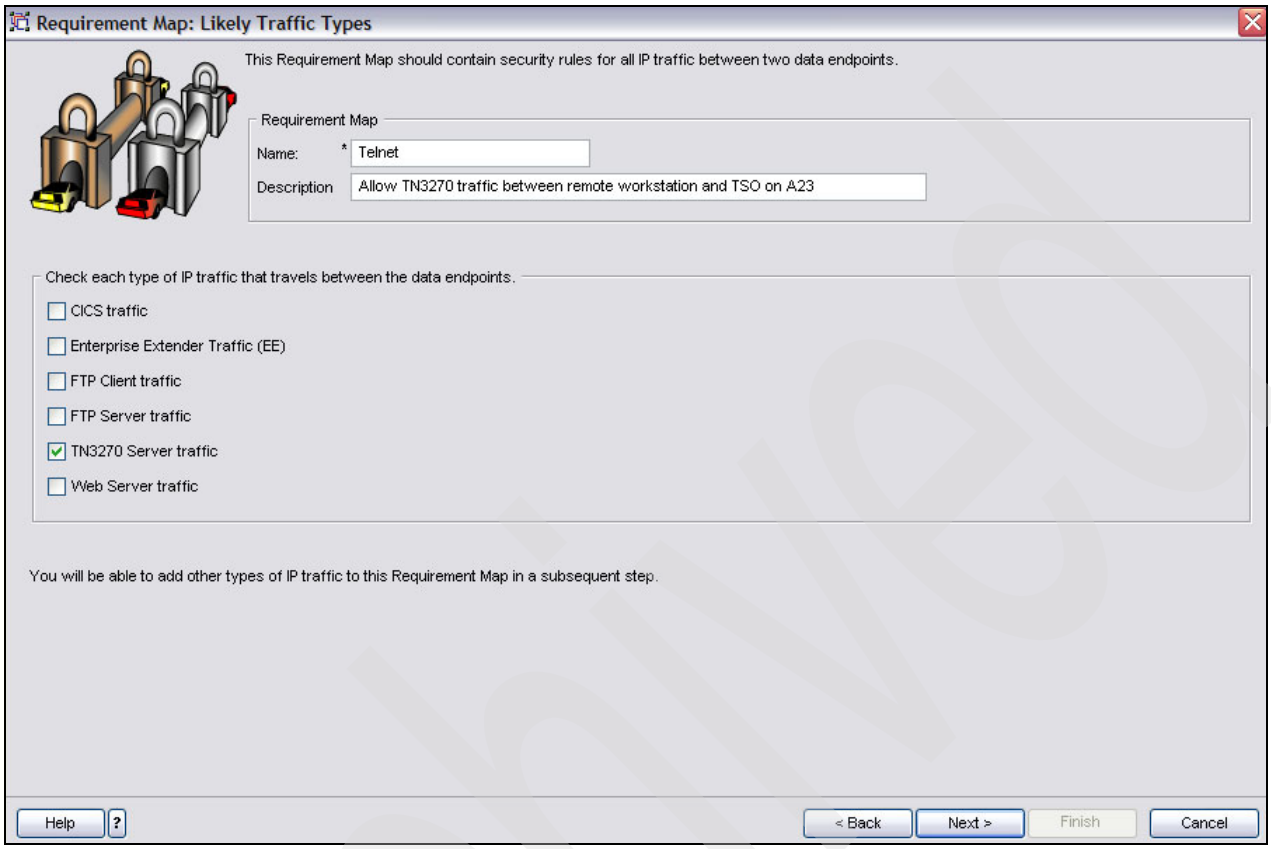


Figure 8-21 Traffic descriptors

16. The window shown in Figure 8-22 gives you the opportunity to add or remove traffic descriptors.

    Note that the IPSECURITY implicit rules (deny all traffic) are in effect. Therefore, the all_other_traffic descriptor is not needed in the requirement map. If required, you can remove it by selecting **All_other_traffic** and clicking **Remove**.

    Click **Next**.

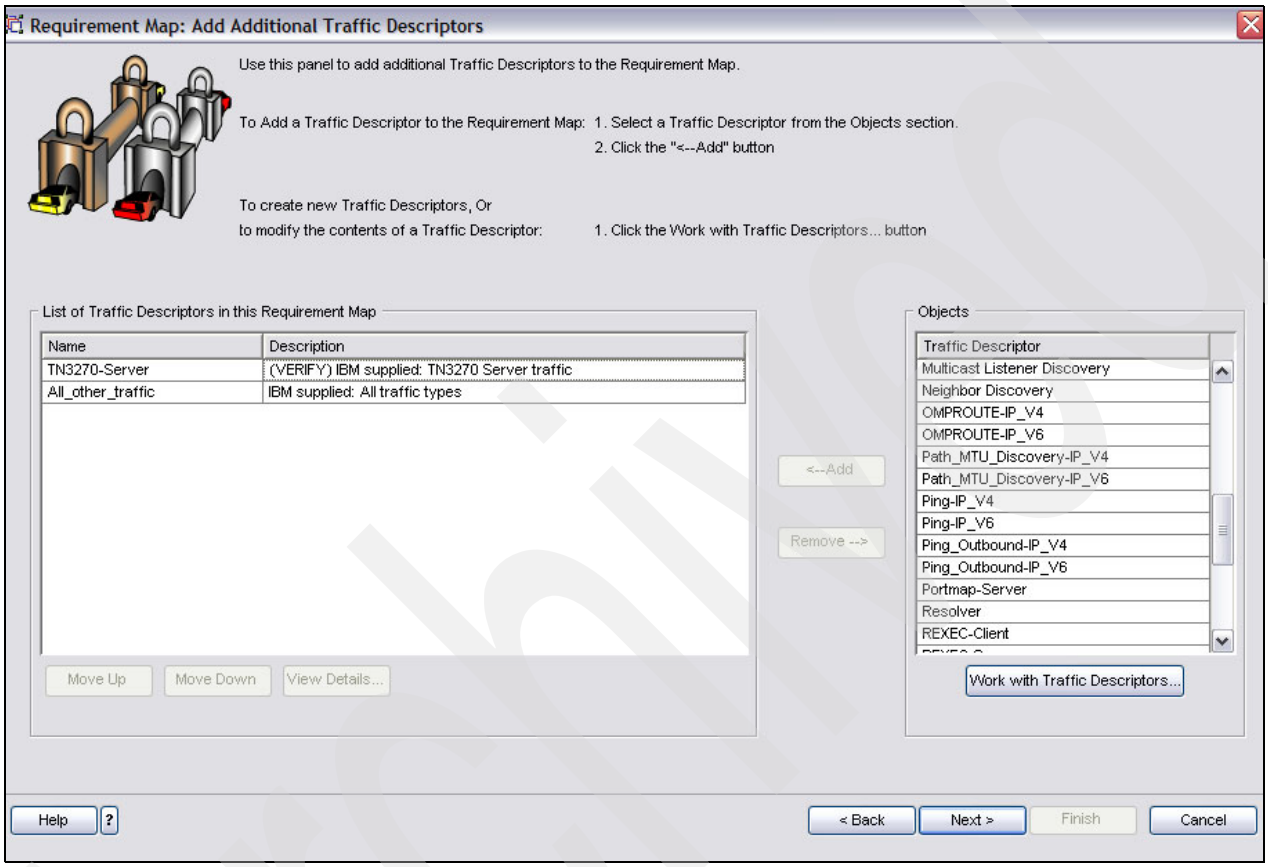


Figure 8-22 Add or remove traffic descriptors

17. Set the security levels. Each entry in the Requirement Map is a mapping between a Traffic Descriptor and either Deny, Permit, or IPSec Security Level (see Figure 8-23).

    Click the I**PSec - Security Level** tab for TN3270-Server and select **Permit**. Click **OK**.

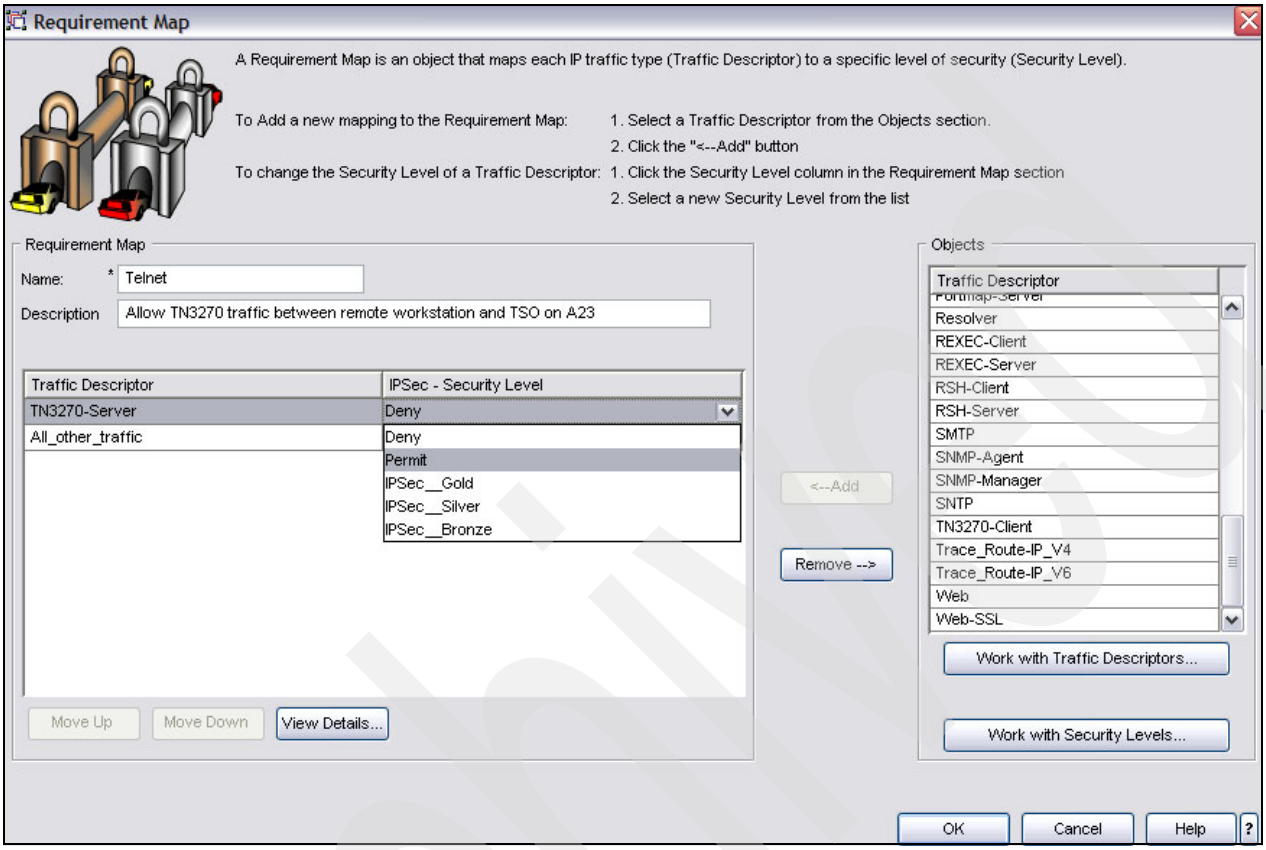


Figure 8-23 Select security level

18. This takes you back to the Connectivity Rule Requirement Map window, which contains the newly created Telnet map (Figure 8-24). Click **Next**.

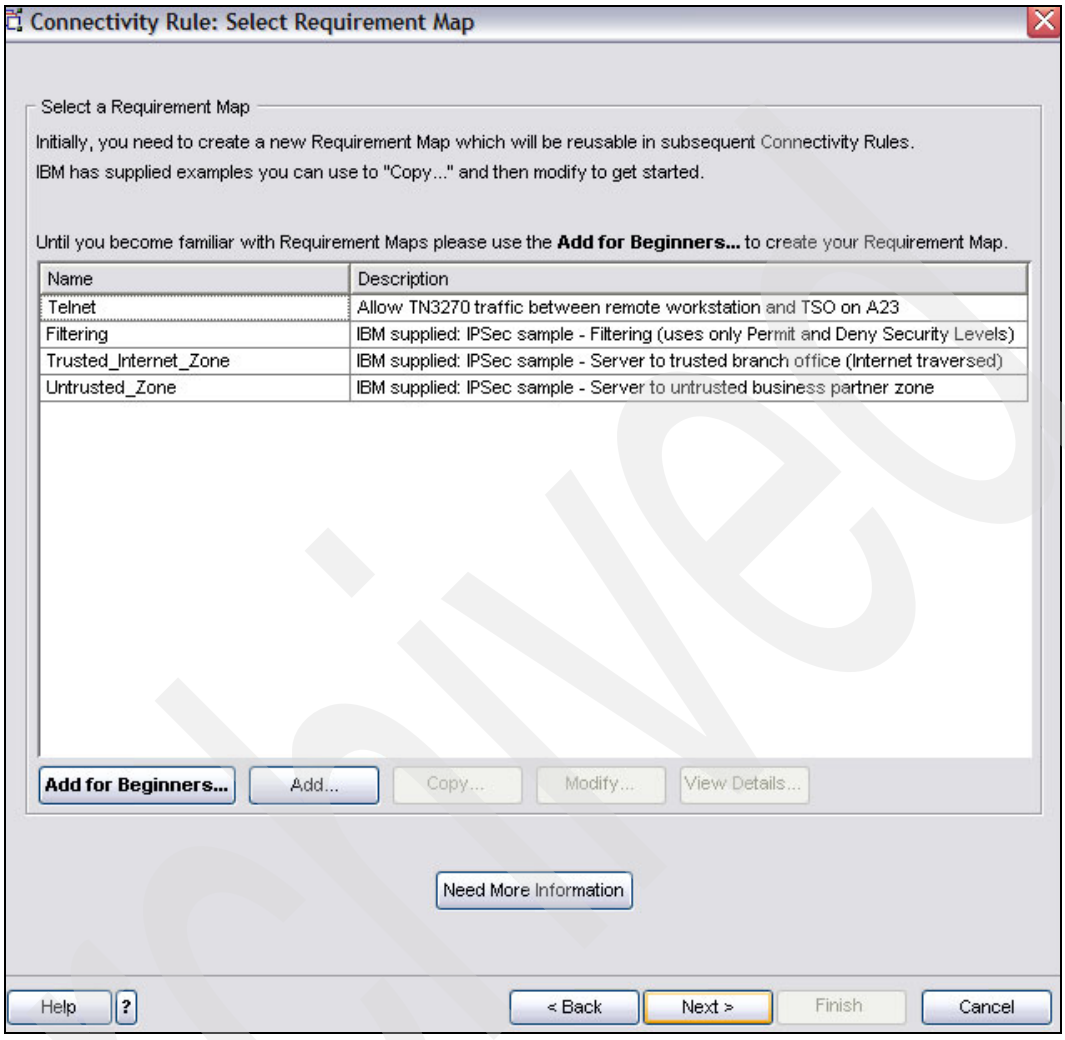


Figure 8-24 Connectivity Rule: Requirement Map

19. The Requirement Map: Additional Settings window shown in Figure 8-25 opens. Click **Finish**.

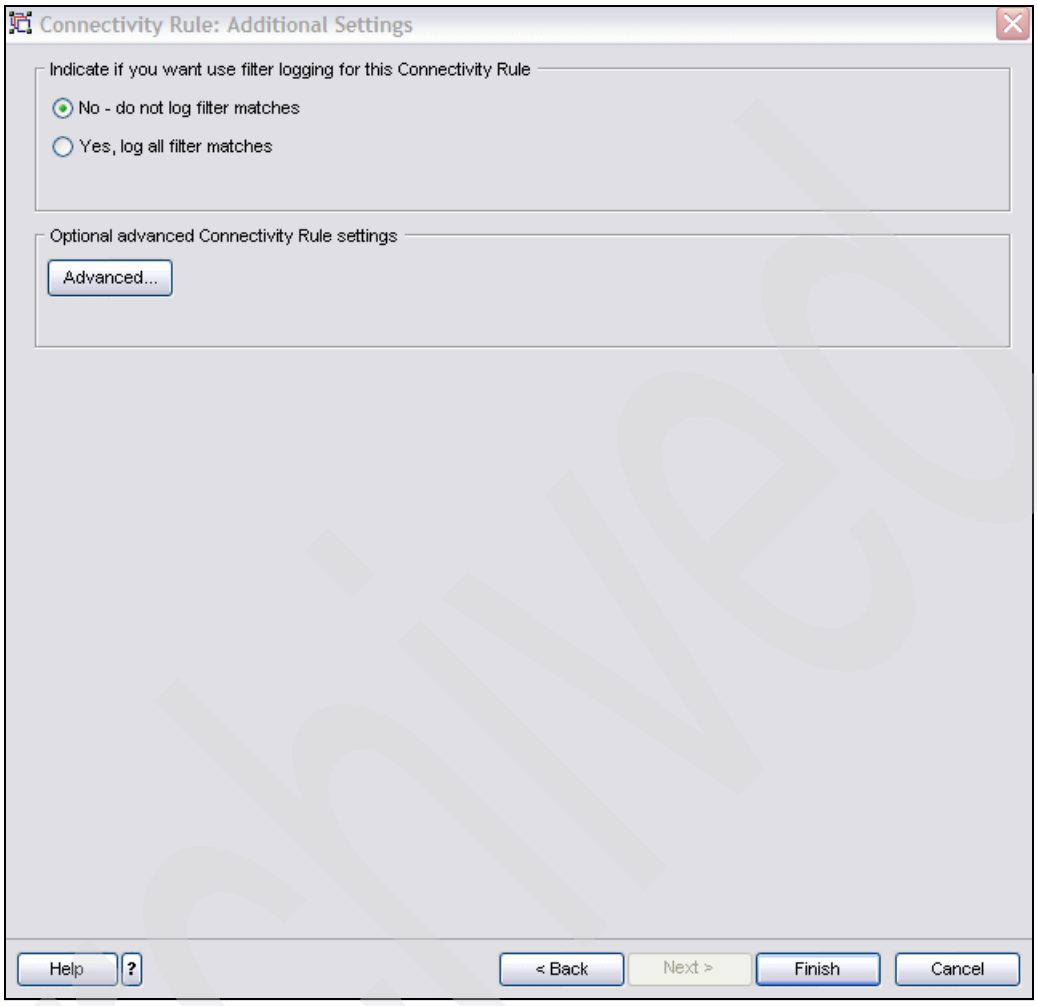


Figure 8-25 Connectivity Rule: Additional Settings

The window shown in Figure 8-26 shows the result of our definitions.

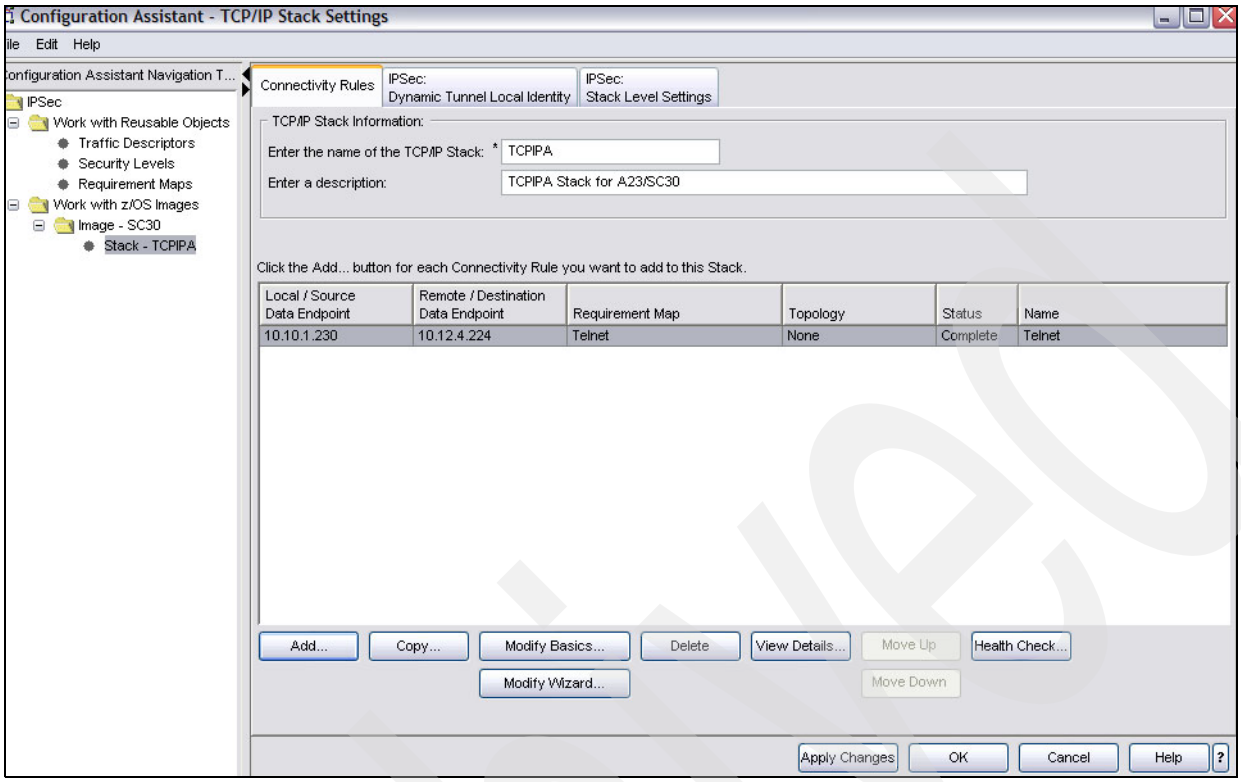


Figure 8-26 Stack settings

We have completed the connectivity rule for Telnet, and now we need to add another rule to allow FTP traffic between LPAR A23 (10.10.1.230) and LPAR A24 (10.10.1.241).

1. Click **Add** and the Connectivity Rule; Welcome window (Figure 8-16 on page 334) opens. We select the **Typical** (Default) rule type and click **Next**.
2. The window shown in Figure 8-27 opens. Select **No topology information is required** and click **Next**.

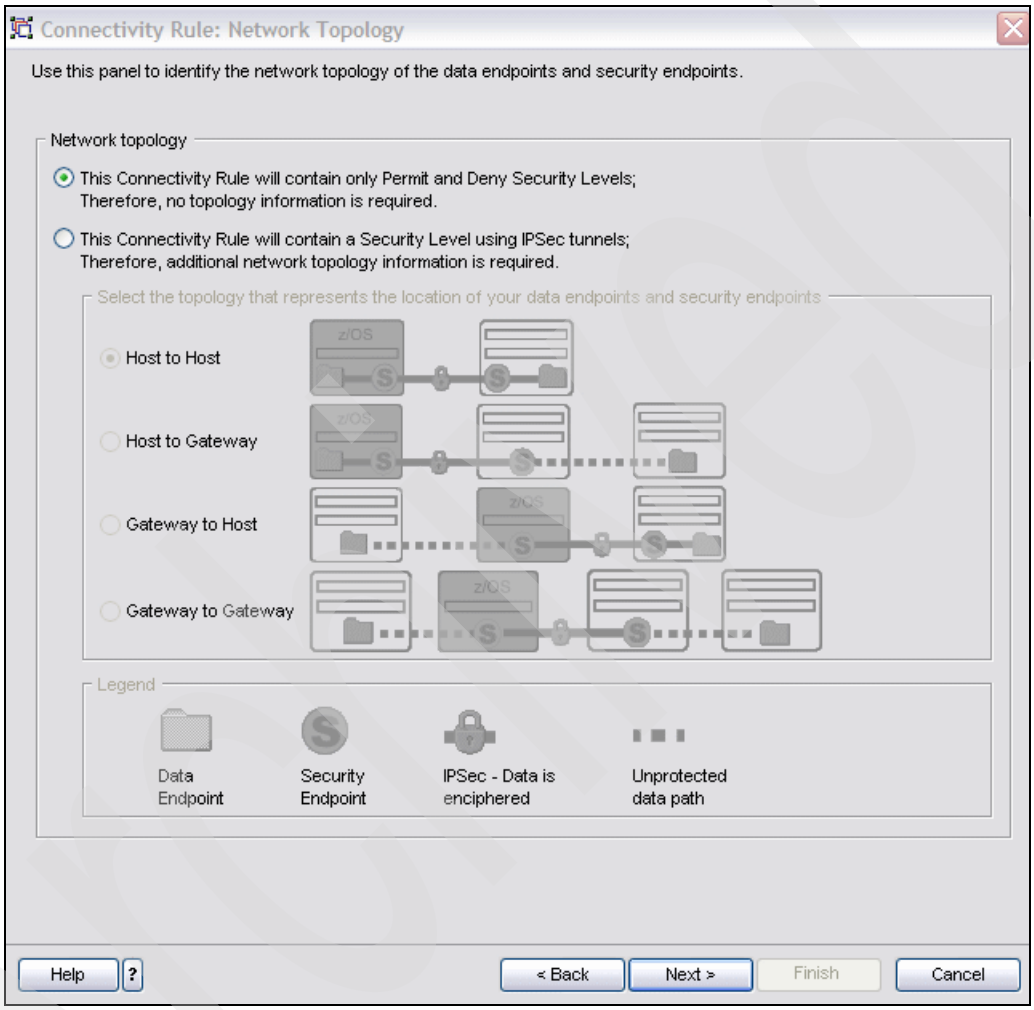


Figure 8-27 Network topology

3. In the window shown in Figure 8-28, we specify the IP addresses of the data endpoints (A23 and A24). This can be a single IP address, an IP address range, an IP subnet, or all IPv4 or all IPv6 addresses. Click **Next**.

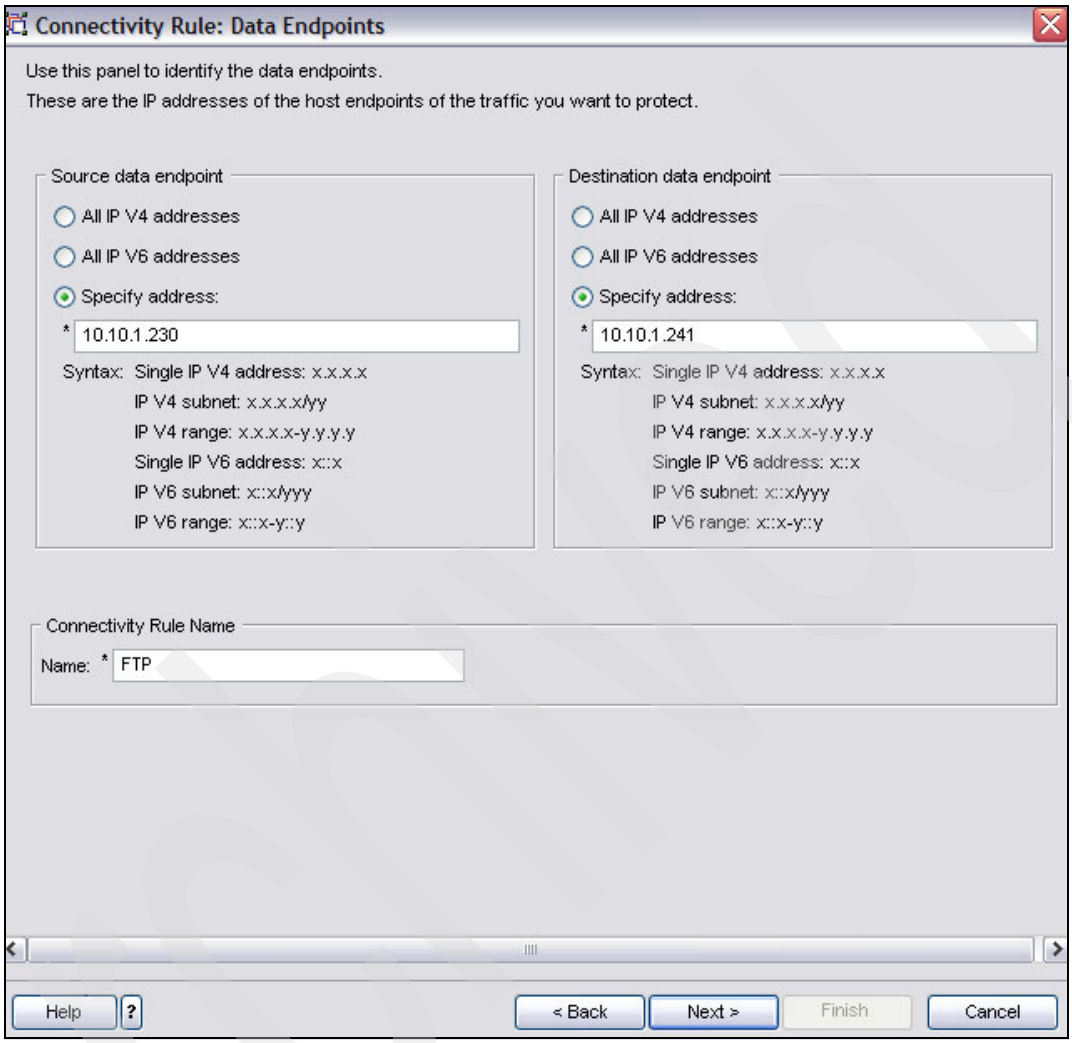


Figure 8-28 Connectivity rule

4. This takes you to the Connectivity Rule: Select Requirement Map window. Click **Add for Beginners**.
5. This opens the Requirement Map: Introduction window. Click **Next** to determine which traffic types to allow.

6. The window shown in Figure 8-29 opens. Select **FTP Client traffic** and **FTP Server traffic**. Click **Next**.

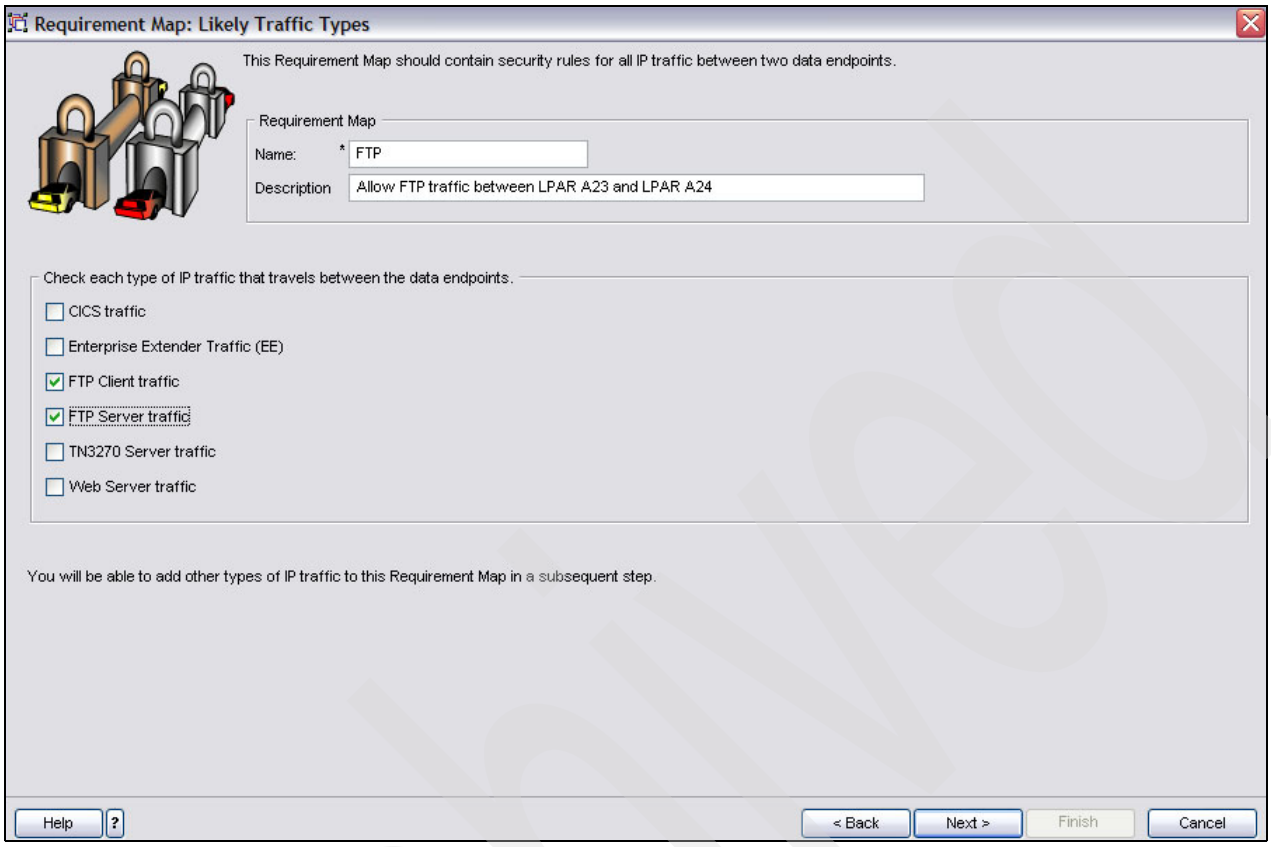


Figure 8-29 Traffic types

7. Figure 8-30 shows that three lines of Traffic Descriptors are generated with the following values:

   – FTP-Server
   – FTP-Client
   – All_other_traffic

   Note that the IPSECURITY implicit rules (deny all traffic) are in effect. Therefore, the all_other_traffic descriptor is not needed in the requirement map. If required, you can remove it by selecting **all_other_traffic** and clicking **Remove**.

   Click **Next**.

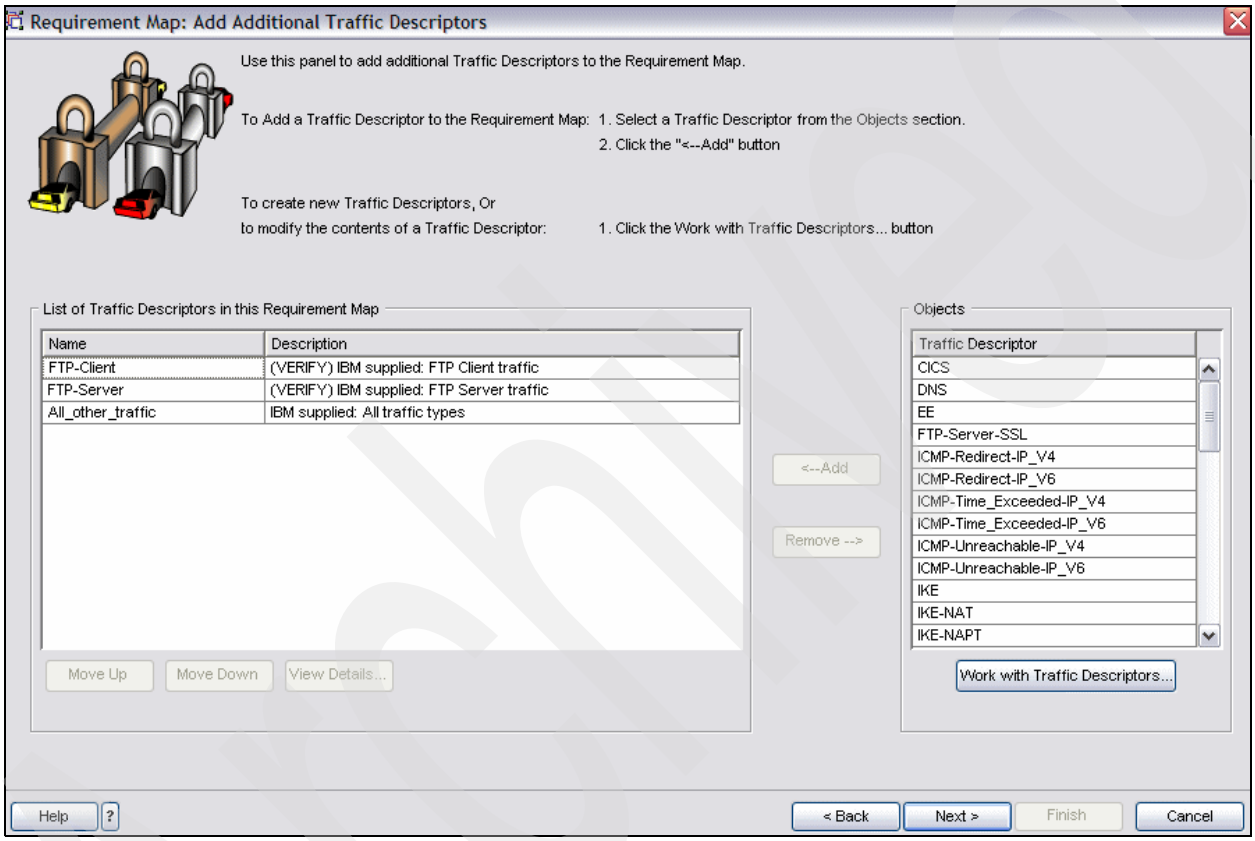


Figure 8-30 Add traffic descriptor

8. Set the security levels. Each entry in the Requirement Map is a mapping between a Traffic Descriptor and either Deny, Permit, or IPSec Security Level (see Figure 8-31). Click the I**IPSec - Security Level** tab for FTP-Server and select **Permit**. Do the same for FTP-Client traffic descriptor. Click **OK**.

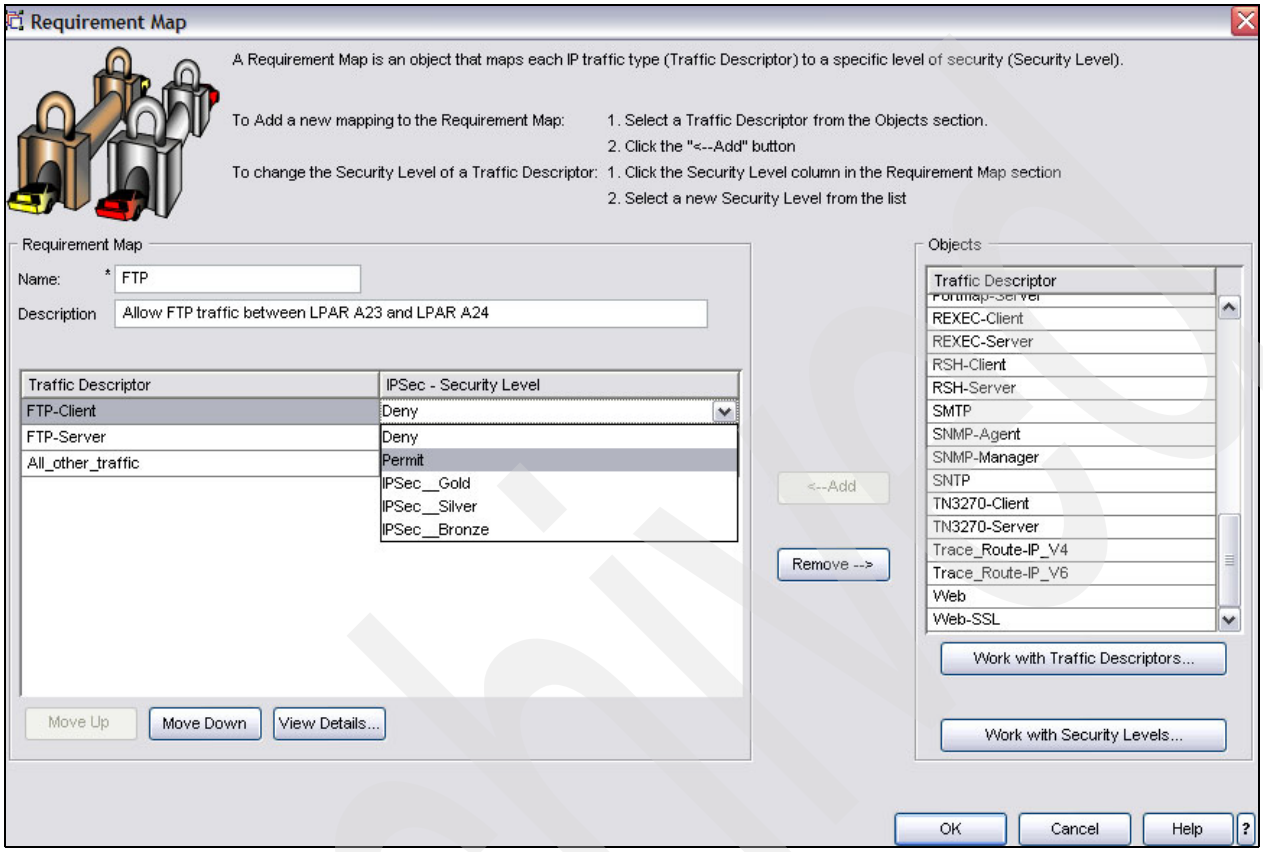


Figure 8-31 Select security level

9. This returns you back to the Connectivity Rule: Select Requirement Map window (see Figure 8-32). Click **Next**.

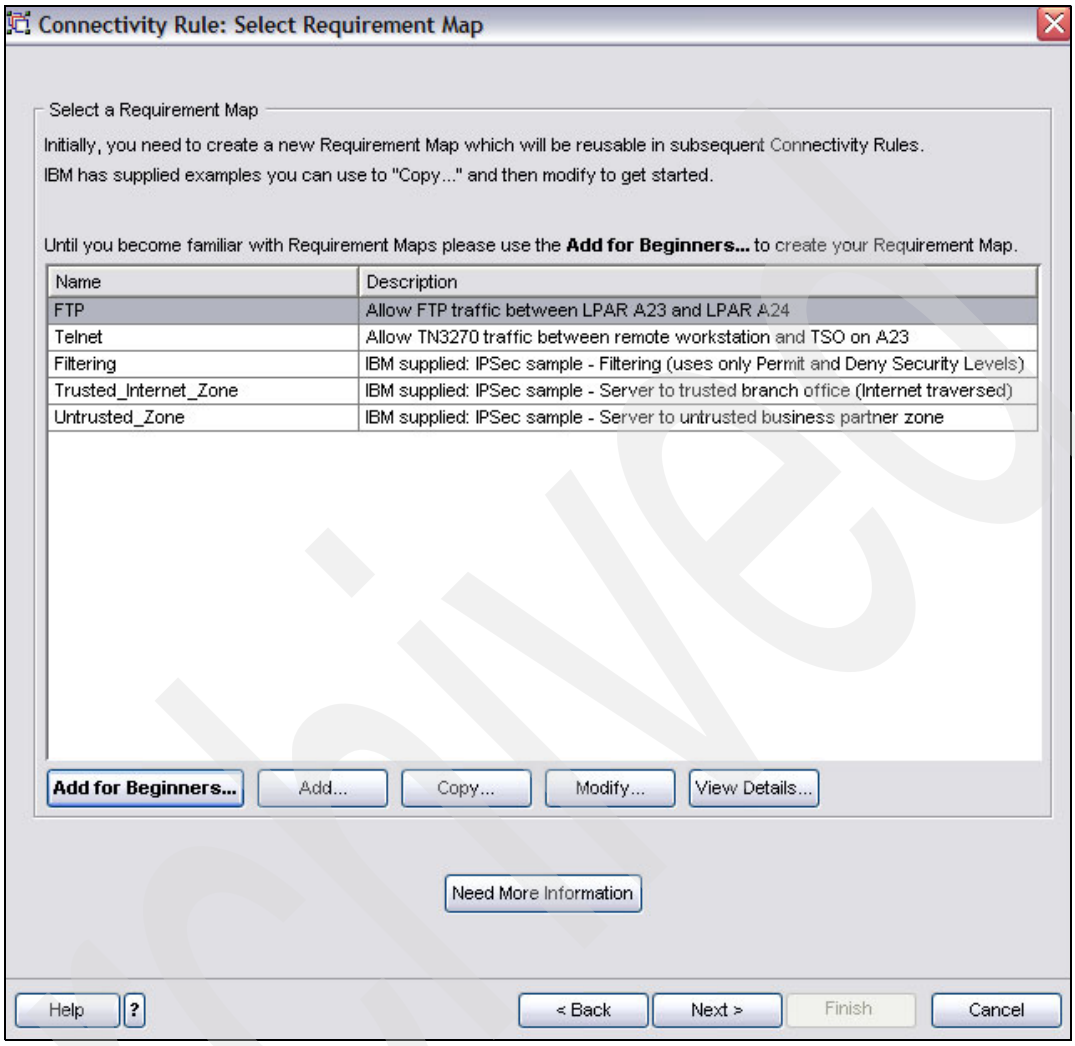


Figure 8-32 Connectivity Rule: Requirement Map

10. The Requirement Map: Additional Settings window shown in Figure 8-33 opens. Click **Finish**.

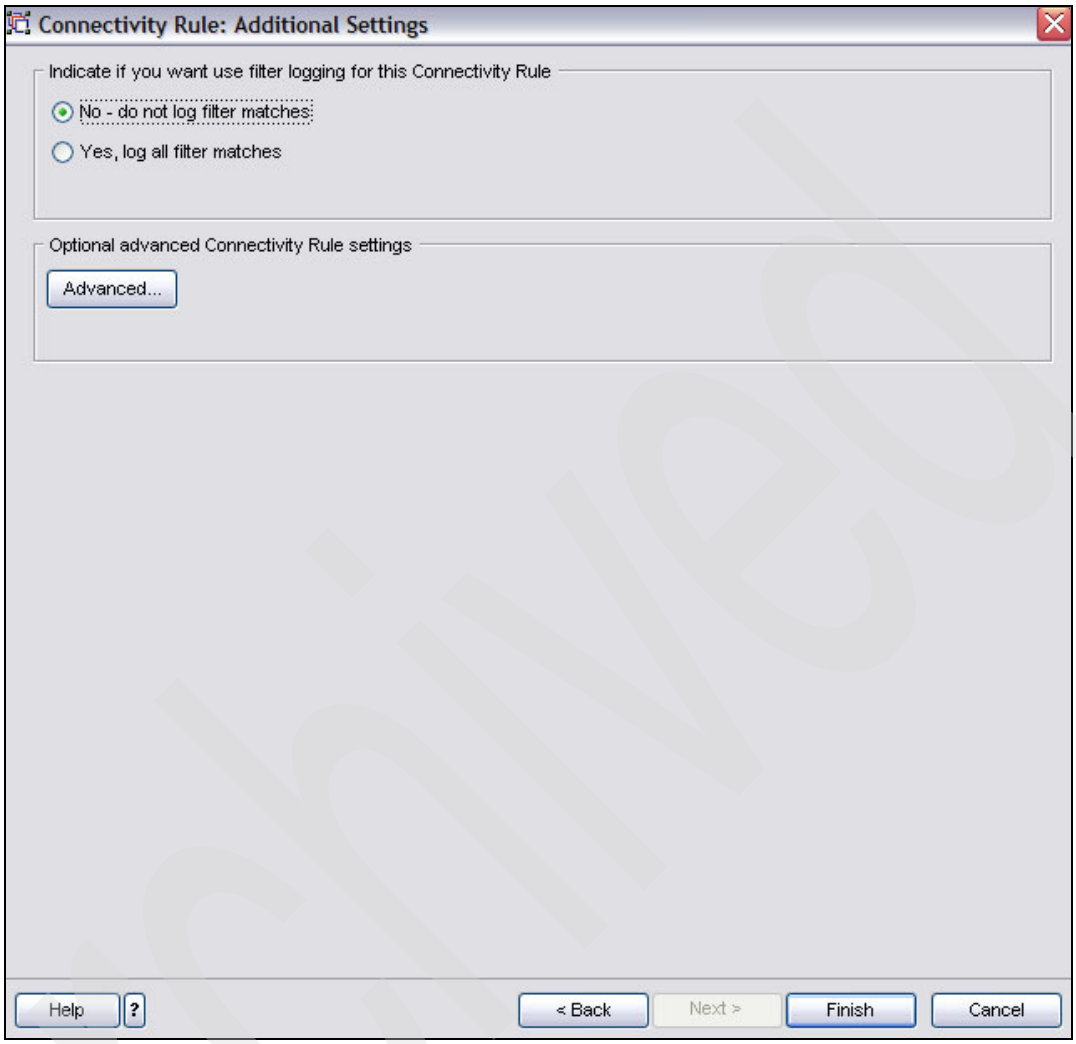


Figure 8-33 Connectivity Rule: Advanced Settings

Figure 8-34 shows the results of our definitions.

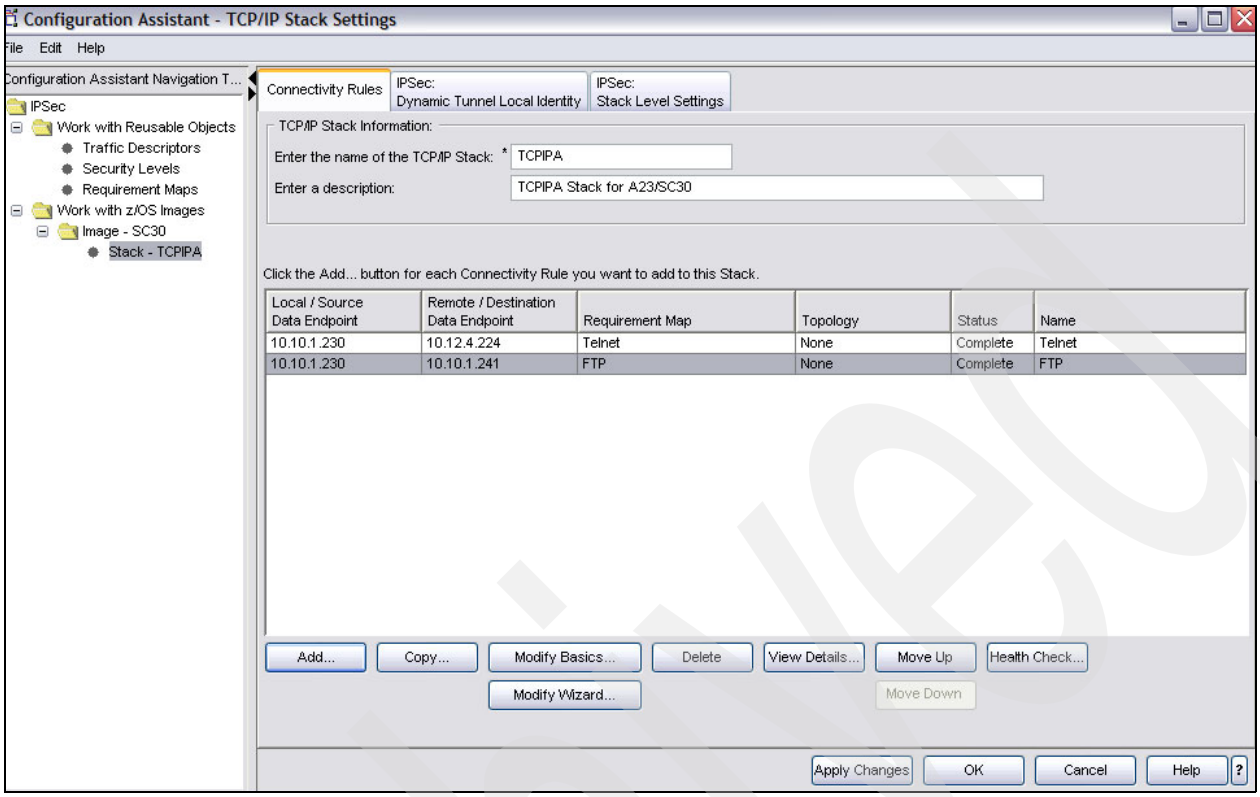


Figure 8-34 Stack settings

Next we need to add a requirement map for the basic services to permit ping, resolver, DNS, and omproute traffic between A23 and all data endpoints.

1. In the Configuration Assistant Navigation Tree (on the left), click **Requirement Maps** (under **Work with Reusable Objects**); see Figure 8-35. Click **Add**.

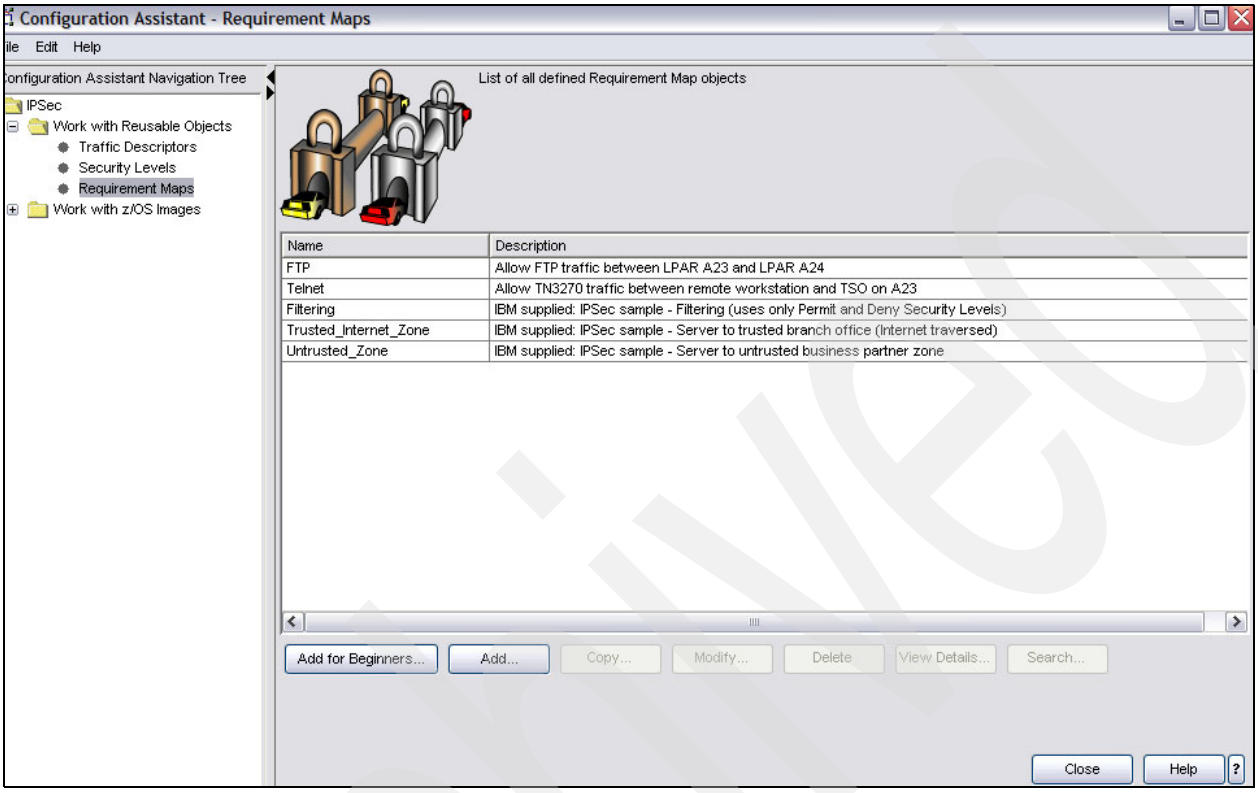


Figure 8-35 Adding a requirement map

2. Provide a name and description, as shown in Figure 8-36.

   Remove All_other_traffic by selecting **All_other_traffic** and clicking **Remove**.

   Add ping, resolver, DNS, and omproute by selecting the appropriate traffic descriptors from the Objects list and clicking **Add**.

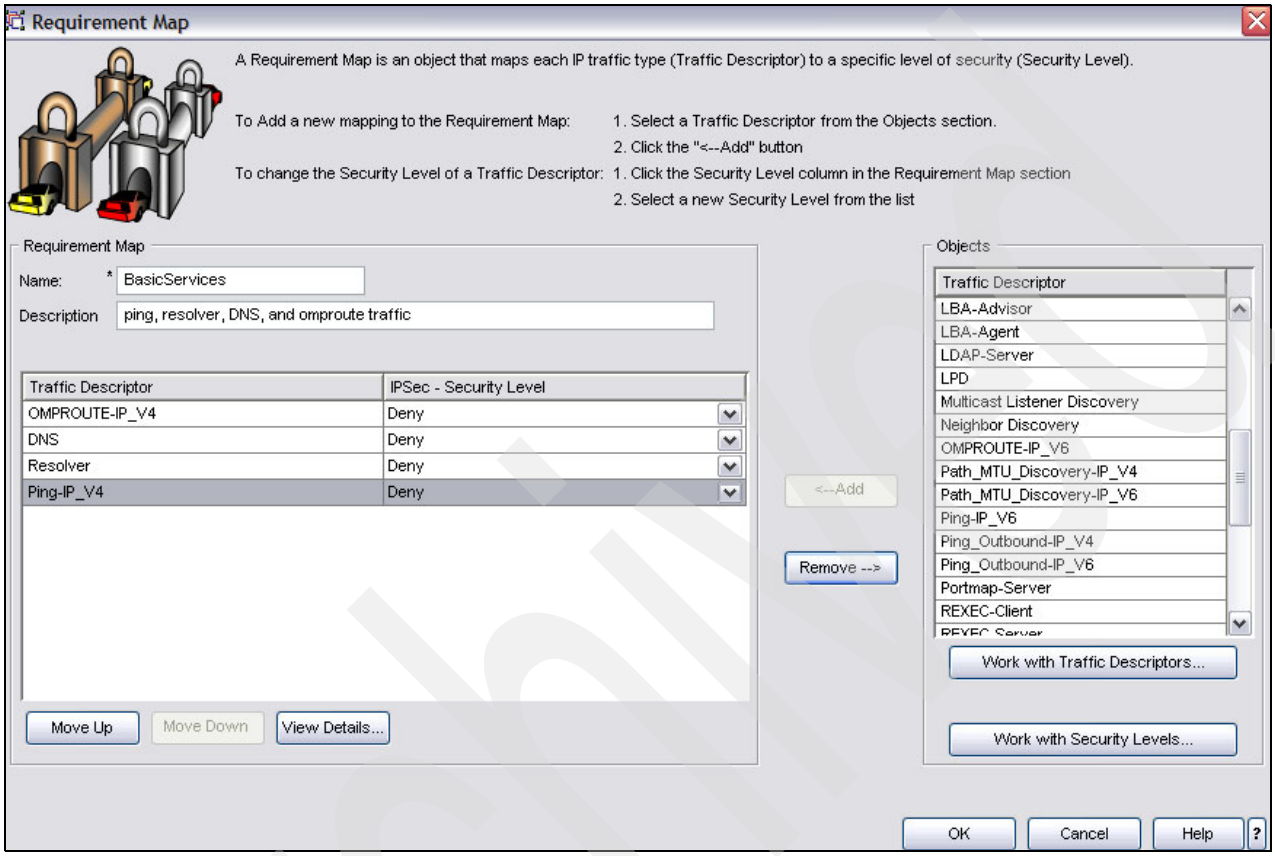


Figure 8-36 Removing and adding traffic descriptors

3. Click the I**PSec - Security Level** tab and select **Permit** for each traffic descriptor; see Figure 8-37. Click **OK**.

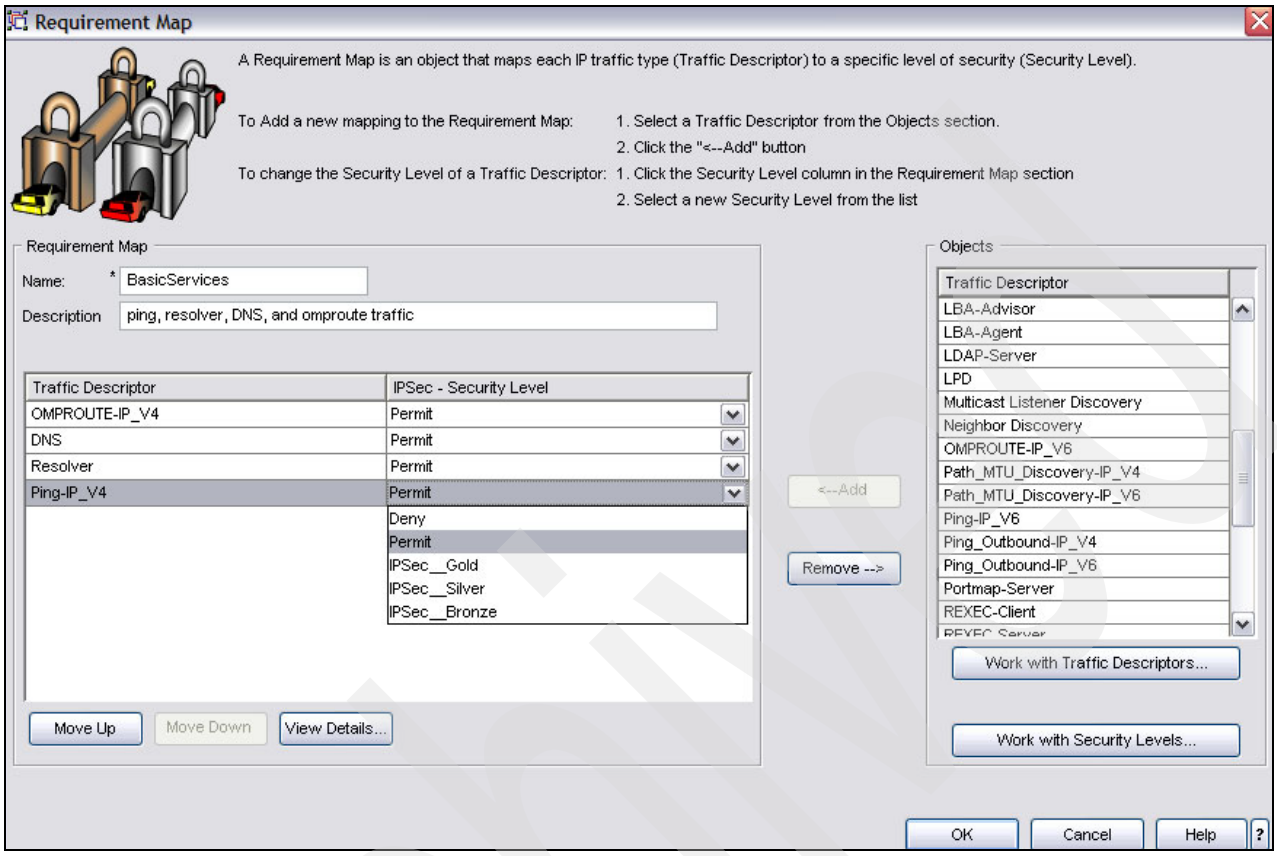


Figure 8-37 Change security level

4. The list of requirement maps opens; see Figure 8-38.

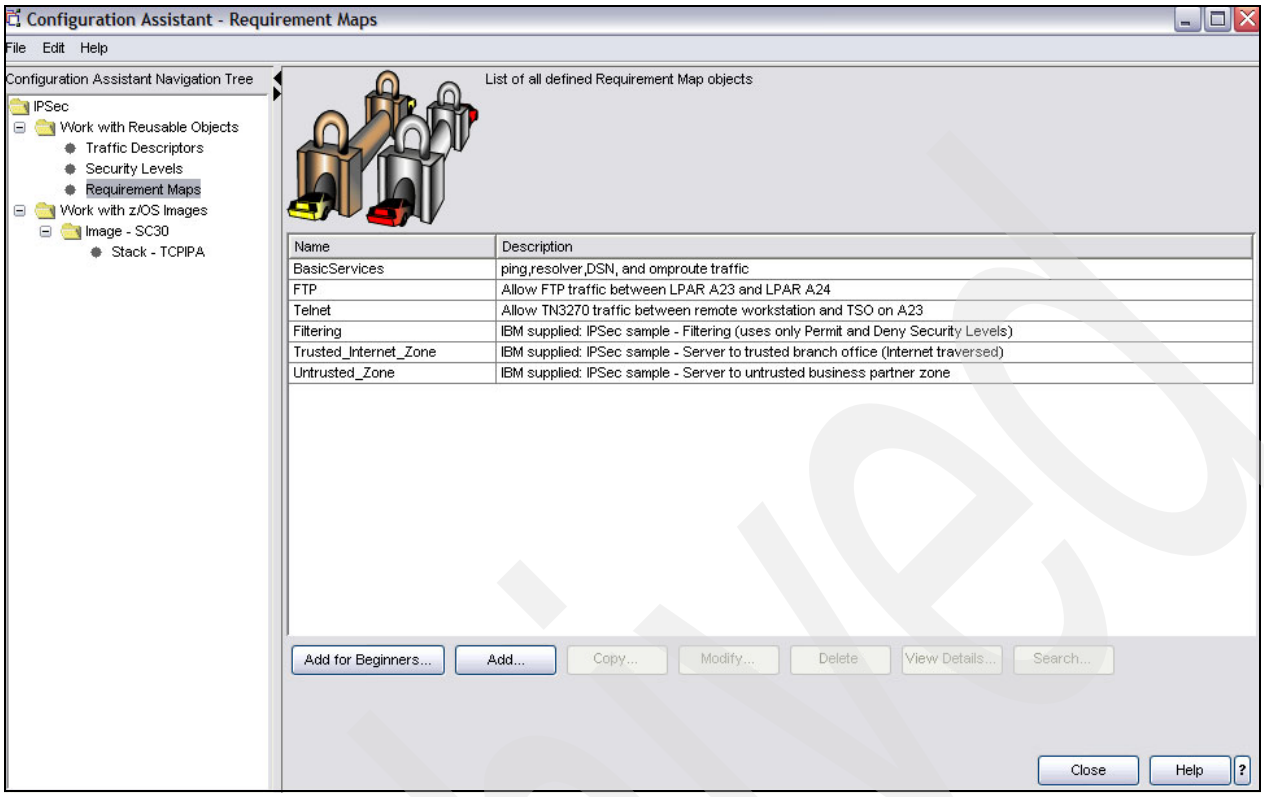


Figure 8-38 Requirement Maps

In the Configuration Assistant Navigation Tree, click the required TCP/IP stack (under **Work with z/OS Images**); see Figure 8-39. Click **Add**.

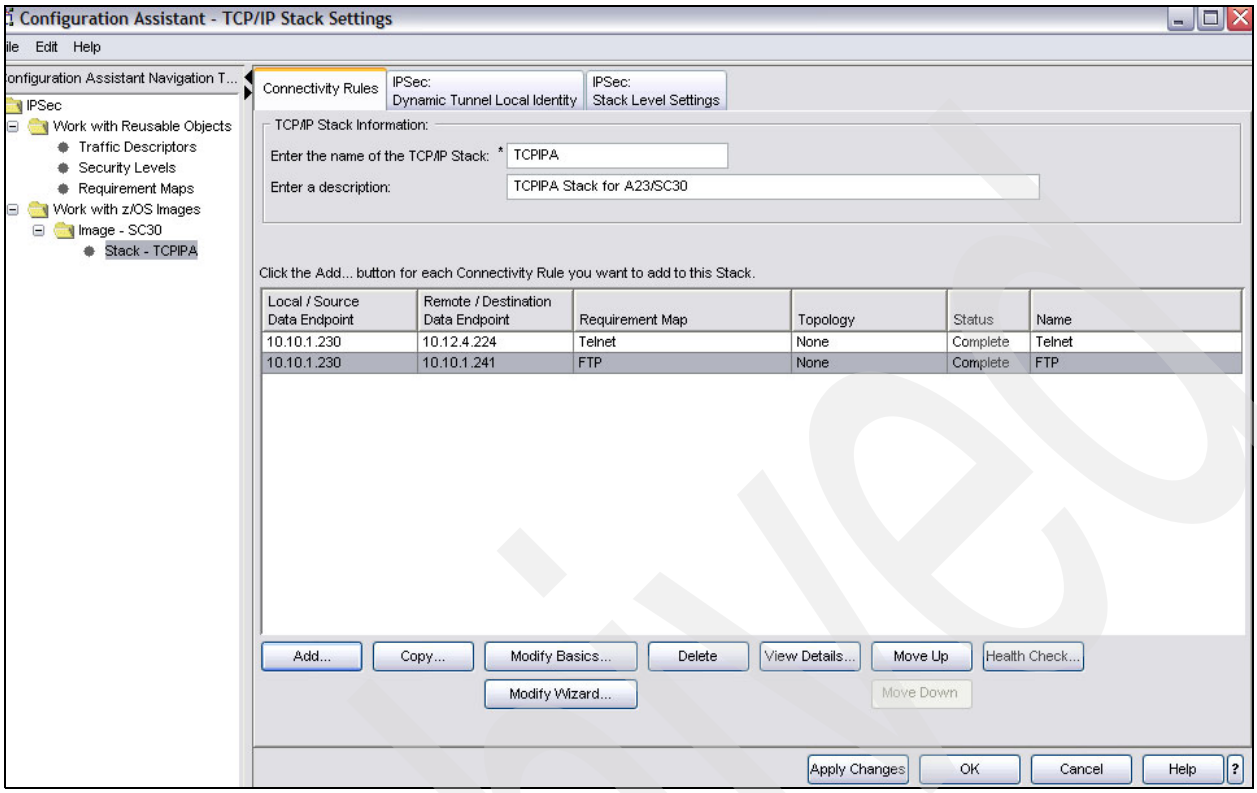


Figure 8-39 z/OS image list

5. The Connectivity Rule: Welcome window opens. We select the **Typical** (Default) rule type and click **Next**.

6. In the window shown in Figure 8-40, select **No topology information is required** and click **Next**.

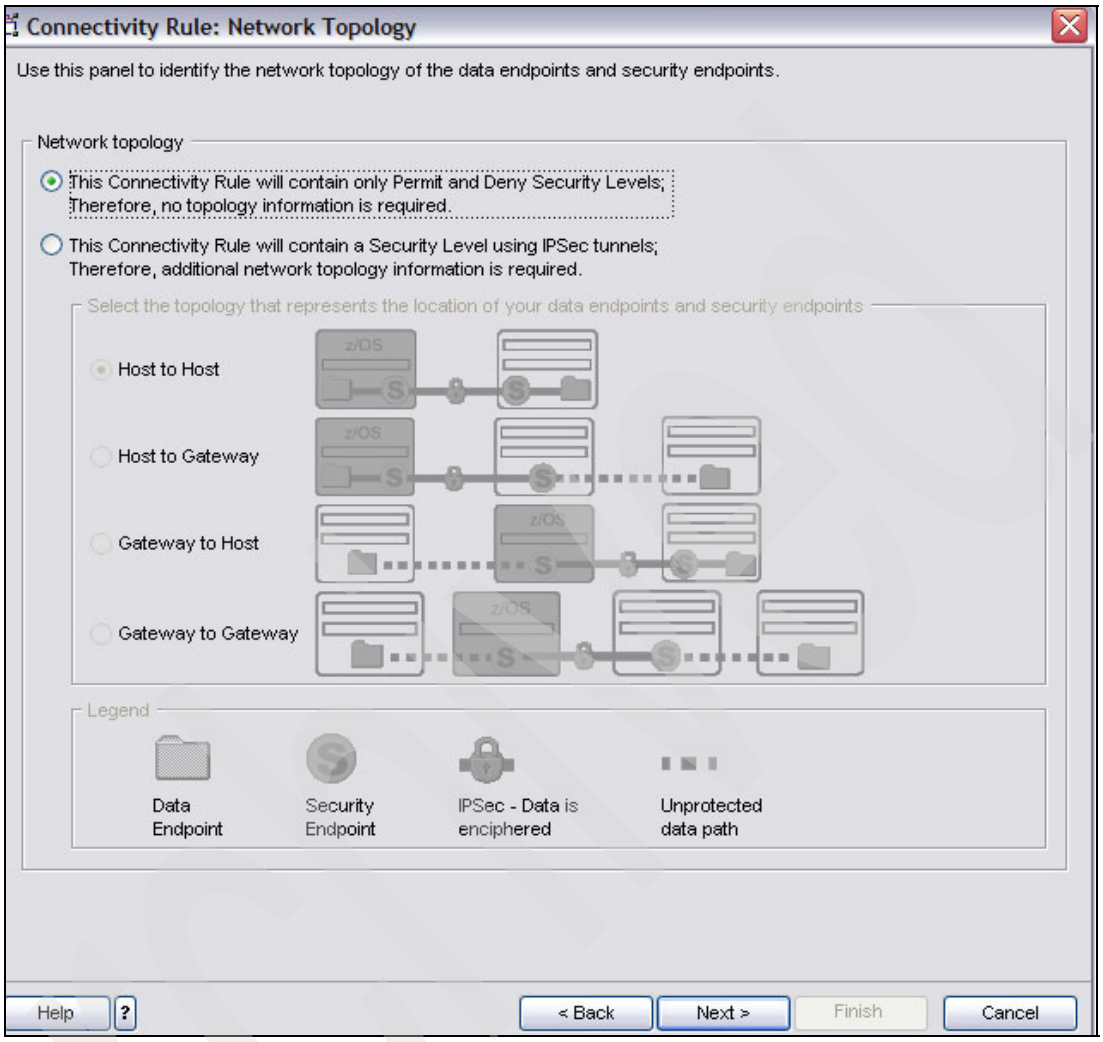


Figure 8-40 Network topology

7. In Figure 8-41, we specify **All IP V4 addresses**, because this rule applies to all data endpoints in our network. Click **Next**.

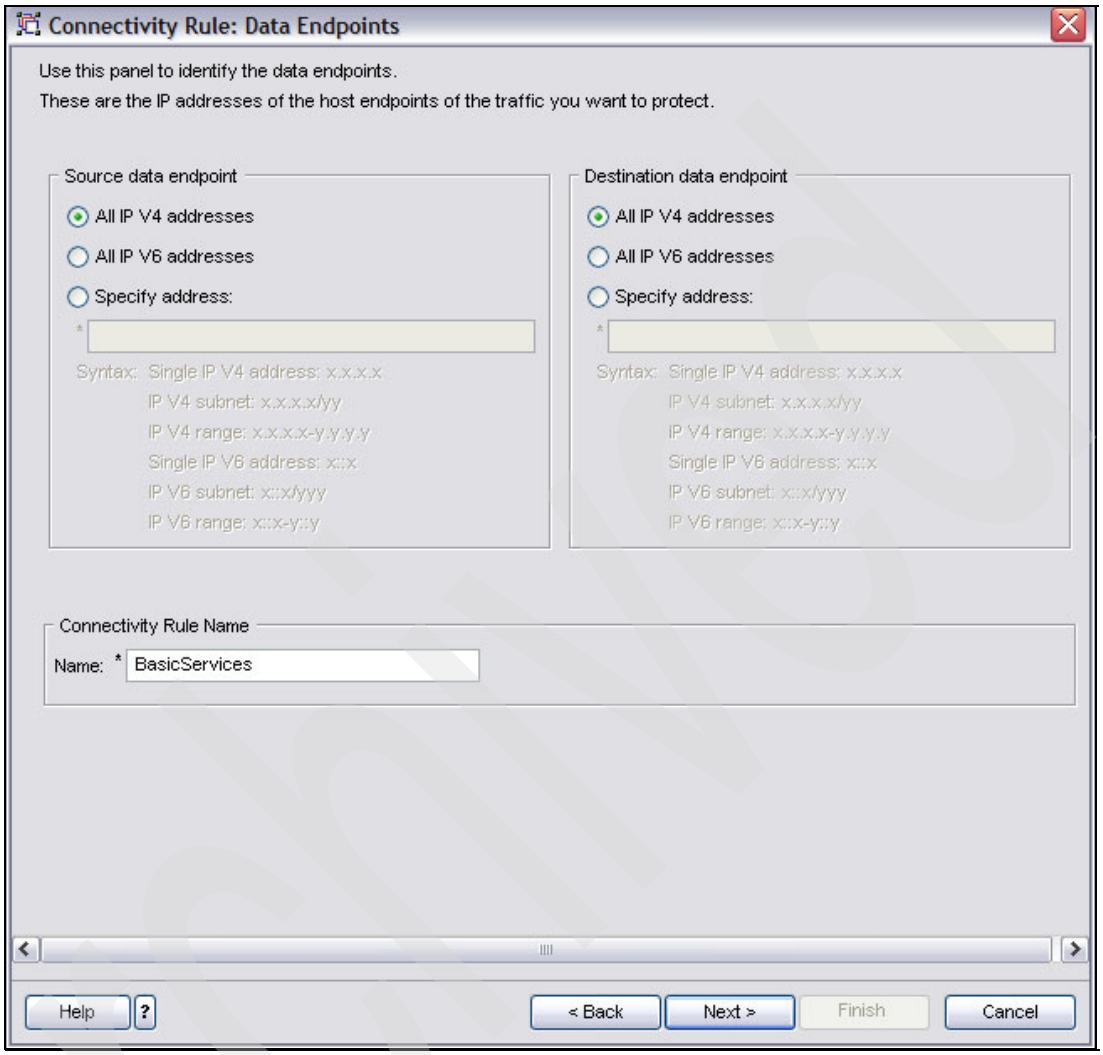


Figure 8-41 Connectivity rule

8. This opens the Select Requirement Map window; see Figure 8-42. Select a requirement map (in our case, it was `BasicServices`), and click **Next**.

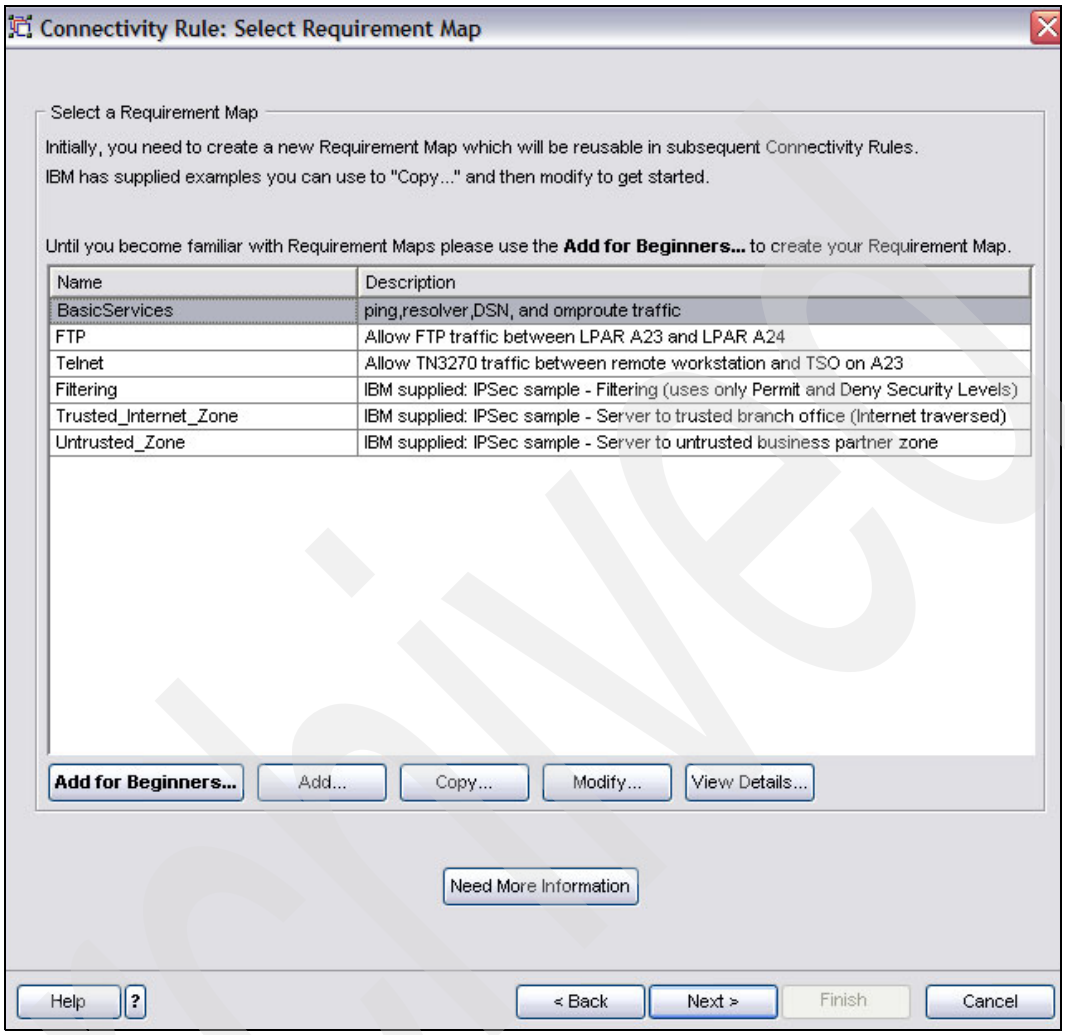


Figure 8-42 Requirement map list

9. The TCP/IP stack settings opens; see Figure 8-43 on page 361. Because we want BasicServices traffic to flow between endpoints in all cases, it must be moved to the top of the list. This is done by clicking **Move Up**.

> **Note:** The IP addresses in a packet are compared with the IP addresses of the data endpoints of the connectivity rules in the order that those rules appear in the table.

Click **OK**.

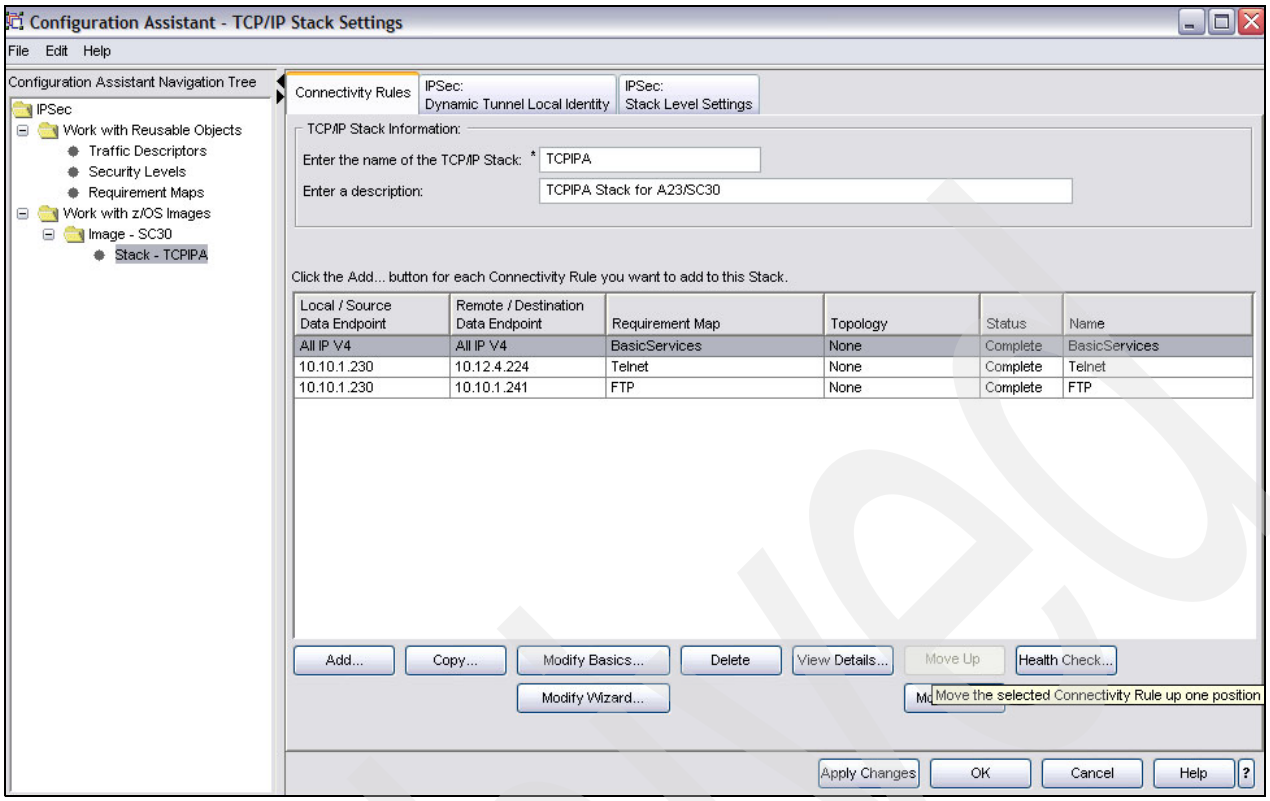


Figure 8-43 TCP/IP stack settings

10. This takes you back to the Connectivity Rule Requirement Map window, which contain the newly created basic services map. Click **Health Check...** to determine if the TCP/IP stack contains configuration errors or inappropriate combinations of rules.

With our IP filtering rules complete, we install the configuration files as follows:

1. Select **Edit** in the top left corner, or right-click the stack name, and click **Install Configuration Files**. In Figure 8-44, we see that there are two files to be installed:
   - A sample TCP/IP profile to insert in the profile TCP/IP
   - An IPSec configuration file

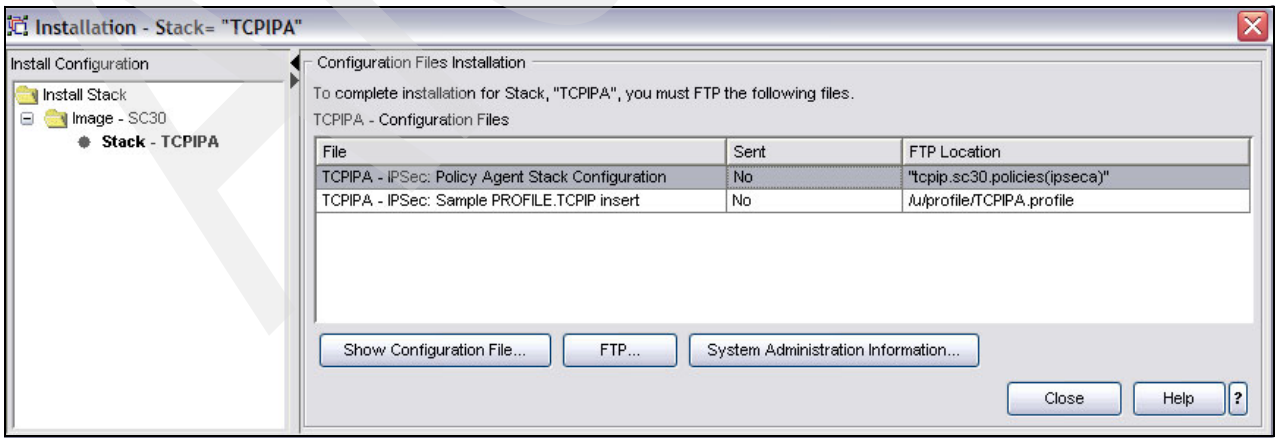


Figure 8-44 Installation stack

We have two options:

- **Show Configuration File**, save the file on your computer using **Save As**, and then upload the file to z/OS later.
- Directly **FTP** the file to z/OS.

In the following examples, we show each of the configuration files and then an example of the FTP.

2. We select **IPSec: Policy Agent Stack Configuration**, and click **Show Configuration File**. The file in Example 8-2 is generated by the Configuration Assistant.

Example 8-2 IPSec configuration file

```
##
## IPSec Policy Agent Configuration file for:
##     Image: SC30
##     Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Thu Dec 07 15:53:28 EST 2006
##
## Copyright = None
##

IpGenericFilterAction          Permit~LogNo
{
  IpFilterAction               Permit
  IpFilterLogging              No
}

IpGenericFilterAction          Deny~LogNo
{
  IpFilterAction               Deny
  IpFilterLogging              No
}

IpService                      OMPROUTE-IP_V4
{
  Protocol                     OSPF
  Type                         Any
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~1
{
  Protocol                     IGMP
  Direction                    BiDirectional
  Routing                      Either
}

IpService                      OMPROUTE-IP_V4~2
{
  Protocol                     UDP
  SourcePortRange              520
  DestinationPortRange         1024 65535
  Direction                    BiDirectional
  Routing                      Either
}
```

```
IpService                       OMPROUTE-IP_V4~3
{
  Protocol                      UDP
  SourcePortRange               1024 65535
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       OMPROUTE-IP_V4~4
{
  Protocol                      UDP
  SourcePortRange               520
  DestinationPortRange          520
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS
{
  Protocol                      UDP
  SourcePortRange               53
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~5
{
  Protocol                      UDP
  SourcePortRange               53
  DestinationPortRange          53
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~6
{
  Protocol                      TCP
  SourcePortRange               53
  DestinationPortRange          1024 65535
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       DNS~7
{
  Protocol                      TCP
  SourcePortRange               53
  DestinationPortRange          53
  Direction                     BiDirectional
  Routing                       Either
}

IpService                       Resolver
{
  Protocol                      TCP
  SourcePortRange               1024 65535
  DestinationPortRange          53
```

```
  Direction                   BiDirectional OutboundConnect
  Routing                     Local
}

IpService                     Resolver~8
{
  Protocol                    UDP
  SourcePortRange             1024 65535
  DestinationPortRange        53
  Direction                   BiDirectional
  Routing                     Local
}

IpService                     Ping-IP_V4
{
  Protocol                    ICMP
  Type                        8
  Code                        Any
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     Ping-IP_V4~9
{
  Protocol                    ICMP
  Type                        0
  Code                        Any
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     TN3270-Server
{
  Protocol                    TCP
  SourcePortRange             23
  DestinationPortRange        1024 65535
  Direction                   BiDirectional InboundConnect
  Routing                     Either
}

IpService                     All_other_traffic
{
  Protocol                    All
  Direction                   BiDirectional
  Routing                     Either
}

IpService                     FTP-Client
{
  Protocol                    TCP
  SourcePortRange             1024 65535
  DestinationPortRange        21
  Direction                   BiDirectional OutboundConnect
  Routing                     Either
}

IpService                     FTP-Client~10
{
  Protocol                    TCP
  SourcePortRange             1024 65535
```

```
  DestinationPortRange         20
  Direction                    BiDirectional InboundConnect
  Routing                      Either
}

IpService                      FTP-Client~11
{
  Protocol                     TCP
  SourcePortRange              1024 65535
  DestinationPortRange         50000 50200
  Direction                    BiDirectional OutboundConnect
  Routing                      Either
}

IpService                      FTP-Server
{
  Protocol                     TCP
  SourcePortRange              21
  DestinationPortRange         1024 65535
  Direction                    BiDirectional InboundConnect
  Routing                      Either
}

IpService                      FTP-Server~12
{
  Protocol                     TCP
  SourcePortRange              20
  DestinationPortRange         1024 65535
  Direction                    BiDirectional OutboundConnect
  Routing                      Either
}

IpService                      FTP-Server~13
{
  Protocol                     TCP
  SourcePortRange              50000 50200
  DestinationPortRange         1024 65535
  Direction                    BiDirectional InboundConnect
  Routing                      Either
}
##
## Connectivity Rule BasicServices combines the following items:
##   Local data endpoint        All4
##   Remote data endpoint       All4
##   Topology                   None (Permit/Deny only)
##   Requirement Map            BasicServices
##     OMPROUTE-IP_V4             => Permit
##     DNS                        => Permit
##     Resolver                   => Permit
##     Ping-IP_V4                 => Permit

IpFilterRule                   BasicServices~1
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 OMPROUTE-IP_V4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~2
```

```
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                OMPROUTE-IP_V4~1
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~3
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                OMPROUTE-IP_V4~2
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~4
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                OMPROUTE-IP_V4~3
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~5
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                OMPROUTE-IP_V4~4
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~6
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                DNS
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~7
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                DNS~5
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~8
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                DNS~6
  IpGenericFilterActionRef    Permit~LogNo
}

IpFilterRule                  BasicServices~9
{
  IpSourceAddr                All4
  IpDestAddr                  All4
  IpServiceRef                DNS~7
```

```
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~10
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 Resolver
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~11
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 Resolver~8
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~12
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 Ping-IP_V4
  IpGenericFilterActionRef     Permit~LogNo
}

IpFilterRule                   BasicServices~13
{
  IpSourceAddr                 All4
  IpDestAddr                   All4
  IpServiceRef                 Ping-IP_V4~9
  IpGenericFilterActionRef     Permit~LogNo
}
##
## Connectivity Rule Telnet combines the following items:
##   Local data endpoint       Telnet~ADR~1
##   Remote data endpoint      Telnet~ADR~2
##   Topology                  None (Permit/Deny only)
##   Requirement Map           Telnet
##     TN3270-Server             => Permit
##     All_other_traffic         => Deny

IpAddr                         Telnet~ADR~1
{
  Addr                         10.10.1.230
}

IpAddr                         Telnet~ADR~2
{
  Addr                         10.12.4.224
}

IpFilterRule                   Telnet~3
{
  IpSourceAddrRef              Telnet~ADR~1
  IpDestAddrRef                Telnet~ADR~2
  IpServiceRef                 TN3270-Server
  IpGenericFilterActionRef     Permit~LogNo
}
```

```
IpFilterRule                    Telnet~4
{
  IpSourceAddrRef               Telnet~ADR~1
  IpDestAddrRef                 Telnet~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      Deny~LogNo
}
##
## Connectivity Rule FTP combines the following items:
##   Local data endpoint        FTP~ADR~1
##   Remote data endpoint       FTP~ADR~2
##   Topology                   None (Permit/Deny only)
##   Requirement Map            FTP
##     FTP-Client                 => Permit
##     FTP-Server                 => Permit
##     All_other_traffic          => Deny

IpAddr                          FTP~ADR~1
{
  Addr                          10.10.1.230
}

IpAddr                          FTP~ADR~2
{
  Addr                          10.10.1.241
}

IpFilterRule                    FTP~3
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~4
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client~10
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~5
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Client~11
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~6
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Server
  IpGenericFilterActionRef      Permit~LogNo
}
```

```
IpFilterRule                    FTP~7
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Server~12
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~8
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  FTP-Server~13
  IpGenericFilterActionRef      Permit~LogNo
}

IpFilterRule                    FTP~9
{
  IpSourceAddrRef               FTP~ADR~1
  IpDestAddrRef                 FTP~ADR~2
  IpServiceRef                  All_other_traffic
  IpGenericFilterActionRef      Deny~LogNo
}

IpFilterPolicy
{
  PreDecap                      OFF
  FilterLogging                 ON
  IpFilterLogImplicit           Yes
  AllowOnDemand                 Yes
  IpFilterRuleRef               BasicServices~1
  IpFilterRuleRef               BasicServices~2
  IpFilterRuleRef               BasicServices~3
  IpFilterRuleRef               BasicServices~4
  IpFilterRuleRef               BasicServices~5
  IpFilterRuleRef               BasicServices~6
  IpFilterRuleRef               BasicServices~7
  IpFilterRuleRef               BasicServices~8
  IpFilterRuleRef               BasicServices~9
  IpFilterRuleRef               BasicServices~10
  IpFilterRuleRef               BasicServices~11
  IpFilterRuleRef               BasicServices~12
  IpFilterRuleRef               BasicServices~13
  IpFilterRuleRef               Telnet~3
  IpFilterRuleRef               Telnet~4
  IpFilterRuleRef               FTP~3
  IpFilterRuleRef               FTP~4
  IpFilterRuleRef               FTP~5
  IpFilterRuleRef               FTP~6
  IpFilterRuleRef               FTP~7
  IpFilterRuleRef               FTP~8
  IpFilterRuleRef               FTP~9
}
```

Figure 8-45 shows the FTP process for transferring the configuration files.

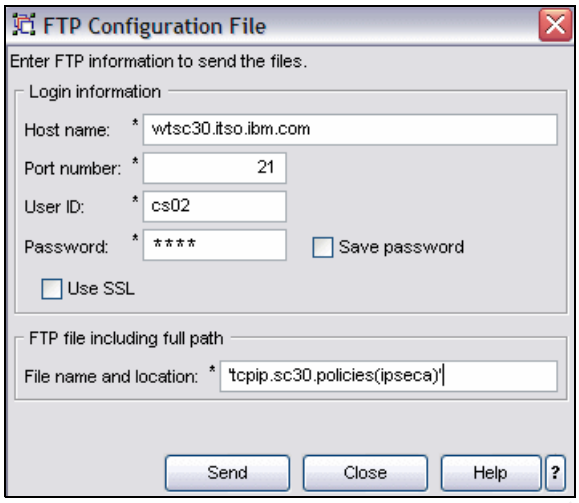


Figure 8-45 File transfer

8.3.3 Verification

We execute the following tasks to verify our IP filtering configurations:

- Log on to LPAR A23 using TN3270
- FTP to LPAR A23 from a remote workstation (should fail)
- FTP from LPAR A24 to LPAR A23

Log on to LPAR A23 using TN3270

From our workstation we create a TELNET session to TCPIPA (10.10.1.230) on A23 (SC30); see Figure 8-46 on page 371.

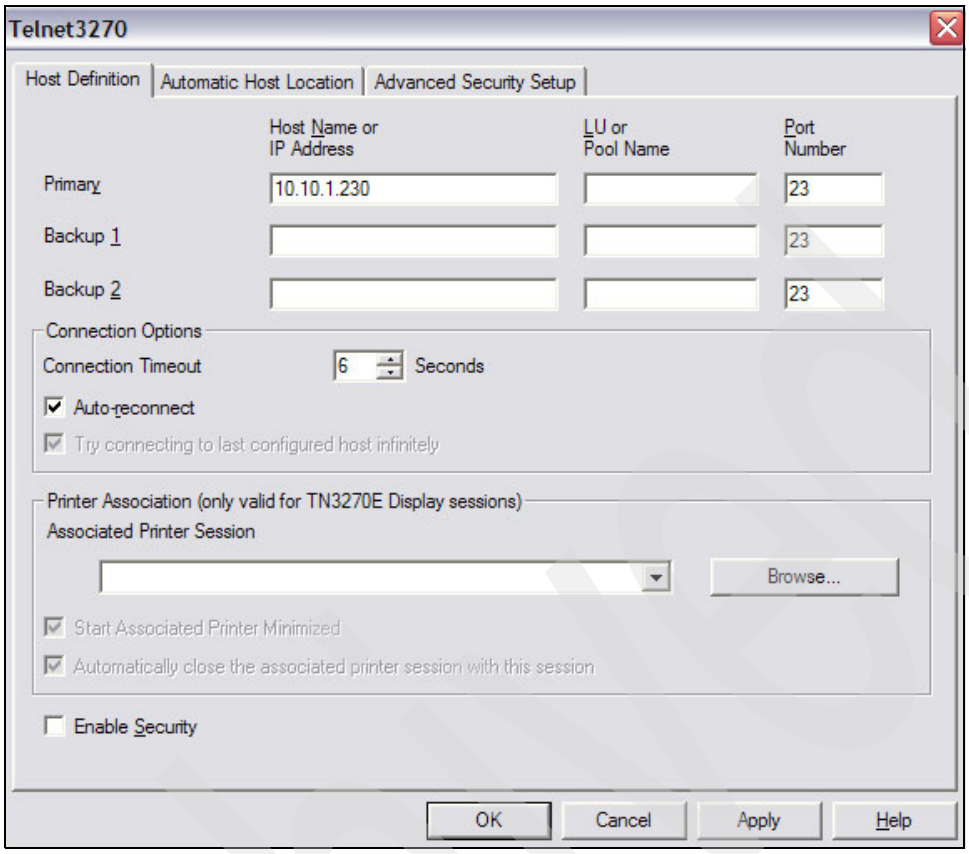


Figure 8-46 TN3270 Host definition

Figure 8-47 shows that the connection between the remote workstation (10.12.4.224) and LPAR A23 (SC30) was established.

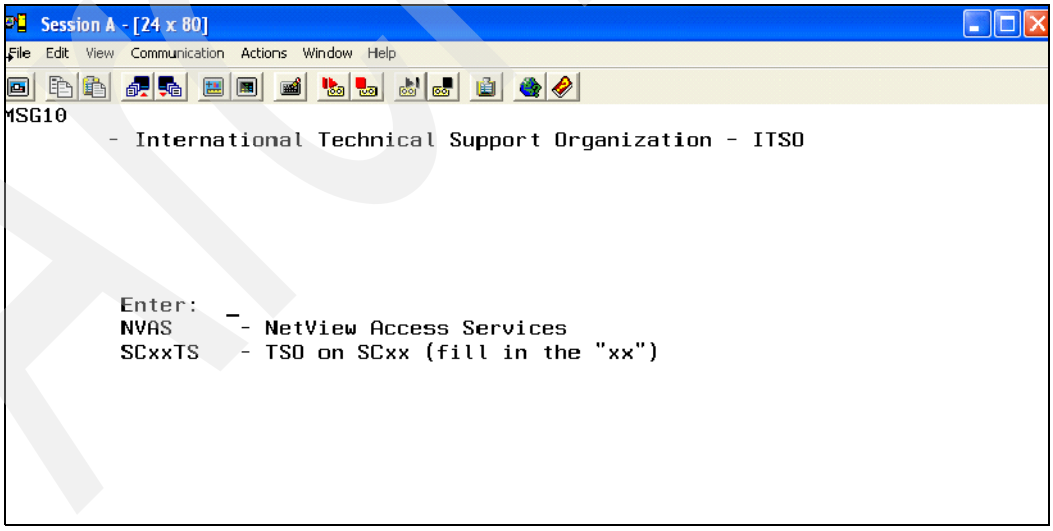


Figure 8-47 TN3270 connection

FTP to LPAR A23 from a remote workstation (should fail)

From the remote workstation, we are not allowed to FTP into LPAR A23, as shown in Figure 8-48.

```
C:\>ftp 10.10.1.230
> ftp: connect :Unknown error number
ftp> ls
Not connected.
ftp>
```

Figure 8-48 Workstation FTP fails

FTP from LPAR A24 to LPAR A23

From our workstation, we can log on to z/OS image A24 (SC31), because we have not set up any IP filtering rules for A24 (see Figure 8-6 on page 328). From there, we can demonstrate that the host-to-host FTP works, because we set up our filter rules to permit FTP traffic between LPARs A24 and A23.

Using TSO on LPAR A24, we then issued the following command:

```
ftp -p tcpipa 10.10.1.230
```

Figure 8-49 shows the result of the successful TSO FTP command.

```
 220-FTPMVS1 IBM FTP CS V1R9 at WTSC30A, 17:19:29
 220 Connection will close if idle for more than 5 minutes.
 NAME (10.10.1.230:CS10):
CS10
 >>> USER CS10
 331 Send password please.
 PASSWORD:


 >>> PASS
 230 CS10 is logged on. Working directory is "CS10.".
 Command:
```

Figure 8-49 TSO output

Problem determination

The following documentation can aid in problem determination:

- TCP/IP CTRACE output: The CTRACE facility has flexibility such as filtering, combining multiple concurrent applications and traces, and using an external writer. (For additional information, see *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.)
- PASEARCH command: Use the **pasearch** command to display policy rules with complex conditions.
- TCP/IP profile output
- POLICY config file output

9

IP Security

IP Security (IPSec) is a suite of protocols and standards defined by the Internet Engineering Task Force (IETF) to provide an open architecture for security at the IP networking layer of TCP/IP. IPSec provides the framework to define and implement network security based on policies defined by your organization. This chapter discusses the implementation of IPSec on z/OS.

The following topics are discussed in this chapter.

Section	Topic
9.1, “IPSec description” on page 374	Basic concepts of IPSec and Virtual Private Network (VPN)
9.2, “Basic concepts” on page 375	Key IPSec components
9.3, “How IPSec is implemented” on page 377	Implementation procedures for installing IPSec
9.5, “Configuring IPSec between two z/OS systems” on page 394	How to implement IP security between two z/OS images
9.6, “Additional information” on page 424	References to additional information about IP Security (IPSec)

9.1 IPSec description

As mentioned, IPSec is a suite of protocols and standards defined by the IETF to provide an open architecture for security at the IP networking layer of TCP/IP. It is the most commonly used protocol for implementing a Virtual Private Network (VPN).

Because IPSec works at the IP networking layer, IPSec can be used to provide security for any TCP/IP application without modification. If necessary, applications can have their own additional security features on top of the underlying IPSec security. Also, unlike TCP-layer-based security implementations (such as SSL/TLS), IPSec can be used to protect both TCP and UDP applications. This means that EE (Enterprise Extender) traffic can be protected too.

The IPSec standards have also been structured so that they can accommodate newer, more powerful, algorithms as they become available in the future.

IPSec technology is used to build what is referred to as a Virtual Private Network (VPN), because the technology enables an enterprise to extend its network across an untrusted network (such as the Internet) without compromising security. Using IPSec protocols, each host can encrypt and authenticate individual IP packets between itself and other communicating hosts. Companies can securely and cost effectively extend the reach of their applications and data across the world by replacing leased lines to remote sites with VPN connections. Because Internet access is increasingly available worldwide, companies can now use VPN technologies to reach places where other connectivity alternatives such as leased lines are expensive or unavailable.

z/OS Communications Server implements the following IPSec RFCs:

- RFC 2401: Security Architecture for the Internet Protocol
- RFC 2402: IP Authentication Header
- 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 2406: IP Encapsulating Security Payload (ESP)
- 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
- RFC 2451: The ESP CBC-Mode Cipher Algorithms
- RFC 3602: The AES-CBC Cipher Algorithm And Its Use with IPSec
- RFC 3715: IPSec-Network Address Translation (NAT) Compatibility Requirements
- RFC 3947: Negotiation of NAT-Traversal in the IKE
- RFC 3948: UDP Encapsulation of IPSec ESP Packets

Most of these RFCs have been superseded; in particular, 2401 to 2409 are superseded by the 4301 to 4309 suite of standards. However, many of the changes to IPSec introduced by RFCs 4301 through 4309 are incremental and allow extensive backward compatibility to implementations supporting RFCs 2401 through 2409.

Note: One thing that can be confusing about z/OS Communications Server IPSec support is that it has been packaged together with IP filtering support and is referred to as *integrated IP Security*. That is because there is a very close affinity between IPSec and IP filtering in z/OS Communications Server; although you can implement IP filtering without IPSec, you cannot implement IPSec without IP filtering (discussed in Chapter 8, “IP filtering” on page 319).

9.2 Basic concepts

The IPSec architecture provides a framework for security at the IP layer of IPv4 and IPv6. Because IPSec protocols perform at this layer, transport protocols and applications can be protected without being modified.

IPSec defines a unidirectional logical connection between two endpoints. The concept of a Security Association (SA) is fundamental to IPSec, defining the security characteristics of the traffic that is carried across the tunnel. The span of protection of an SA can vary. For example, the SA can protect traffic for multiple connections (all traffic between networks), or the SA can protect traffic for a single connection.

IPSec can provide a secured connection and an encrypted payload with its implementation. The authentication proves data origin authentication, data integrity, and replay protection, which are explained as follows:

- Data origin authentication confirms that the data origin was from a device that knows the correct cryptographic key.
- Data integrity proves that the contents of a datagram have not been changed since the authentication data was created.
- Replay protection prevents an attacker from sending bogus IPSec packets resulting in unnecessary cryptographic operations. For example, if an attacker kept retransmitting the ESP last packet sent, replay protection will prevent that packet from being decrypted and authenticated each time. The sequence number in the IP header is always in clear text.

The Authenticated Header (AH) protocol is the IPSec-related protocol that provides authentication. The Encapsulated Security Payload (ESP) protocol provides data encryption, which conceals the content of the payload. ESP also offers authentication. Internet Key Exchange (IKE) protocol exchanges the secret number that is used for encryption or decryption in the encryption protocol.

AH and ESP support two mode types: transport mode and tunnel mode, as shown in Figure 9-1. These modes tell IP how to construct the IPSec packet. Transport mode is used when both endpoints of the tunnel are hosts (data endpoints). Tunnel mode is used whenever either endpoint of the tunnel is a router or firewall.

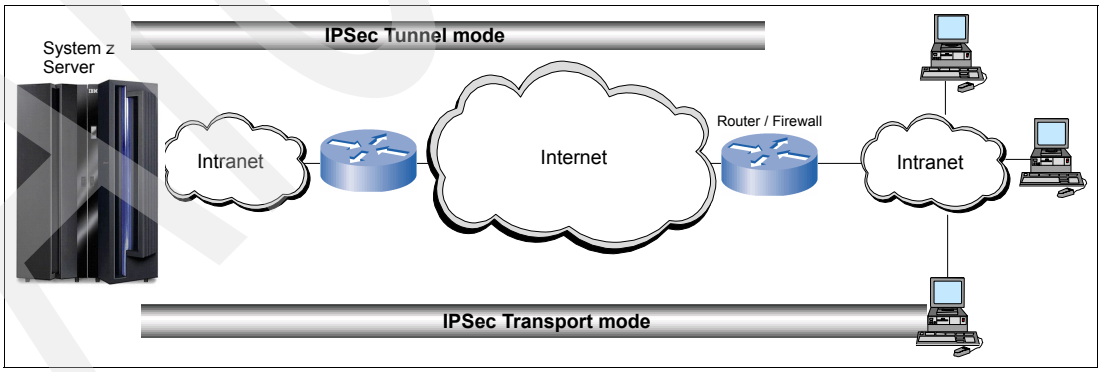


Figure 9-1 Two mode types: transport mode and tunnel mode

With transport mode, the IP header of the original transmitted packet remains unchanged. With tunnel mode, a new IP header is constructed and placed in front of the original packet.

9.2.1 Key components

Three of the most important IPSec components implemented by z/OS Communications Server include:

- RFC 2402: IP Authentication Header (AH) protocol, which provides for data authentication, IP header authentication, and data origin authentication
- RFC 2406: IP Encapsulating Security Payload (ESP), which provides for data authentication, data origin authentication, and data privacy (encryption)
- RFC 2409: The Internet Key Exchange (IKE), which provides protocols for automated encryption key management

In the following sections, we describe these components in more detail.

9.2.2 IP Authentication Header protocol

As the name suggests, IPSec Authentication Header (AH) authenticates IP packets, ensuring that they came from a legitimate origin host and that they have not been changed. IPSec AH provides:

- Data integrity by authenticating the *entire* IP packet using a message digest that is generated by algorithms such as HMAC-MD5 or HMAC-SHA
- Data origin authentication by using a shared secret key to create the message digest
- Replay protection by using a sequence number field within the AH header

9.2.3 IP Encapsulating Security Payload protocol

Encapsulating Security Payload (ESP) provides additional protection beyond (or in addition to) AH, including:

- Encapsulating and encrypting the IP packet
- Authenticating the IP datagram portion of the IP packet, including most of what is listed under AH: data integrity for all but the IP header; data origin authentication; and replay protection. For most users, the authentication protection provided by ESP should be sufficient, and AH should not be necessary if ESP is already being used for encryption.

In ESP, before leaving a host, outbound packets are rebuilt with additional IPSec headers using a cryptographic key that is known to both communicating hosts. This is called *encapsulation*.

On the receiving side, the inbound packets are stripped of their IPSec headers (decapsulated) using the same cryptographic key, thereby recovering the original packet. Any packet that is intercepted on the IP network is unreadable to anyone without the encryption key. Any modifications to the IP packet while in transit are detected by authentication processing at the receiving host and is discarded.

9.2.4 Internet Key Exchange protocol

The Internet Key Exchange (IKE) protocol (which is implemented in z/OS Communications Server by the IKE daemon) manages the transfer and periodic changing of security keys between senders and receivers. Key exchange, defined in IKE, is normally a two-step process:

1. The partners establish a secure logical connection, called a Security Association (SA), and decide on security parameters such as encryption and hashing algorithms and

authentication methods (IKE SA). This is sometimes referred to as the *phase 1* tunnel or IKE tunnel.

2. After the appropriate security parameters have been negotiated, they set up a second Security Association for the actual data transfer (IPSec SA). This is sometimes referred to as the *phase 2* tunnel or IPSec tunnel.

Such secure logical connections between pairs of endpoints are often called *tunnels*. z/OS Communications Server IPSec implementation refers to two types of tunnels:

- Manual IPSec tunnels: The security parameters and encryption keys are statically configured and are manually managed by a security administrator. Manual tunnels are not commonly implemented.
- Dynamic IPSec tunnels: The security parameters are negotiated and the encryption keys are generated dynamically using IKE.

The IKE daemon uses the IP security policies defined by you in the Policy Agent (PAGENT) and dynamically manages the keys that are associated with dynamic IPSec VPNs.

9.3 How IPSec is implemented

IPSec uses the services of a number of z/OS Communications Server components (illustrated in Figure 9-2) to provide policy-based security.

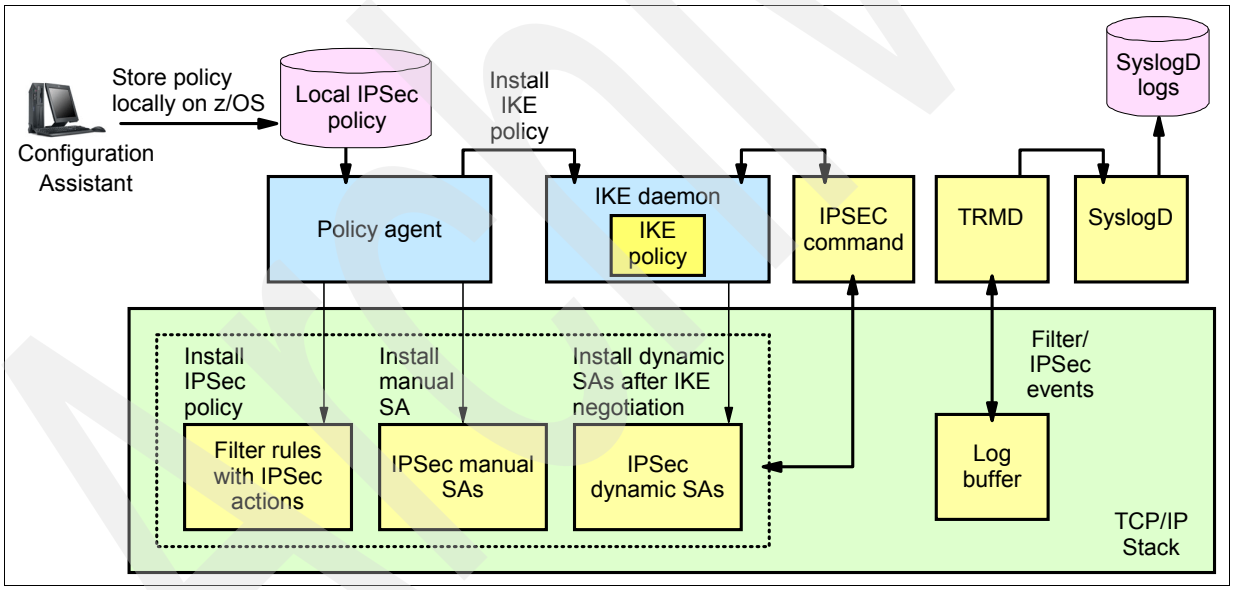


Figure 9-2 z/OS Communications Server components involved with IPSec

Figure 9-2 shows a local IPSec policy repository. In fact, it is possible to provide a centralized IPSec policy repository under the control of Centralized Policy Server or Distributed Policy Server. This subject is described in Chapter 6, “Central Policy Server” on page 257.

The diagram also shows an IKE daemon, which is responsible for dynamic Internet Key Exchange (IKE) protocols. The IKE daemon can be configured to act as a client for Network Security Services (NSS) on behalf of multiple TCP/IP stacks. Multiple TCP/IP stacks can establish connections with the Network Security Services Daemon (NSSD) to obtain certificate management and network management interface services from the server. This topic is discussed in detail in Chapter 4, “Network Security Services” on page 85.

In the following sections we briefly describe the steps required to implement IPSec:

1. In the following sections, we explain these steps in more detail..
2. Setting up the Traffic Regulation Management Daemon.
3. Updating the TCP/IP stack to activate IPSec.
4. Restricting the use of the **ipsec** command.
5. Installing the IBM Configuration Assistant for z/OS.
6. Defining the IPSec policies to PAGENT.
7. Setting up the IKE daemon.
8. Setting up the system logging daemon to log IKED messages.
9. Starting the IKE daemon and verifying it initializes.

In the following sections, we explain these steps in more detail.

9.3.1 Installing the Policy Agent

PAGENT reads the configuration files that contain the IP security policy configuration statements, checks them for errors, and installs them into the IKE daemon and the TCP/IP stack. Setting up the PAGENT is described in Chapter 5, “Policy Agent” on page 221.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an Authorized Program Facility (APF)-authorized library.

After setting it up, you need to define the IpSecConfig statement to specify the path of the policy file that contains stack-specific IPSec policy statements to PAGENT. Example 9-1 shows the IP security statements in the TCP/IP stack configuration file used by Policy Agent.

Example 9-1 The pagent /SC32/etc/pagent32_TCPIPA.conf file with IP security configured

```
######################################################################
#                IMAGE FILE FOR TCPIPA on LPAR SC32
######################################################################
IPSecConfig  /etc/pagent32_TCPIPA_IPSec.conf
```

9.3.2 Setting up the Traffic Regulation Management Daemon

The Traffic Regulation Management Daemon (TRMD) is responsible for logging IP security events that are detected by the stack, including IP filter events, updates to the IP security policy, and the creation, deletion, and refresh of IPSec security associations.

For a detailed example of implementing TRMD, see 5.4, “Setting up Traffic Regulation Management Daemon” on page 254.

9.3.3 Updating the TCP/IP stack to activate IPSec

To activate IPSec, you need to add the IPSECURITY and SOURCEVIPA options in the IPCONFIG statement in the TCP/IP Profile. In addition, you need to add IPSec rules, which are also added to the stack’s profile. See Example 9-2 on page 379.

Important: Make certain that you build some IPSEC rules at the same time as you add the IPSECURITY keyword. You need those rules to allow connectivity to your stack prior to pagent loading its filter rules.

You may also need these rules as a fail-safe. You can force the TCP/IP stack to ignore policy agent rules using the **ipsec -f default** command. The TCP/IP stack will revert to the rules found on your IPSEC statements. This might be needed if you have a network security problem, or if you inadvertently load bad policies into the policy agent.

Example 9-2 Our definitions for IPSEC in TCP/IP profile of stack TCPIPA in image SC32

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING
IPSECURITY  SOURCEVIPA
DYNAMICXCF 10.1.7.51 255.255.255.0 1
;
NETMONITOR SMFSERVICE
;
; Added IPSEC statement
;
IPSEC  LOGENABLE
;  ;;  OSPF protocol used by Omproute
  IPSECRULE *  *  NOLOG  PROTOCOL OSPF
;  ;;  IGMP protocol used by Omproute
  IPSECRULE *  *  NOLOG  PROTOCOL 2
;  ;;  DNS queries to UDP port 53
 IPSECRULE *  *  NOLOG  PROTOCOL UDP      SRCPORT *  DESTPORT 53  SECCL ASS 100
;  ;;  Administrative access
 IPSECRULE *  10.1.1.10 LOG  PROTOCOL *
ENDIPSEC
```

9.3.4 Restricting the use of the ipsec command

The **ipsec** command is very powerful and needs to be protected from unauthorized use. We created a RACF profile for this and gave command access to the IKE daemon. Example 9-3 shows the commands to do this.

Example 9-3 Define access control for the ipsec command

```
SETROPTS GENERIC(SERVAUTH) RDEFINE SERVAUTH EZB.IPSECCMD.* UACC(NONE)
PERMIT EZB.IPSECCMD.* CLASS(SERVAUTH) ID(IKED) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

9.3.5 Installing the IBM Configuration Assistant for z/OS

IBM provides a graphical user interface (GUI) called IBM Configuration Assistant for z/OS Communication Server to help you to code your security policies. This is a Windows-based interface that you can download from the IBM Web site:

http://www.ibm.com/software/network/commserver/zos/support/

When your policy has been coded using this software, it can then be sent via File Transfer Protocol (FTP) to your z/OS system to be used by the Policy Agent.

Important: The IBM Configuration Assistant for z/OS Communication Server is updated continuously with enhancements. Therefore, you should download the latest version before you use it.

You can also contact the ibm.software.commserver.os390.ip newsgroup about problems and comments, at:

news://news.software.ibm.com/ibm.software.commserver.os390.ip

We used IBM Configuration Assistant for z/OS Communication Server to implement two scenarios using dynamic IPSec tunnels, as follows:

- Configuring IPSec between two z/OS systems, where we executed these steps:
  - Setting up the policy
  - Installing the configuration files
  - Verifying IPSEC between two z/OS images
- Appendix C, “Configuring IPSec between z/OS and Windows” on page 711, where we executed these steps:
  - Set up the IKE daemon
  - Set up the certificates
  - Set up the z/OS IPSec policy
  - Set up a Windows IPSec policy
  - Verify that things are working

9.3.6 Defining the IPSec policies to PAGENT

IPSec provides flexible building blocks that can support a variety of configurations. You can choose from a number of protocols and encryption algorithms provided by IPSec to suit to the security requirements of your installation. You can define your IPSec security policies to PAGENT in one of two ways:

- Manually code all of the required policy statements to create a configuration file in a z/OS UNIX file or an MVS data set.
- Use the IBM Configuration Assistant for z/OS Communication Server to create the IP security configuration file.

To implement these security requirements, the IBM Configuration Assistant for z/OS Communication Server provides by default three predefined security level objects, which are used for the Phase2 negotiation. They are:

- IPSEC_Gold
- IPSEC_Silver
- IPSEC_Bronze

We can also create our own security levels to meet our business requirements using IBM Configuration Assistant for z/OS Communication Server. IPSec on z/OS Communications Server supports PFS (Perfect Forward Secrecy) to generate keys on phase 2 (IPSec tunnel) negotiation and can use it to initiate or to respond to phase 2 negotiation requests.

Important: PFS is important because of the following security and performance implications:

Long-running tunnels have phase 1 (IKE tunnel) and phase 2 (IPSec tunnel) “refresh” values. The phase 1 refreshes are the most costly in terms of resource consumption (CP), because the master key phase 1 must reestablish from scratch, followed by a phase 2 key rebuild as well. Phase 2 refreshes, if no PFS is configured, will use earlier keys as a starting point when refreshing the phase 2 key. This implies that if a hacker were able to compromise an earlier key, the presently-used key could more easily be determined.

By specifying any form of PFS, you can force a phase 2 key to be rebuilt from scratch each time it is scheduled to be refreshed. In this case, a compromised earlier key is of no benefit. This does mean that the processor (or crypto card) will be required to do a little more work during a phase 2 refresh than if no PFS was in use.

The usage of PFS is optional in a phase 2 negotiation. To be able to work with different clients, z/OS Communications Server accepts multiple PFS values. Some clients may only be able to support lower PFS groups. Other clients may support higher PFS groups for higher security.

In our case, before we started to define our scenarios we used IBM Configuration Assistant for z/OS Communication Server to create a specific security level called PFS which implement s PFS values allowing multiple security levels. The steps to create PFS are shown next.

Creating a security level for PFS

To create a security level for PFS, start IBM Configuration Assistant for z/OS Communication Server.

1. In the Main Perspective panel, select the IPSec technology and click **Configure**, as shown in Figure 9-3 on page 382.

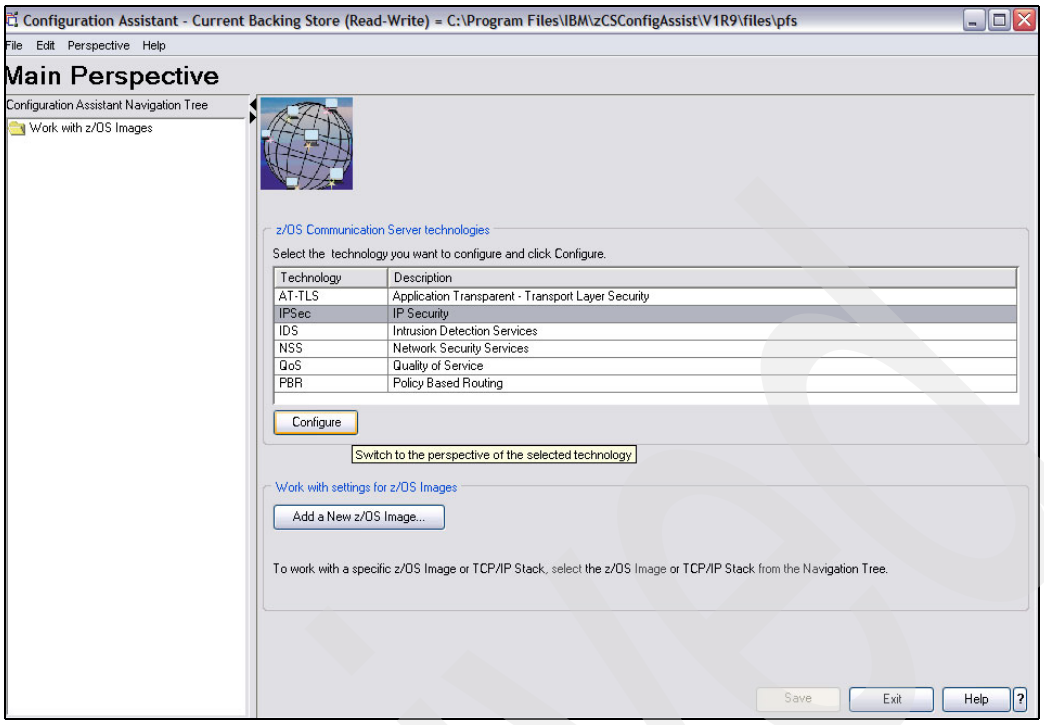


Figure 9-3 Ibm Configuration Assistant Main Perspective panel

2. In the IPSec Perspective main panel, click the **Security level** button, as shown in Figure 9-4.

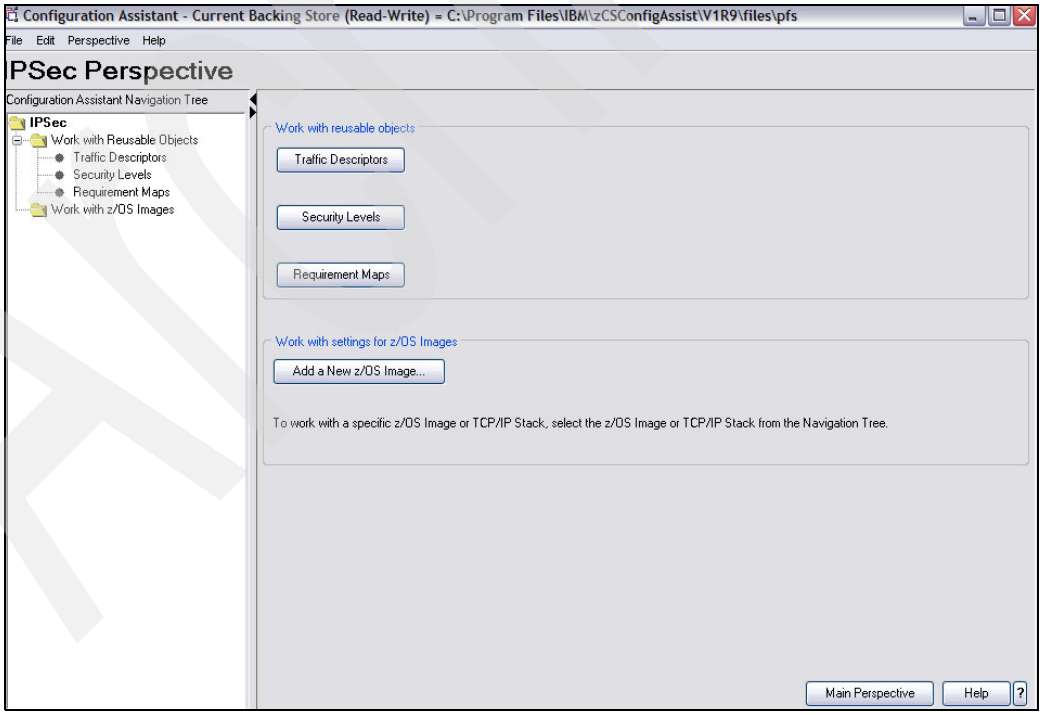


Figure 9-4 IPSec Perspective panel

3. In the Configuration Assistant - Security Levels panel, click **Add**, as shown in Figure 9-5 on page 383.

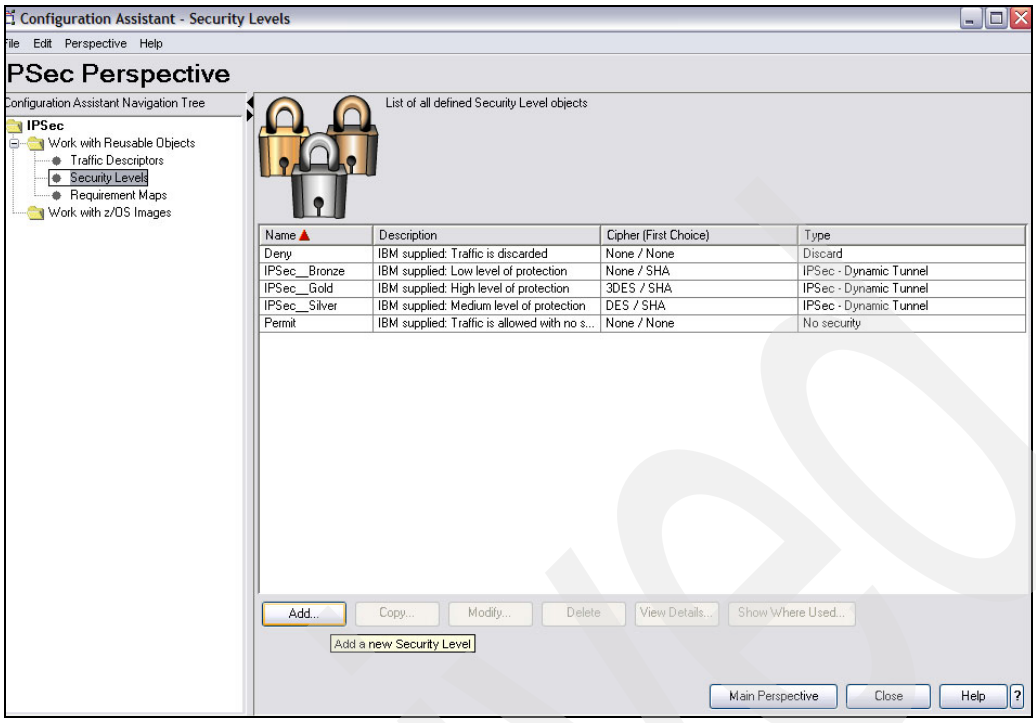


Figure 9-5 Configuration Assistant - Security Levels panel

4. In the New Security Level: Names and Type panel, enter the name our new security level object PFS and description. Select **IPSec Dynamic Tunnel** and click **Next**, as shown in Figure 9-6.

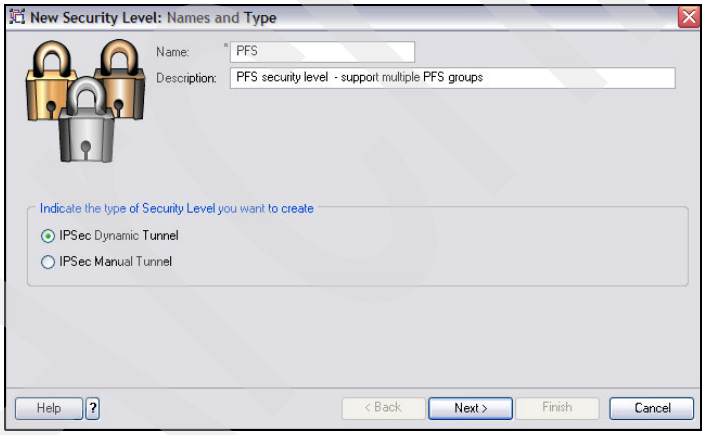


Figure 9-6 New Security Level: Names and Type panel

5. In the New Security Level: Cipher Selections panel, select **Triple DES** as the encryption algorithm and **HMAC SHA** as the authentication algorithm, as shown in Figure 9-7 on page 384. Click **Next**.

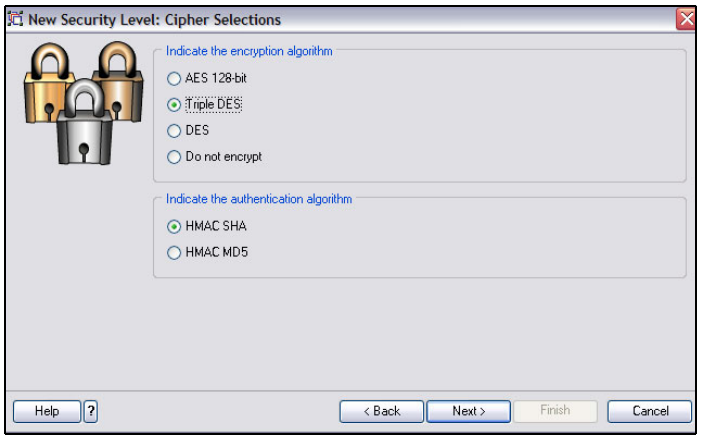


Figure 9-7 New Security Level: Cipher Selections

6. In the New Security Level: Additional Settings panel, click **Advanced Settings**.
7. In the Advanced Dynamic Tunnel Settings panel, select the Additional Settings tab and select **Diffie-Hellman Group 2** in the `Initiator Perfect Forward Secrecy (PFS) Level` field, and in the `Acceptable Perfect Forward Secrecy Levels` field, select **None** and **Diffie-Hellman Group 1, 2** and **5,** as shown in Figure 9-8. Click **OK** and then click **Finish**.

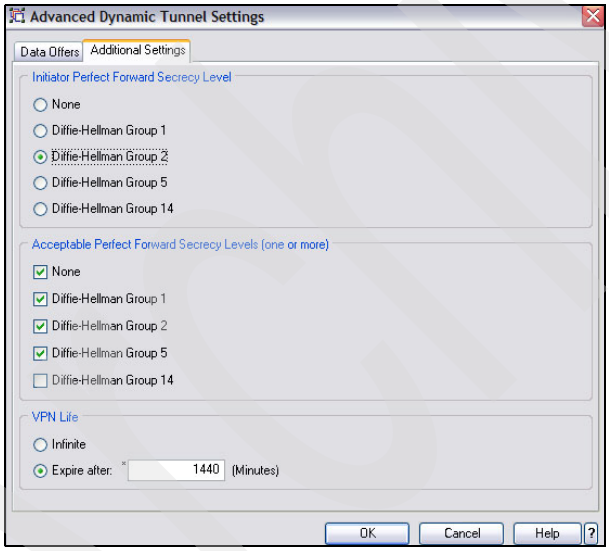


Figure 9-8 Advanced Dynamic Tunnel Settings panel

The `List of all defined Security Level objects` field should now include the `PFS` object, as shown in Figure 9-9.

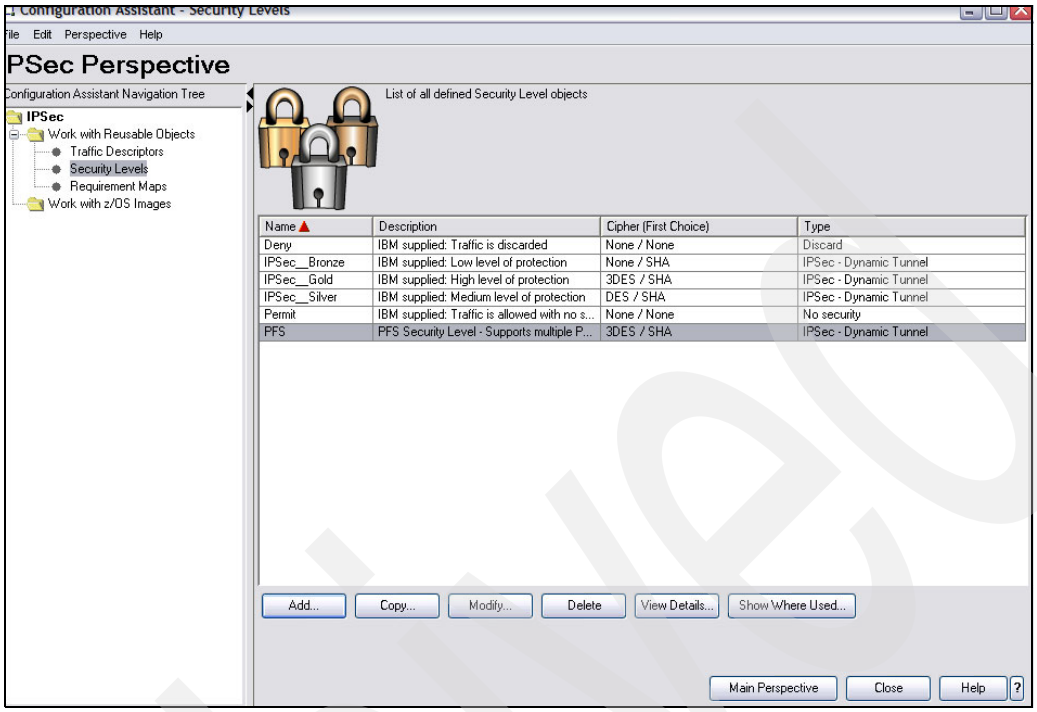


Figure 9-9 Security Levels panel with PFS security level added

9.3.7 Setting up the IKE daemon

The Internet Key Exchange daemon (IKED) is responsible for retrieving the IP security policy from the Policy Agent, and dynamically managing keys that are associated with dynamic tunnels. The IKE daemon implements the protocols to dynamically establish IKE SAs with peers that also support these protocols. It can provide automatic management of cryptographic keys and remove the administrative burden associated with key creation, distribution, and maintenance.

IKE provides the following services:

- Host authentication (ensuring that each host is certain of the other's identity)
- The negotiation of a security association as follows:
  - Agreeing on the type of traffic to be protected
  - Agreeing on the authentication and encryption algorithms to be used
  - Generating cryptographic keys
- Nondisruptive periodic refresh of keys
- The deletion of security associations whose lifetimes have expired

IKE operates at the application layer and communicates between two IKE peers using a series of UDP messages.

Only one instance of the IKE daemon can run on a single z/OS image. The IKE daemon obtains operational parameters from the configuration file and the IP security policy from the Policy Agent. These configuration parameters can be stack-specific, allowing a single IKE daemon to provide the appropriate services to each stack as needed.

Important: When the IKE daemon has obtained the IP Security policy, the Policy Agent may be stopped without impacting the IKE daemon. However, any changes to the IP Security policy are not detected until the Policy Agent is restarted. The IKE daemon reconnects to the Policy Agent when it is restarted.

The steps to set up the IKE daemon are:

1. Setting up the IKE daemon cataloged procedure.
2. Creating the IKE daemon configuration file.
3. Reserving the IP ports for the IKE daemon.
4. Associating a RACF user ID and group with the IKE daemon.
5. Defining profiles to control access to the RACDCERT command.
6. Creating a RACF key ring.
7. Installing an X.509 digital certificate for the IKE daemon.
8. (Optionally) Authorizing use of hardware cryptographic encryption.

These steps are explained in the following sections. Steps 6 and 7 are optional if you are not using RSA signature.

Setting up the IKE daemon cataloged procedure

A sample of the procedure can be obtained from the z/OS Communications Server installation file TCPIP.SEZAINST(IKED). Example 9-4 shows the procedure we used. Copy this procedure into your SYS1.PROCLIB library.

Example 9-4 IKE daemon cataloged procedure

```
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD DUMMY
//STDENV   DD DSN=TCPIP.IKED.ENV(IKED),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

TCPIP.IKED.ENV(IKED) contains the following:

```
IKED_FILE=/etc/security/iked.conf
IKED_CTRACE_MEMBER=CTIIKE00
```

The file /etc/security/iked.conf is the IKE daemon configuration file shown in Example 9-5.

IKED_CTRACE_MEMBER is the name of a parmlib member that contains default CTRACE settings for IKE daemon.

Creating the IKE daemon configuration file

A sample configuration is supplied in the z/OS Communications Server installation file /usr/lpp/tcpip/samples/IBM/EZAIKCFG. The configuration we used is shown in Example 9-5.

Example 9-5 IKE daemon configuration file

```
IkeConfig
{
 IkeSyslogLevel    255
 PagentSyslogLevel 255
 Keyring           IKED/IKED_keyring
 KeyRetries        10
 KeyWait           30
 DataRetries       10
```

```
 DataWait          15
 Echo              no
 PagentWait        0
}
```

These parameters control the workings of the IKE daemon, and each is explained as follows:

- IkeSyslogLevel: Level of logging from the IKE daemon. We left it at the highest level to get all messages during testing. On the production system, this can be set to 1.
- PagentSyslogLevel: Level of logging from pagent. We set it to the highest level of 255 for testing.
- Keyring: Owning user ID and ringname for RSA Signature Mode of authentication. We demonstrate how to set this up in “Creating a RACF key ring” on page 388.
- KeyRetries: Number of times the IKE daemon retransmits a key negotiation before it stops retrying.
- KeyWait: Number of seconds between retransmissions of key negotiations.
- DataRetries: Number of times the IKE daemon retransmits a data negotiation before it stops retrying.
- DataWait: Number of seconds between retransmissions of data negotiations.
- Echo: Option to echo all IKE daemon log messages to the IKEDOUT DD file.
- PagentWait: The time limit in seconds to wait for connection to the Policy Agent. A value of zero (0) means retry forever.

Note: In our case, IkeSyslogLevel and PagentSyslogLevel were set for the maximum level of tracing for our testing. In the production environment, however, you should set them to low levels to avoid a performance impact from excessive logging.

We edited our IKE daemon configuration file as follows:

1. We issued the **su** command in UNIX to get superuser authority.
2. We copied the file /usr/lpp/tcpip/samples/IBM/EZAIKCFG to /etc/security/iked.conf using the UNIX command:

   ```
   cp /usr/lpp/tcpip/samples/IBM/EZAIKCFG /etc/security/iked.conf
   ```

3. We updated this file using the UNIX command:

   ```
   oedit /etc/security/iked.conf
   ```

Tip: Newly created z/OS UNIX files might not have write access. Issue the **ls -al** command for the file to verify this and issue the **chmod 700** command to change the access, if necessary.

Reserving the IP ports for the IKE daemon

Update the PORT statement in PROFILE.TCPIP to reserve UDP ports 500 and 4500 for the IKE daemon:

```
PORT
500  UDP IKED
4500 UDP IKED
```

Make sure to specify the correct name of the IKE daemon that you are using here. (We used the name IKED.)

Associating a RACF user ID and group with the IKE daemon

We defined a user ID IKED with default group TCPGRP and with an OMVS segment. This user ID needs to be defined with UID=0. A home directory was also assigned to this user ID.

We then defined the started task IKED to RACF and associated the user IKED and group TCPGRP using the RDEFINE command. We refreshed the RACLIST and GENERIC for the STARTED class to update the profiles in storage with this new information.

Example 9-6 shows the commands we used. Note that IKE daemon user ID IKED requires read access to RACF profile BPX.DAEMON in the FACILITY resource class to work as a daemon.

Example 9-6 Associate a RACF user ID and group with IKE daemon

```
ADDUSER IKED DFLTGRP(TCPGRP) OMVS(UID(0) SHARED HOME('/'))
RDEFINE STARTED IKED.* STDATA(USER(IKED) GROUP(TCPGRP))
PERMIT BPX.DAEMON CLASS(FACILITY) ID(IKED) ACCESS(READ)
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED) REFRESH
```

Important: If you are a network programmer, you might not have the necessary RACF authority to issue many of these commands. You might need to work with your RACF administrator who will have the necessary "SPECIAL" authority to issue them.

Defining profiles to control access to the RACDCERT command

The RACDCERT command is used to generate keys and key rings and to connect the keys to key rings. This facility should be protected and only authorized users such as the IKE daemon should have access to it. Example 9-7 shows the commands we used.

Example 9-7 Control access to the RACDCERT command

```
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.LISTRING UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(IKED) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IKED) ACC(UPDATE)
```

Creating a RACF key ring

Digital certificates are made available to the IKE server by connecting them to a key ring that is owned by the IKE server. To create a key ring for the IKE server, issue the TSO command:

```
RACDCERT ID(IKED) ADDRING(IKED_keyring)
```

Note: The value used for key ring name is case-sensitive.

Installing an X.509 digital certificate for the IKE daemon

You can install an X.509 digital certificate in the following ways:

- Generate an X.509 digital certificate for the IKE server and have it signed by a certificate authority.
- Generate a self-signed X.509 digital certificate for the IKE server.
- Migrate an existing key database to a RACF key ring.

We generated a self-signed X.509 digital certificate by following these steps:

1. Activating RACF classes DIGTCERT and DIGTNMAP if not already active.
2. Generating a self-signed certificate to represent the local certificate authority.
3. Creating a certificate for the server.
4. Connecting the certificates to IKED's key ring.
5. Telling the IKE daemon where to find the key ring.
6. Verifying certificate creation.

These steps are explained in the following sections.

Activating RACF classes DIGTCERT and DIGTNMAP if not already active

The DIGTCERT (contains digital certificates and information related to them) and DIGTNMAP (mapping class for certificate name filters) classes should be active for RACF certificate creation. The command to do this is:

```
SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)
```

Generating a self-signed certificate to represent the local certificate authority

We created a certificate to act as local certificate-issuing (signer) authority. The label for our certificate was My Local Certificate Authority. This label will be used to refer to the certificate in the following steps. Example 9-8 shows the command to do this.

Example 9-8 Generate a self-signed certificate

```
RACDCERT ID(IKED) CERTAUTH GENCERT SUBJECTSDN( O('I.B.M Corporation')
-
          CN('itso.ibm.com')                          -
          C('US'))                                    -
          WITHLABEL('My Local Certificate Authority') -
          KEYUSAGE(certsign)
```

Creating a certificate for the server

We created a certificate for the IKED daemon and signed the new certificate with authority of My Local Certificate Authority, which was created to represent the local certificate authority. Example 9-9 shows the command we used.

Example 9-9 Create a certificate for the server

```
RACDCERT ID(IKED) GENCERT
        SUBJECTSDN  (CN('IKE Daemon on SC32')
                     OU('ITSO')
                     C('US'))
                     WITHLABEL('IKE Daemon on SC32')
                     SIGNWITH(CERTAUTH
                     label('My Local Certificate Authority'))
```

Connecting the certificates to IKED's key ring

The certificates created in the previous two steps need to be connected to IKED's key ring. The commands shown in Example 9-10 accomplish this.

Example 9-10 Connect the certificate to IKED's existing key ring

```
RACDCERT ID(IKED) CONNECT(ID(IKED)                        -
                  LABEL('IKE Daemon on SC32')             -
                  RING(IKED_keyring)                      -
                  USAGE(personal))
RACDCERT ID(IKED) CONNECT(ID(IKED) CERTAUTH               -
                  LABEL('My Local Certificate Authority') -
                  RING(IKED_keyring)                      -
                  USAGE(certauth))
```

Telling the IKE daemon where to find the key ring

We then added the following statement to the IKE daemon configuration file, etc/security/iked.conf, which we defined earlier:

```
Keyring IKED_keyring
```

Verifying certificate creation

You can verify that the certificates that you have created are connected to the key ring associated with user ID IKED by using the RACDCERT command and examining the output of the Ring Associations field. Example 9-11 shows the commands to do the verification.

Example 9-11 Verify certificate creation

```
RACDCERT ID(iked) LIST(LABEL('IKE Daemon on SC32'))
RACDCERT ID(IKED) CERTAUTH   -
                  LIST(LABEL('My Local Certificate Authority'))
RACDCERT id(IKED) LISTRING(IKED_keyring)
```

Example 9-12 shows the output of these commands.

Example 9-12 Verify certificate creation

```
Label: IKE Daemon on SC32
Certificate ID: 2QTJOsXEydLFQMSBhZSWlUCWlUDiw/Py
Status: TRUST
Start Date: 2007/09/21 00:00:00
End Date:   2008/09/21 23:59:59
Serial Number:
     >01<
Issuer's Name:
     >CN=ITSO z/OS CS.O=I.B.M Corporation.C=US<
Subject's Name:
     >CN=IKE Daemon on SC32.OU=ITSO.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
  Ring Owner: IKED
  Ring:
     >IKED_keyring<
READY
RACDCERT ID(IKED) CERTAUTH LIST(LABEL('My Local Certificate Authority'))
Digital certificate information for CERTAUTH:
Label: My Local Certificate Authority
  Certificate ID: 2QiJmZmDhZmjgdSoQNOWg4GTQMOFmaOJhomDgaOFQMGko4iWmYmjqEBA
```

```
  Status: TRUST
  Start Date: 2007/09/21 00:00:00
  End Date:   2008/09/21 23:59:59
  Serial Number:
       >00<
  Issuer's Name:
      >CN=ITSO z/OS CS.O=I.B.M Corporation.C=US<
  Subject's Name:
      >CN=ITSO z/OS CS.O=I.B.M Corporation.C=US<
  Key Usage: CERTSIGN
  Private Key Type: Non-ICSF
  Private Key Size: 1024
  Private Key Size: 1024
  Ring Associations:
    Ring Owner: IKED
    Ring:
       >IKED_keyring<
READY
  RACDCERT id(IKED) LISTRING(IKED_keyring)
Digital ring information for user IKED:
Ring:
       >IKED_keyring<
  Certificate Label Name            Cert Owner     USAGE      DEFAULT
  -------------------------------   ------------   --------   -------
  IKE Daemon on SC32                ID(IKED)       PERSONAL     NO
  My Local Certificate Authority    CERTAUTH       CERTAUTH     NO
```

Authorizing use of hardware cryptographic encryption

This step is optional and is required only if you are going to use the IBM System z hardware cryptographic feature to encrypt or decrypt TCP/IP packets and digital signatures.

To authorize the use of this feature, define the appropriate profiles in the CSFSERV class and give access to authorized users and daemons. The commands required are shown in Example 9-13.

Example 9-13 Authorize use of hardware cryptographic encryption

```
RDEFINE CSFSERV service-name UACC(NONE)
PERMIT service-name CLASS(CSFSERV) ID(stackname) ACCESS(READ)
PERMIT service-name CLASS(CSFSERV) ID (userid)
SETROPTS CLASSACT(CSFSERV) SETROPTS RACLIST(CSFSERV) REFRESH
```

In our setup, we did not use this feature.

Additional information

For more information about RACF, refer to the following manuals:

- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683, for use of RACDCERT command
- *z/OS Security Server RACF Command Language Reference*, SA22-7687, for other RACF commands

For more information about System z hardware cryptography, refer to the following manuals:

- *z/OS Cryptographic Services ICSF Overview*, SA22-7519
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521

9.3.8 Setting up the system logging daemon to log IKED messages

The system logging daemon (syslogd) manages the logging of messages and events for all of the other components, including where the log messages are written. We added the following line in our SYSLOGD configuration file /etc/syslogd.conf to route all IKED daemon logs:

```
*.IKED*.*.* /tmp/iked-sc32.log
```

9.3.9 Starting the IKE daemon and verifying it initializes

Start IKED and make sure it comes up correctly. Example 9-14 shows the startup messages of IKED.

Example 9-14 Starting IKED

```
S IKEDC
$HASP373 IKEDC    STARTED
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R9 SERVICE LEVEL CS070402 CREATED ON Apr 3, 2007
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/security/iked32.conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1133I IKE STATUS FOR STACK TCPIPA   IS ACTIVE WITHOUT IPSECURITY
SUPPORT
EZD1046I IKE INITIALIZATION COMPLETE
```

9.3.10 AES cryptographic support for integrated IPSec/VPN

z/OS Communications Server supports the Advanced Encryption Standard (AES) algorithm for IP security with a 128-bit key length. The IETF IPSec Working Group intends for AES to eventually be adopted as the default IPSec Encapsulating Security Payload (ESP) cipher; therefore, AES must be included in compliant IPSec implementations.

To configure AES, a Security Level object can be created following the steps described in “Creating a security level for PFS” on page 381 and in step 5, in the New Security Level: Cipher Selections panel, select **AES** as the encryption algorithm, as shown in Figure 9-10 on page 393.

Important: AES encryption software is subject to export restrictions and might not be available in your country.

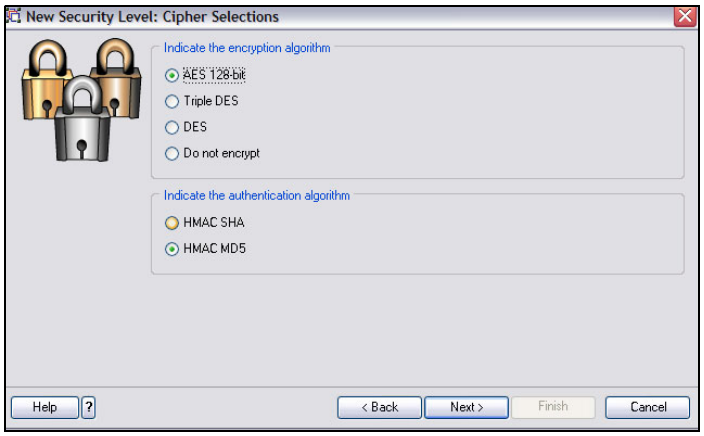


Figure 9-10 Implementing AES security level

Note: The TCP/IP stack requires ICSF to perform all AES encryption. If ICSF is not currently installed on your z/OS image, follow the steps for installation and initialization in *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521.

9.3.11 Commands used to administer IP security

The commands used to administer IP security are:

- **ipsec**: Use this z/OS UNIX System Services command to display information about active filters and security associations, and to control aspects of security association negotiation. Authority to use this command is controlled through the z/OS security server (RACF).
- **pasearch**: Use this command to display Policy Agent information that is defined in the Policy Agent configuration files, including IP security and other types of policies. If the user is not a superuser, authority is controlled though RACF.
- **MODIFY**: Use this console command to make IKE daemon reread the IKED configuration file or to make Policy Agent reread the policy configuration agent files.
- **Netstat**: Use this command to display IPSECURITY status for a particular stack or to display the SecurityClass (SECCLASS) for a specific interface.

For more detailed information about the syntax and usage of those commands, refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

9.4 zIIP Assisted IPSec function

IBM System z9® Integrated Information Processor (IBM zIIP) is a specialty engine, available on the System z9 Enterprise Class (EC) and System z9 Business Class (BC) servers.

The zIIP's execution environment accepts eligible work from z/OS, which manages and directs the work between the general purpose CPU and the zIIP.

The zIIP Assisted IPSec function allows Communication Server to interact with z/OS Workload Manager to have its enclave Service Request Block (SRB) work dispatched to a zIIP. In Communications Server, processing related to security routines, such as encryption and authentication algorithms (AH and ESP), run in enclave SRBs.

zIIP is designed to help free up general computing capacity and lower the overall total cost of computing. Therefore, by using the zIIP Assisted IPSec function, you may be able to achieve significant reduction in general purpose CPU consumption.

For information about configuring zIIP Assisted IPSec, refer to Appendix D, “zIIP Assisted IPSec” on page 741.

9.5 Configuring IPSec between two z/OS systems

In this scenario, we show how to set up a VPN tunnel between two z/OS systems; see Figure 9-11.

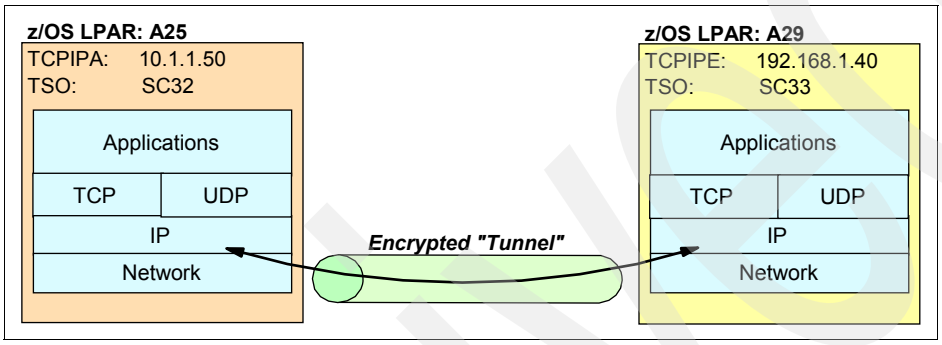


Figure 9-11 VPN traffic between two z/OS systems

We used the IBM Configuration Assistant for z/OS Communication Server to set up a dynamic tunnel between the two z/OS systems. To implement our scenario we used z/OS images SC32 on LPAR A25 and SC33 on LPAR A29.

For each image we configured one TCP/IP stack. For each TCP/IP stack we defined three connectivity rules: one for omproute, one for DNS, and one for all the other traffic. In this section, we demonstrate the step-by-step process of defining the policy to set up this tunnel.

This configuration creates a tunnel with the following characteristics:

- The OSPF traffic is permitted.
- All IP packets between stacks TCPIPA on SC32 and our client, TCPIPE on SC33, will be encrypted.
- The dynamic tunnel is activated by the outbound traffic flow without user intervention.
- The tunnel uses *transport mode* encapsulation (therefore, it does not encapsulate the original IP header as would be done if using *tunnel mode*).
- The tunnel uses shared key authentication for IKE peers.
- The tunnel uses AES (or DES) encryption for both phase 1 (IKE) and phase 2 (IPSec) tunnels.
- The tunnel uses ESP HMAC MD5 authentication

9.5.1 Setting up the policy

To set up the policy using IBM Configuration Assistant for z/OS Communication Server, we performed these steps:

1. Adding Requirement Map objects - we created the Requirement Map objects named Omproute and All_Other_Traffic. These objects will have common policies that can be used for all z/OS images.
2. Adding a new z/OS image
3. Adding a TCP/IP stack for the new z/OS image

4. Adding connectivity rules for the TCP/IP stack
5. Adding another new z/OS image

Adding Requirement Map objects

A Requirement Map is a set of mappings of Traffic Descriptors to Security Levels, where we implement the level of security we want provide for any given type of traffic. An example would be to create a Map named TN3270_Tunnel that uses IPSec only for TN3270E traffic and deny all other protocols. Later, when we create Connectivity Rules for a TCP/IP Stack, we can specify the Local and Remote data endpoints, and select the Requirement Map we want to use.

1. To create our entire configuration, we needed to start the IBM Configuration Assistant for z/OS Communication Server, which opens the Main Perspective panel, as shown in Figure 9-12. In the z/OS Communications Server Technologies field, we selected the **IPSec** technology and clicked **Config**.

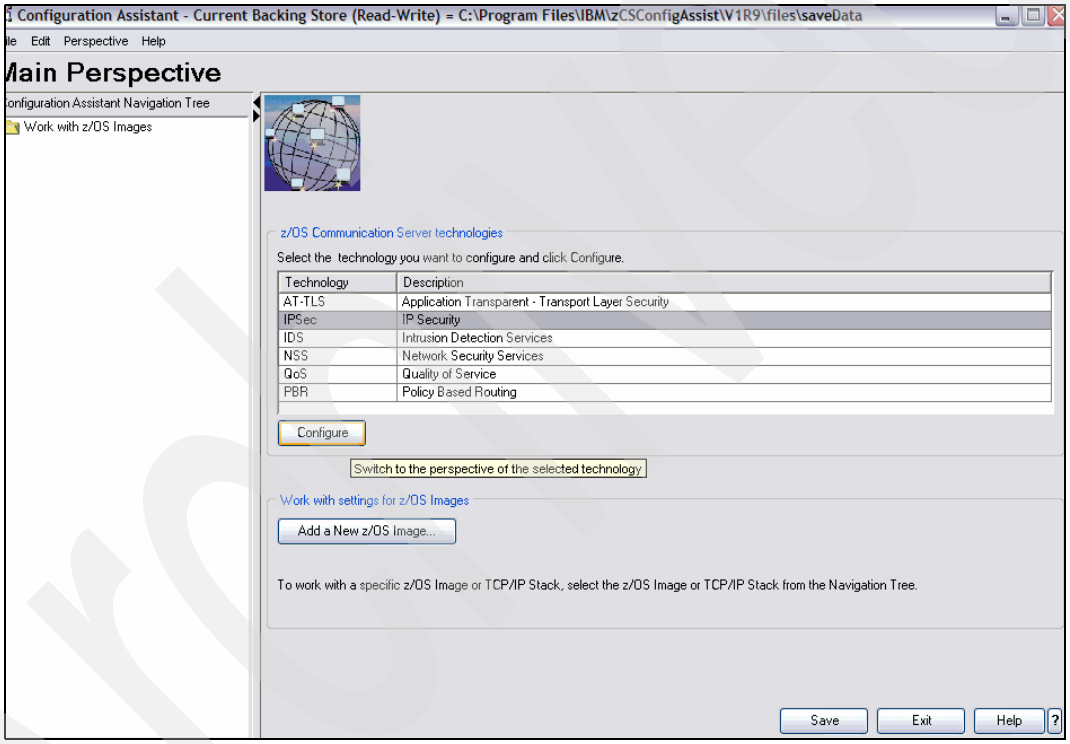


Figure 9-12 IBM Configuration Assistant for z/OS Communication Server Main Perspective panel

2. In the IPSec Perspective panel, we clicked **Requirement Maps**, as shown in Figure 9-13 on page 396.

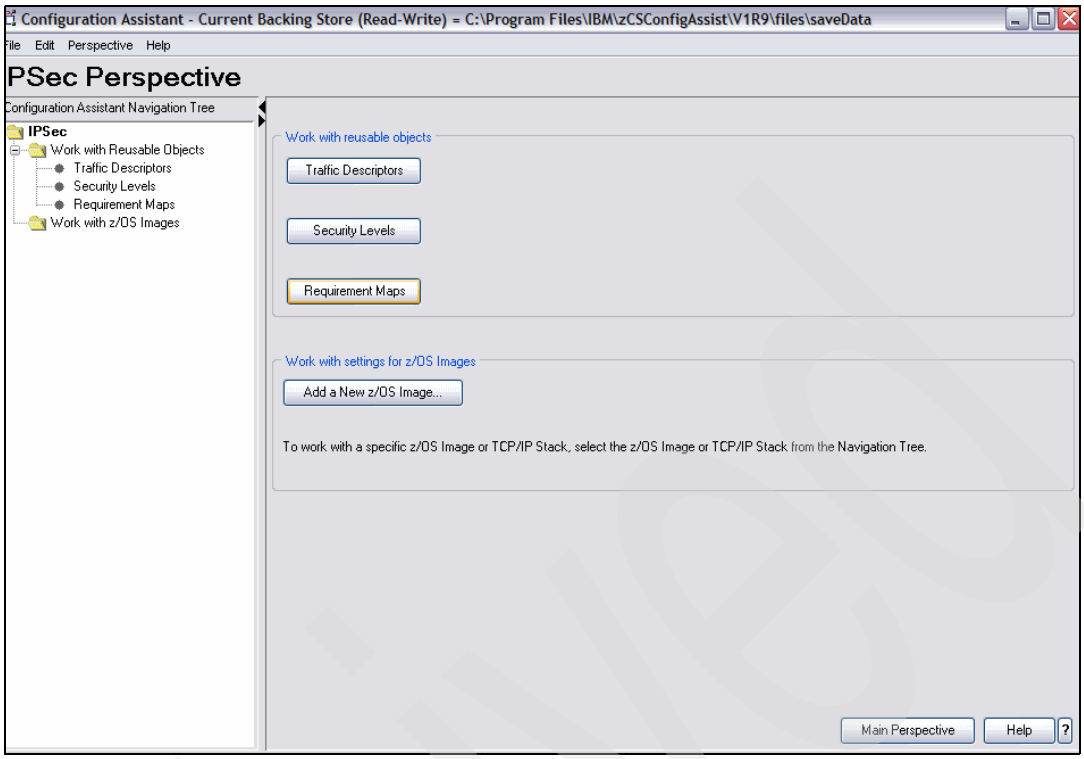


Figure 9-13 IPSec perspective panel

3. Next, in the Configuration Assistant - Requirement Maps panel, we selected **Add** to create our Requirement Maps, as shown in Figure 9-14.

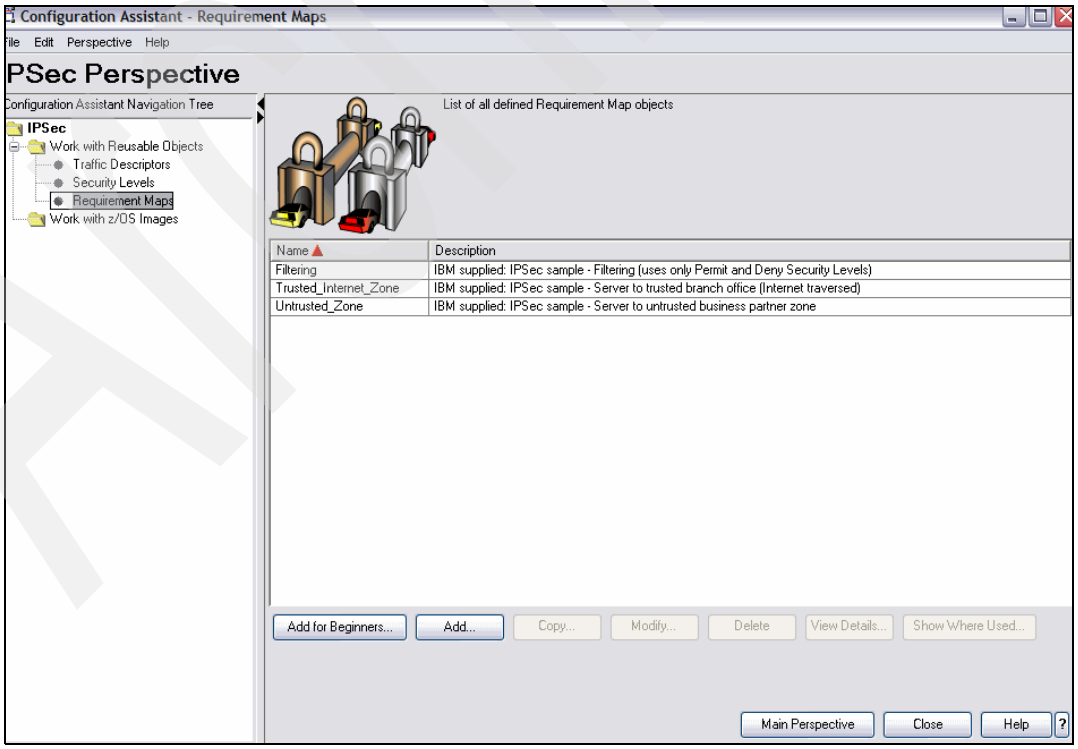


Figure 9-14 Requirement Maps panel

4. In the Requirement Map field, we selected the **All_other_Traffic traffic** object, which is the default object descriptor, denying all traffic, and then clicked **Remove**, as shown in Figure 9-15.

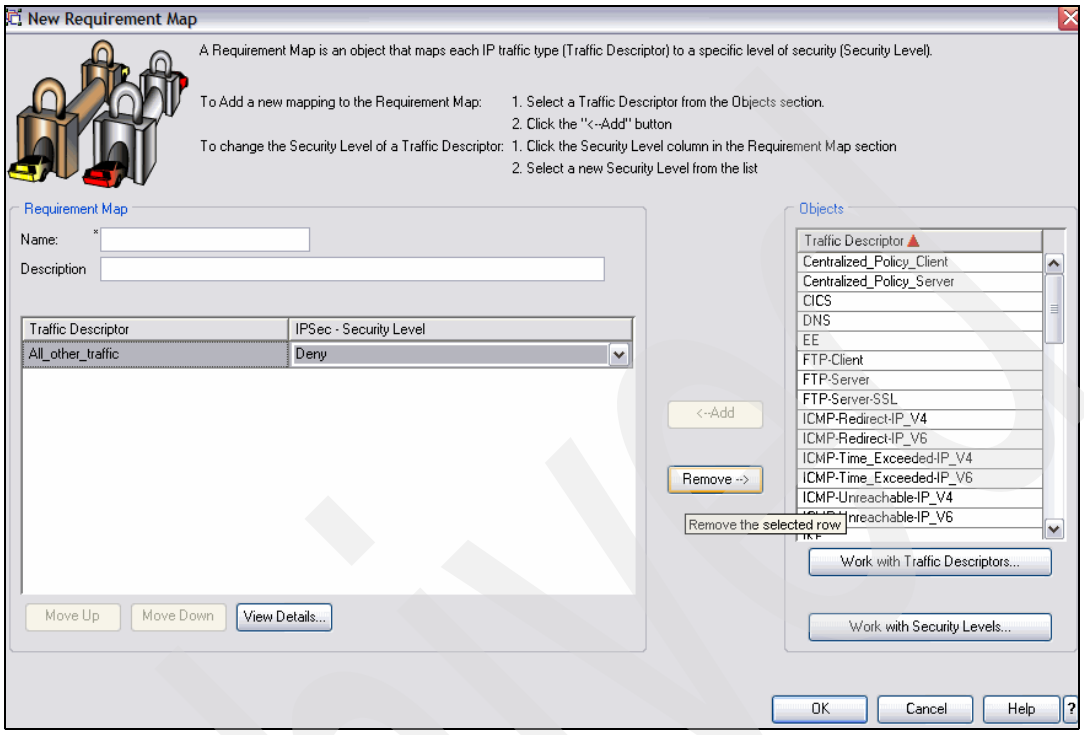


Figure 9-15 Removing the default descriptor

5. To create our **Omproute** Requirement Map, we added a Traffic Descriptor and defined the security level we wanted, as follows:

   a. In the Requirement Map field, we typed the name of our map (`Omproute`), selected **OMPROUTE-IP_V4** from the Objects list, and clicked **Add**, as shown in Figure 9-16 on page 398.

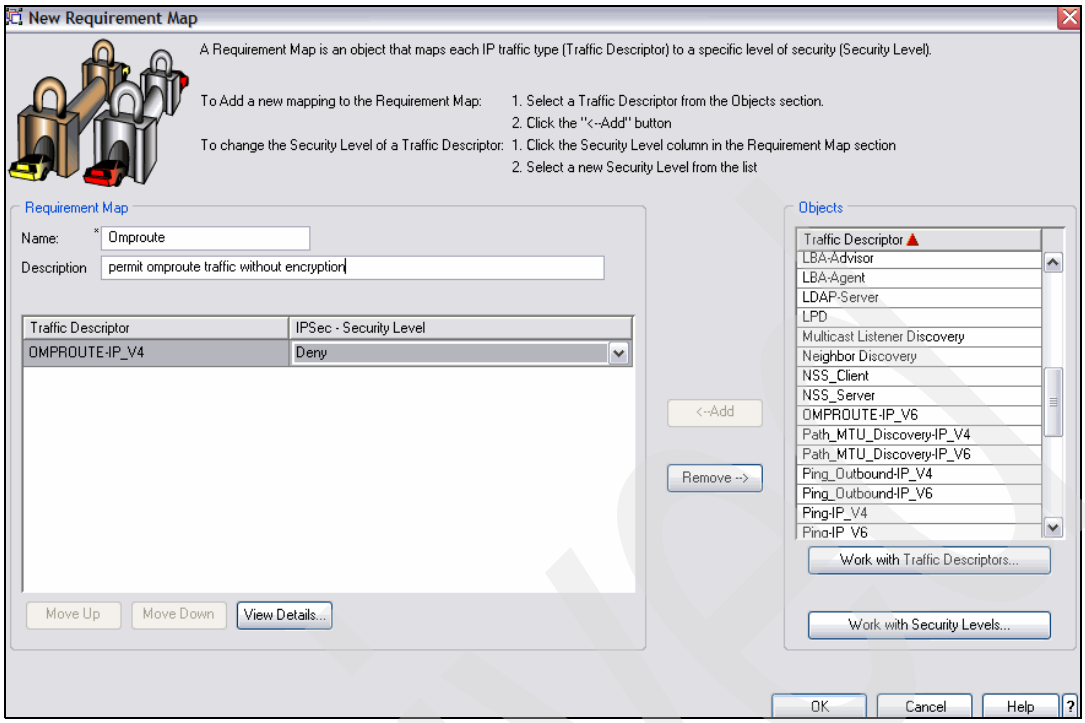


Figure 9-16 Creating the Omproute Requirement Map: selecting Omproute traffic descriptor for IPv4 network

b. Next, we clicked in the **IPSec - Security Level** list box and changed the security level to **Permit**, as shown in Figure 9-17. We clicked **OK**, and then clicked **Proceed**.

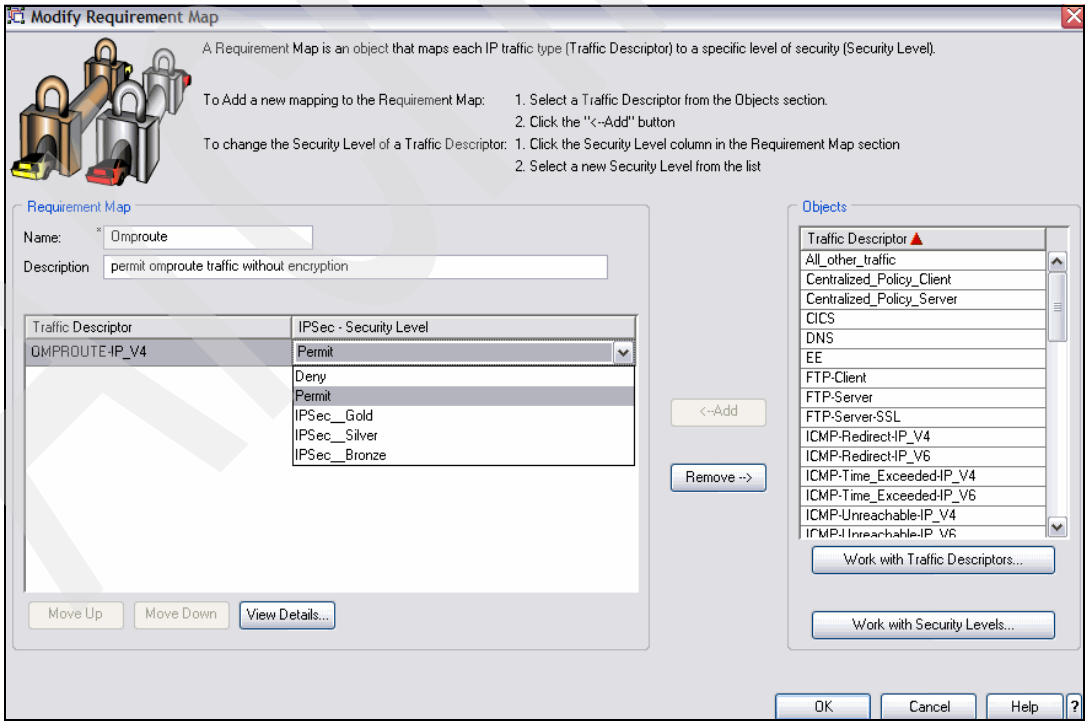


Figure 9-17 Requirement Map window: Changing IPSec security level to Permit

6. Back at the IPSec Perspective: Requirement Map panel, we clicked **Add** to create our next Requirement Map, **All_Traffic_Silver**. On the **Requirement Map** field, we typed the name and description for the rule, selected the object **All_Other_Traffic** from the **Objects** list, clicked **Add** and changed the IPSec security level to **IPSec_Silver**, as shown in Figure 9-18. We clicked **OK** to save the new map and then clicked **Close** to return to the IPSec Perspective panel.

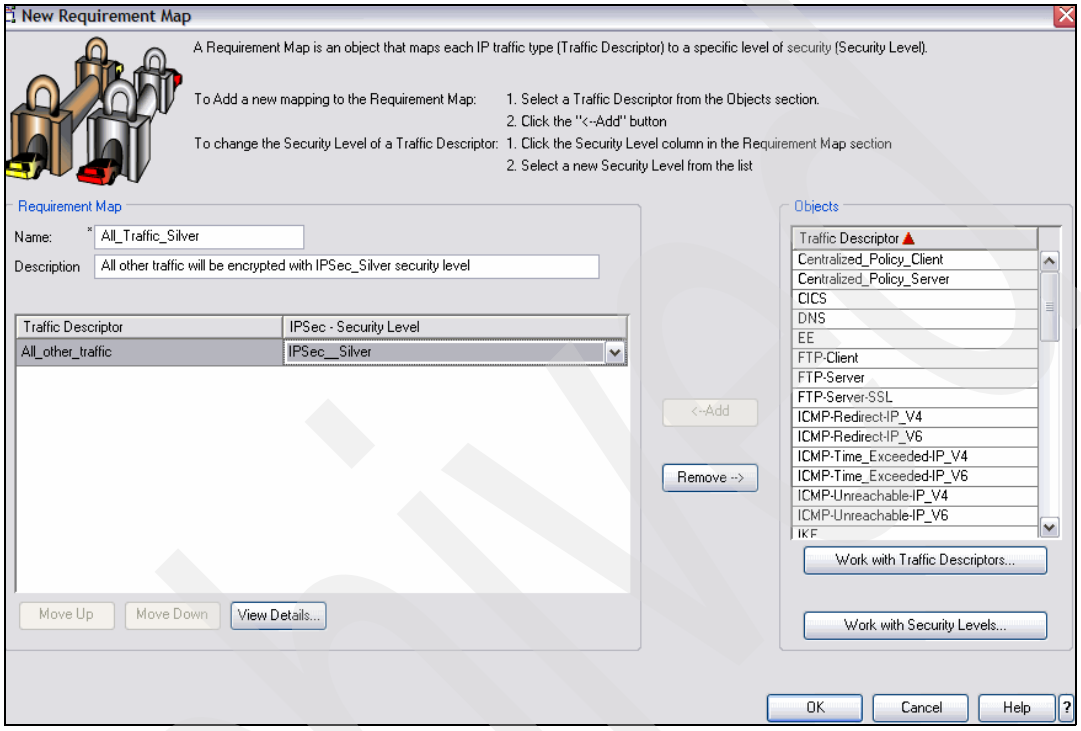


Figure 9-18 Requirement Map: Adding IPSec requirement map object for all types of traffic

The objects which we had created at this point will be used later to create the connectivity rules, in step 4.

Note: IBM provides five built-in security level objects: Permit, Deny, IPSec_Bronze, IPSec_Silver, and IPSec_Gold. The first two objects are for non-secured connections. The other three objects are used for secured connections. Each object has different definition, which can be viewed from the **IPSec** -> **Work with Reusable Objects** -> **Security Levels** panel. You can also define new security objects from this panel.

Adding a new z/OS image

To add a new z/OS image:

1. On the IPSec perspective panel, click **Add a New z/OS Image**, and then click **Next**. In the New z/OS Image: Information panel, enter the name of the z/OS image (in our case, SC32) and its description. Specify that you want to set up dynamic tunnels for this image by selecting the **Yes, this Image will have dynamic tunnels** radio button as shown in Figure 9-19 on page 400. Click **Next**.

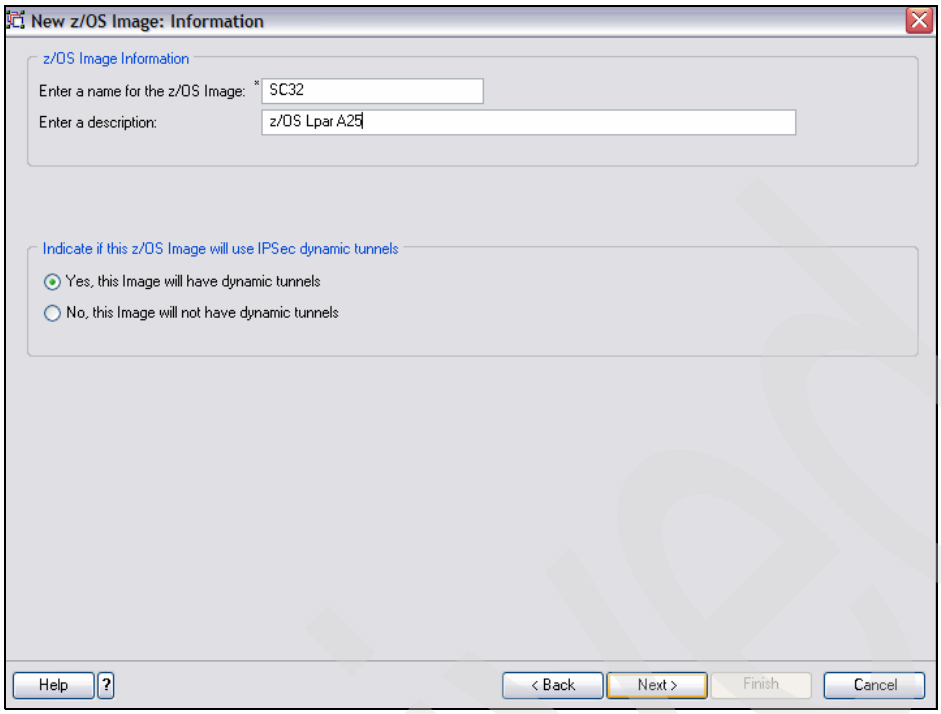


Figure 9-19 Adding z/OS image SC32

2. This opens the New z/OS Image: IPSec IKE Daemon Settings panel. In the Key ring data field, write the name of the RACF key ring database, generated for the IKE server as described in “Creating a RACF key ring” on page 388. We used the key ring named IKED/IKED_keyring database, as shown in Figure 9-20. Click **Next**, then click **Next** again and then click **Finish**.

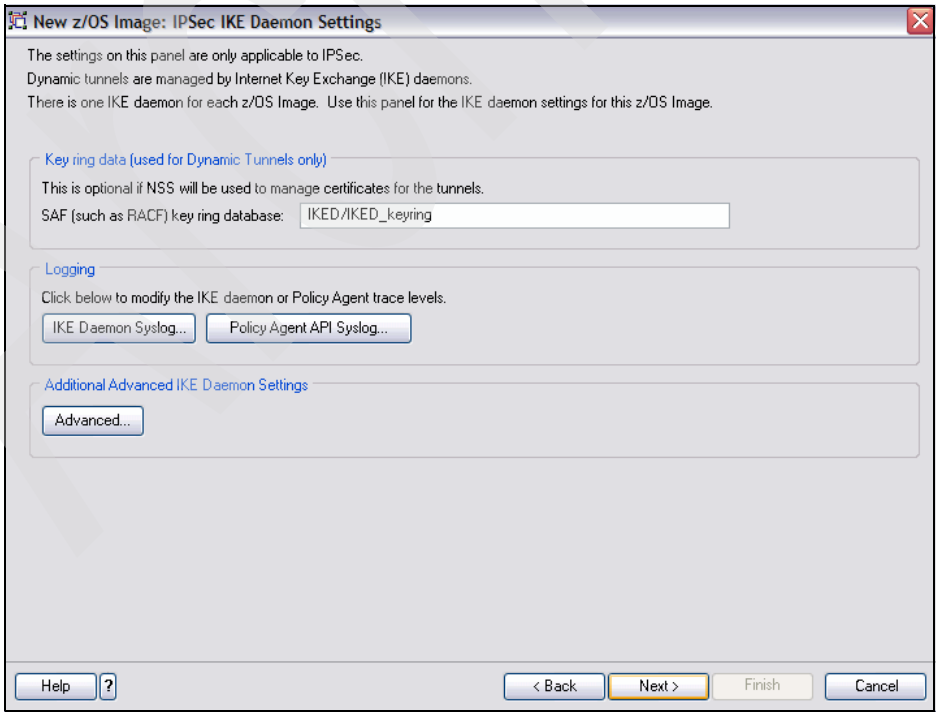


Figure 9-20 Setting our key ring name

Back at the IPSec Perspective panel, to proceed to the next step click **Add new TCP/IP Stack**; this opens the New TCP/IP Stack Wizard panel.

Adding a TCP/IP stack for the new z/OS image

To add a TCP/IP stack for the new z/OS image:

1. In the Welcome window, click **Next**.
2. In the New TCP/IP Stack: Name panel, specify the name of the TCP/IP stack that you want to configure in the z/OS image. (Our stack name is `TCPIPA`.) Select the **Yes, this Stack will have dynamic tunnels** radio button to specify that you want to set up dynamic tunnel for this stack, as shown in Figure 9-21. Click **Next**.

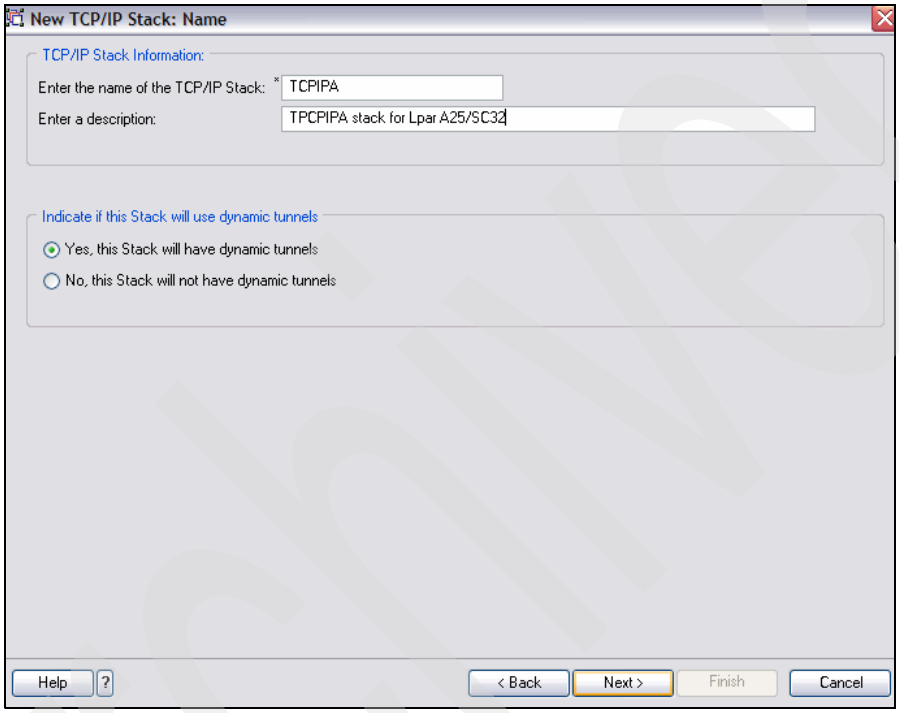


Figure 9-21 New TCP/IP stack name

3. The window shown in Figure 9-22 opens. There are two options for configuring a local identity:

   – Single tunnel identity: One identity for all IP addresses on the TCP/IP stack
   – Separate tunnel identities: One identity for each IP address on the TCP/IP stack

   We set up a single tunnel for the stack. We selected the **I want to use a single identity for all IP addresses** radio button. You can identify the stack in four different ways, as shown by the radio buttons in the window. (We identified our stack by its IP address, `10.1.1.50`.) Click **Next**.

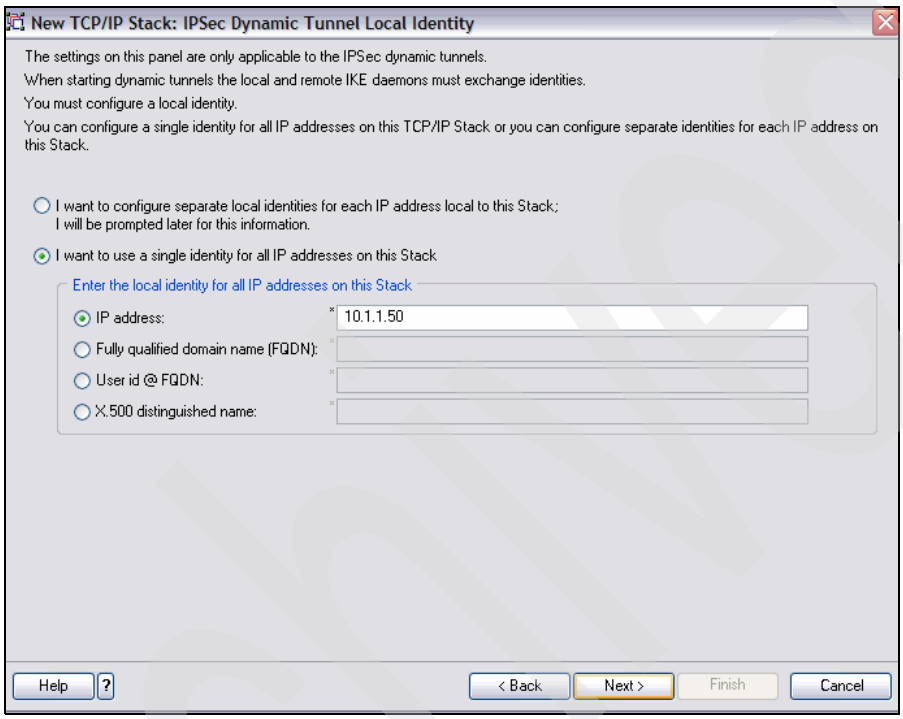


Figure 9-22 Identifying the local dynamic tunnel

4. This opens the window shown in Figure 9-23 on page 403. This window is used to specify the log levels. For our test, we selected the radio buttons to enable filter logging and to log all implicit deny events. When everything is working, you can disable the logging later.

   This window is also used to specify the Network Address Translation (NAT) Traversal Policy. For the IPSec test, we disallowed NAT. Also, we decided not to filter the IPSec headers.

   Click **Next**, and then **Finish**.

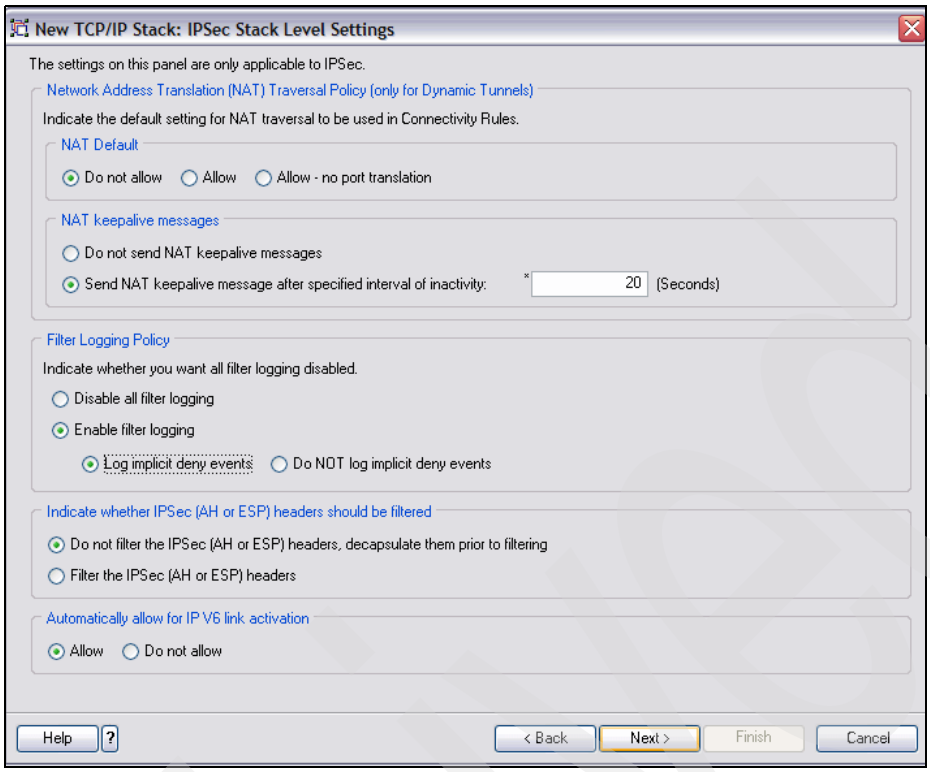


Figure 9-23 IPSec log level settings

Back at the IPSec Perspective panel, we can proceed to the next step, to add the connectivity rules.

Adding connectivity rules for the TCP/IP stack

To add connectivity rules for the TCP/IP stack:

1. In the IPSec Perspective panel we see our TCP/IP Stack with a red warning, meaning the stack is added, but the configuration is incomplete. The Connectivity Rule field for this stack is empty. Click **Add** to implement the new rules, as shown in Figure 9-24 on page 404.

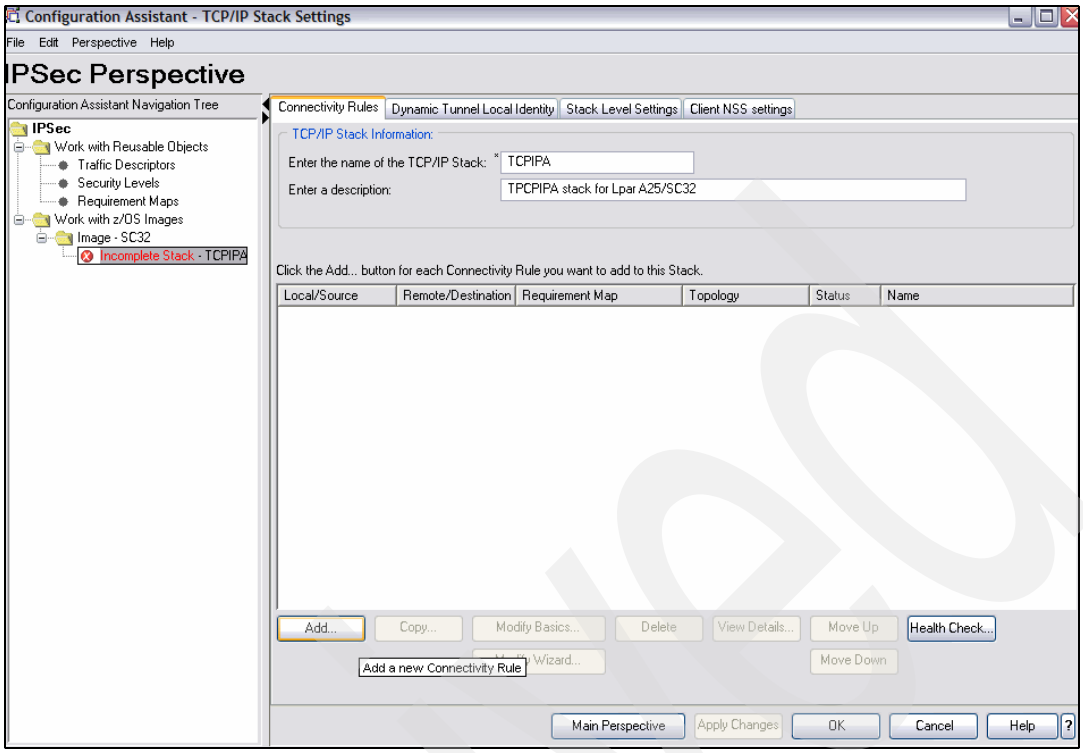


Figure 9-24 Adding connectivity rules to stack TCPIPA

2. We created two connectivity rules:
   - Omproute: A rule that permits omproute traffic between all IPv4 addresses.
   - All_Traffic_Silver: A rule that forces all other traffic to be secured, using IP security with security level IPSec_Silver.

   In this way, we made sure that each connection between the two LPARs were encrypted and therefore secured.

Create the Omproute connectivity rule first. In the Network Topology window, select the first radio button, as shown in Figure 9-25.

Click **Next**.

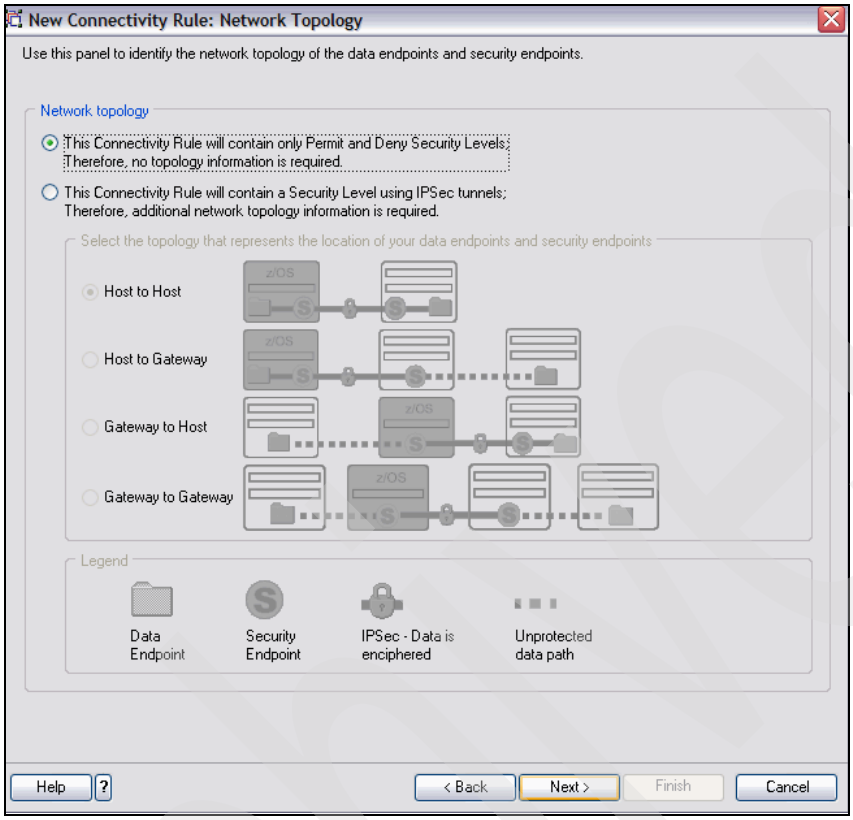


Figure 9-25 Connectivity Rule: Network Topology window - defining non-secure network topology

3. In the Connectivity Rule: Data Endpoints window, select **All V4 addresses** for Source data endpoint and for Destination data endpoint. Enter a name for the connectivity rule, as shown in Figure 9-26. Click **Next**.

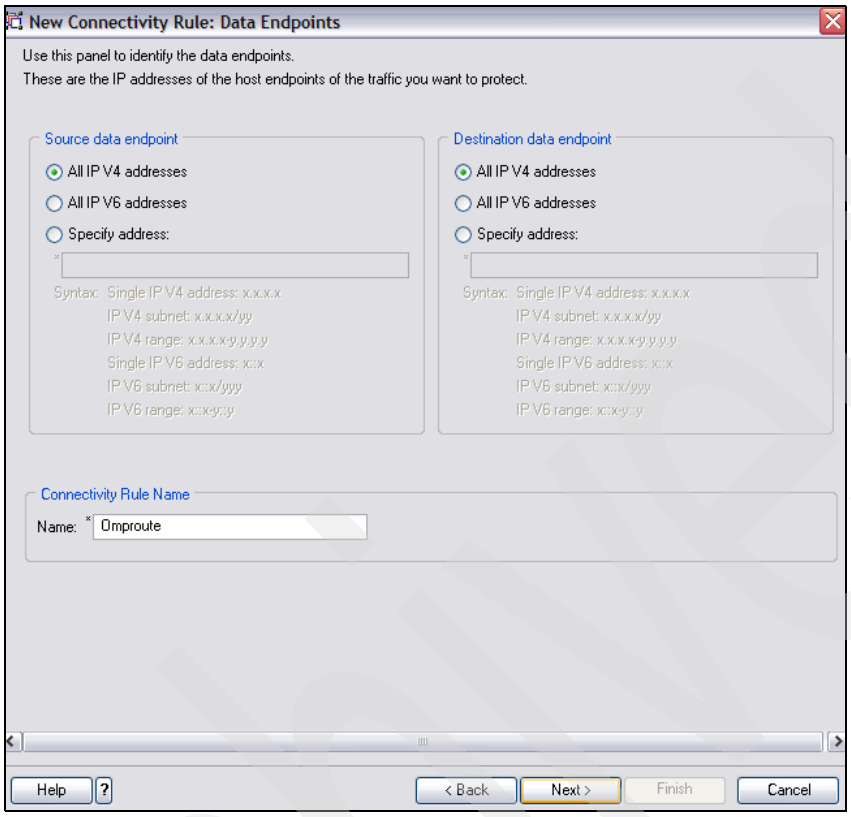


Figure 9-26 Connectivity Rule: Data Endpoints panel

4. In the Connectivity Rule: Select Requirement Map window, select the **Omproute** requirement map object from the list, as shown in Figure 9-27.

Notes:

- The Select Requirement Map window shows all the requirement map objects that are defined in the current configuration file, including the ones we defined (see “Adding Requirement Map objects” on page 395) and the IBM-supplied samples.
- Adding a new Requirement Map object from this window affects the list of all defined requirement map objects. That is, the new object, which is added from this panel, is also added to the general list, and therefore to the other z/OS images and to the other TCP/IP stacks as well. This is also true for any modification that you make to the objects.

  Our recommendation: Use the Requirement Map window under Work with Reusable Objects to make modifications to the requirement map objects. For more information, see “Adding Requirement Map objects” on page 395.

Click **Next** and then **Finish**.

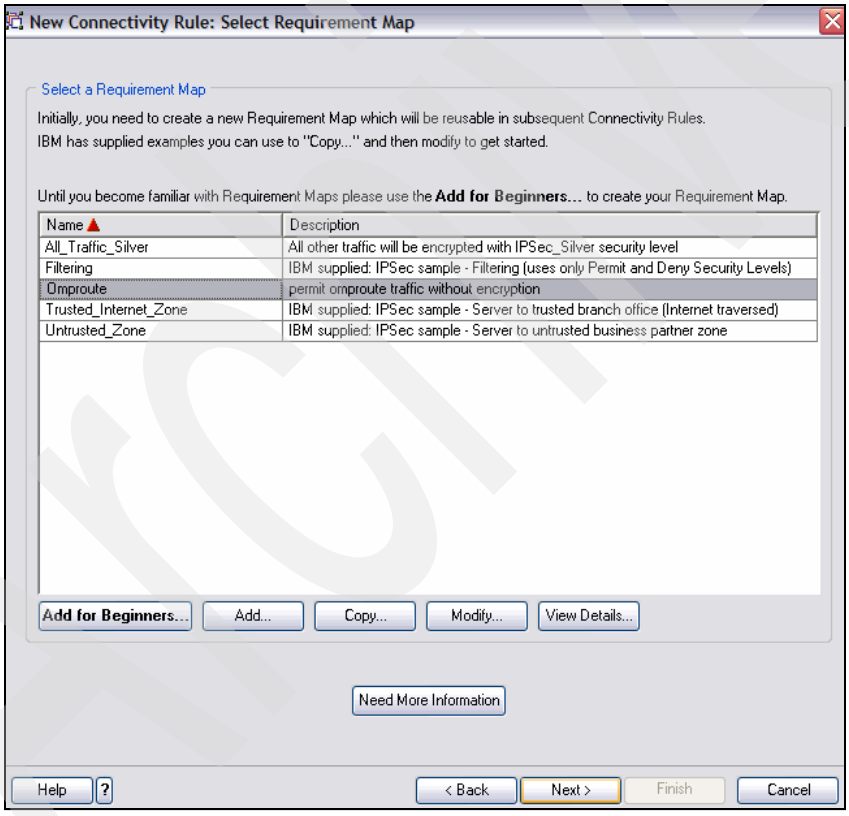


Figure 9-27 Connectivity Rule: Select Requirement Map window

5. Back at the IPSec Perspective panel, we added the next connectivity rule clicking the **Add** button, as shown in Figure 9-28

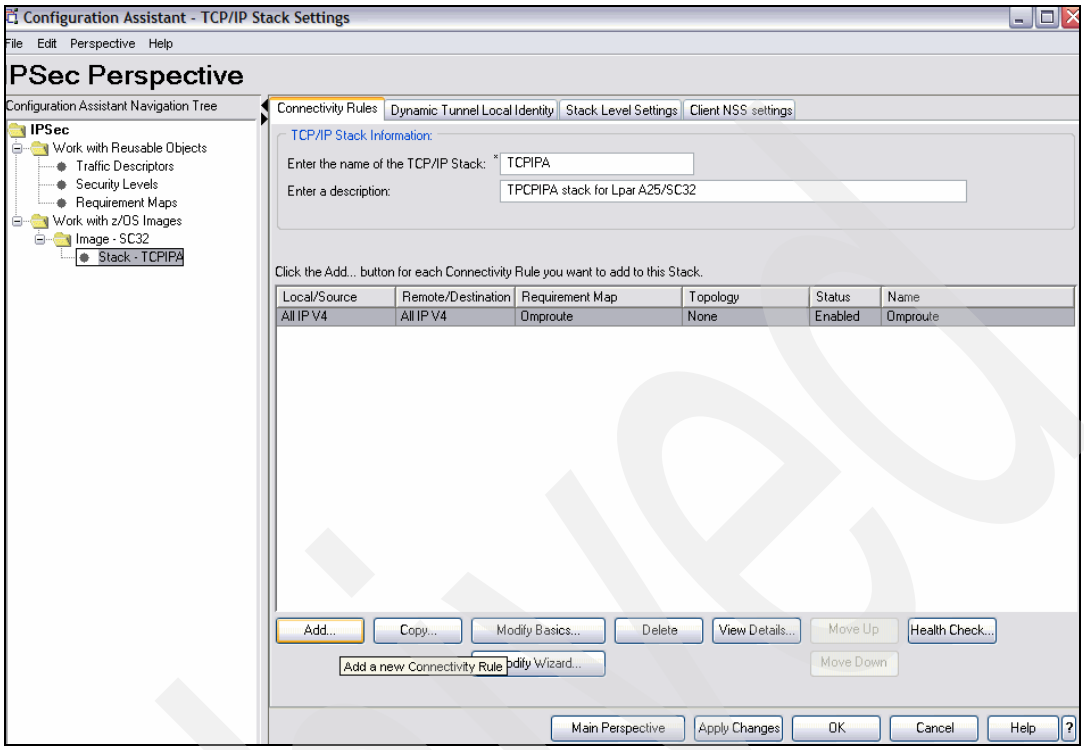


Figure 9-28 Add new connectivity rule

6. In the Connectivity Rule: Network Topology panel, under the text “This Connectivity Rule will contain a Security Level using IPSec tunnels”, select the **Host to Host** topology option as shown in Figure 9-29. Click **Next**.

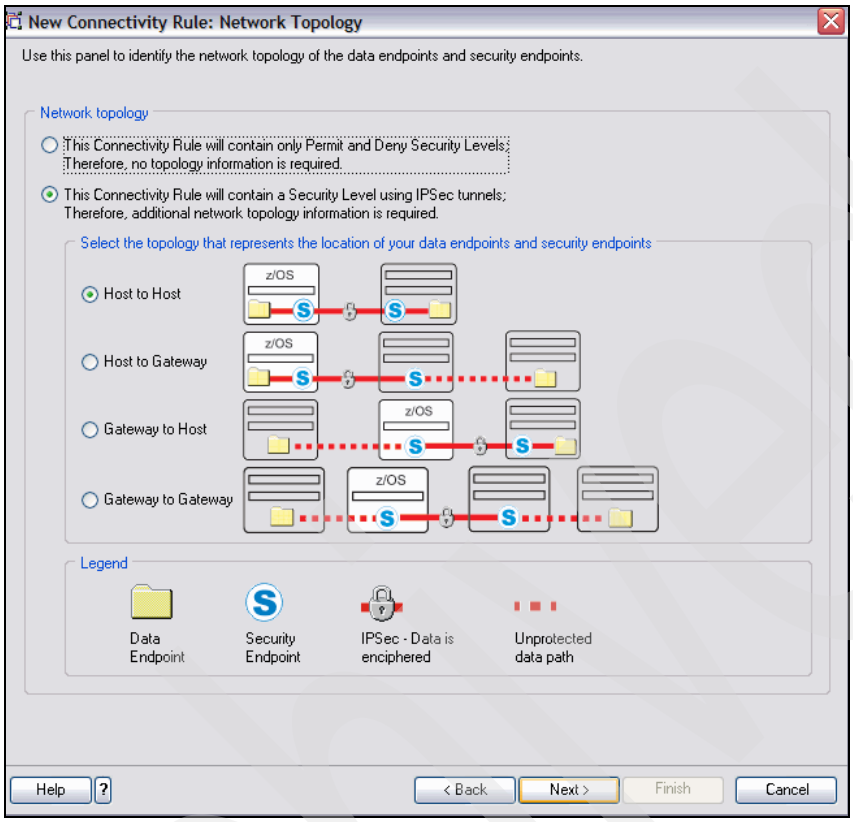


Figure 9-29 Connectivity Rule: Network Topology window: Defining host-to-host connectivity

Note: An IPSec VPN does not have to be an end-to-end (called host-to-host) entity. In many instances, a VPN might not begin at your local workstation. A VPN endpoint could begin at the border between your secure network and a non-secure one. This would be considered a gateway VPN.

The same concept applies to the remote end: the VPN may end at the gateway to the destination, secure network. Or, the VPN might make it all the way to the destination host, or workstation.

7. This opens the Connectivity Rule: Data Endpoints panel shown in Figure 9-30. Specify the endpoints of the connection by entering the IP addresses of the z/OS images and give a name for the connection. Click **Next**.

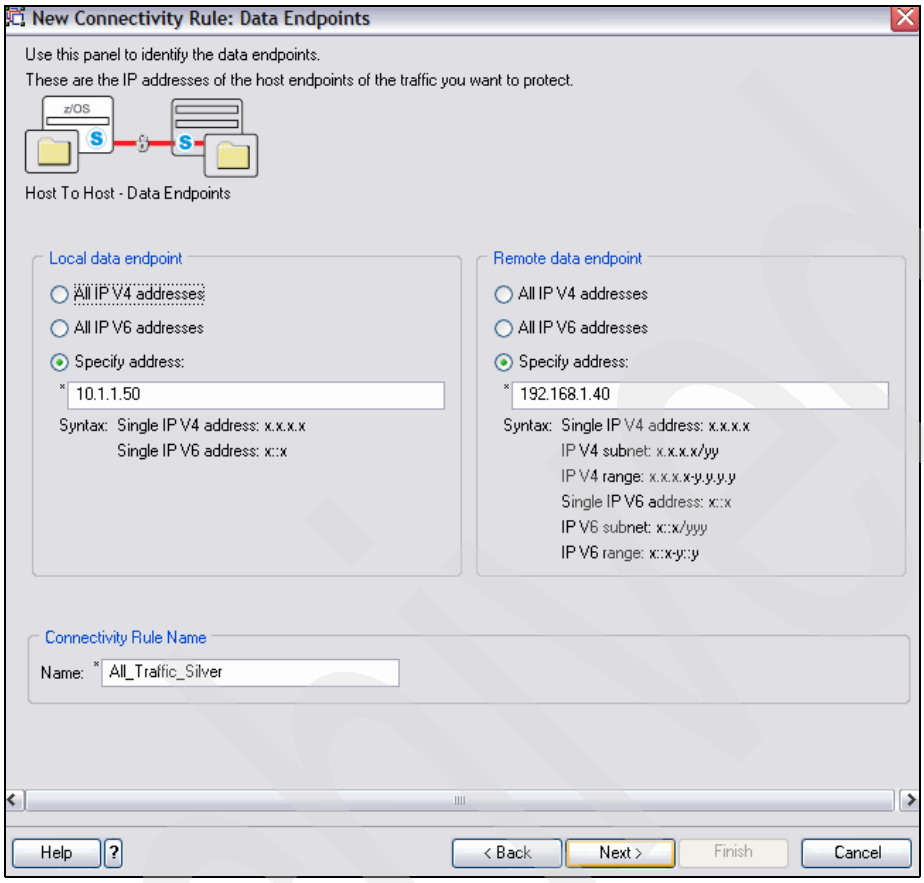


Figure 9-30 Connectivity Rule: Data Endpoints window - defining the data endpoints

8. In the Connectivity Rule: Select Requirement Map panel, select **All_Traffic_Silver** requirement map object, as shown in Figure 9-31. Click **Next**.

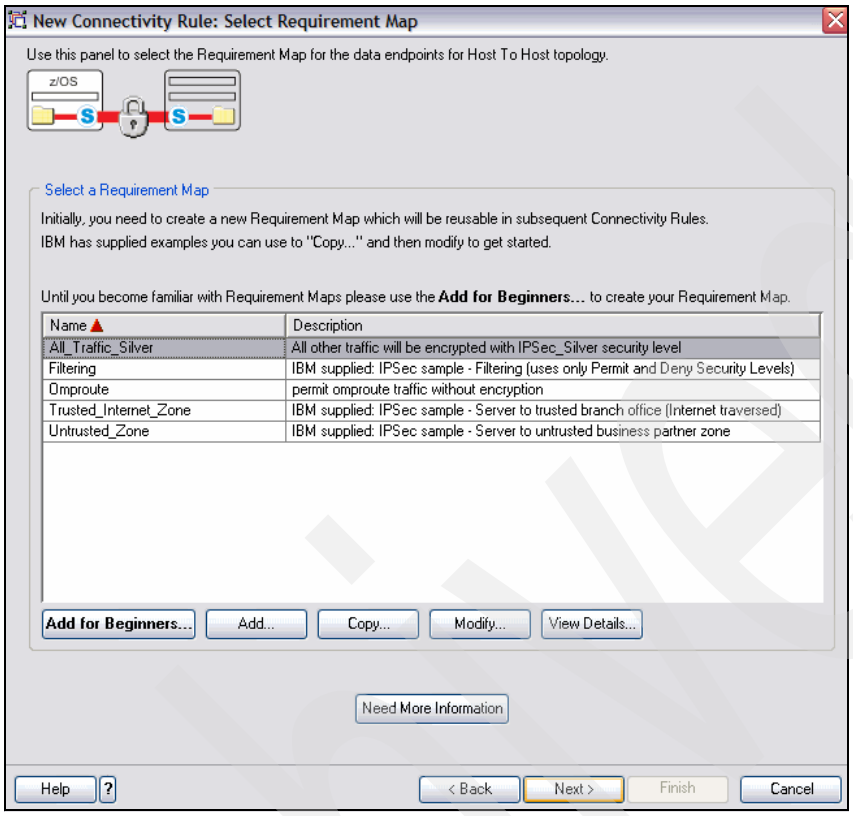


Figure 9-31 Select Requirement Map window: selecting the secured traffic object

9. In the Connectivity Rule: Remote Security Endpoint Information panel, select the remote IKE identity for the remote endpoint. We chose **IP address** and entered the IP address of the remote z/OS image of this rule (in our case, `192.168.1.40`). There are two ways to authenticate the remote IKE peers:
   - RSA signature
   - Shared key, in which you can specify a shared key as American Standard Code for Information Interchange (ASCII) string, an Extended Binary Coded Decimal Interchange Code (EBCDIC) string, or a hexadecimal string.

> **Tip:** RSA signature authentication takes advantage of the public key cryptography and X.509 certificate capabilities. It is very secure, but requires the (relatively minor, in most cases) overhead of establishing certificate authorities, personal certificates and certificate repositories (key rings).
>
> Shared key authentication can be quite robust and for testing, it is quite easy. But, ultimately, the safe distribution and storage of shared keys can present a long-term issue: where do you keep this shared secret so that you will know what it is a year from now?

For our test environment, we chose **Shared key** and entered an EBCDIC string, as shown in Figure 9-32. Click **Next**, and **Finish** in the next panel.

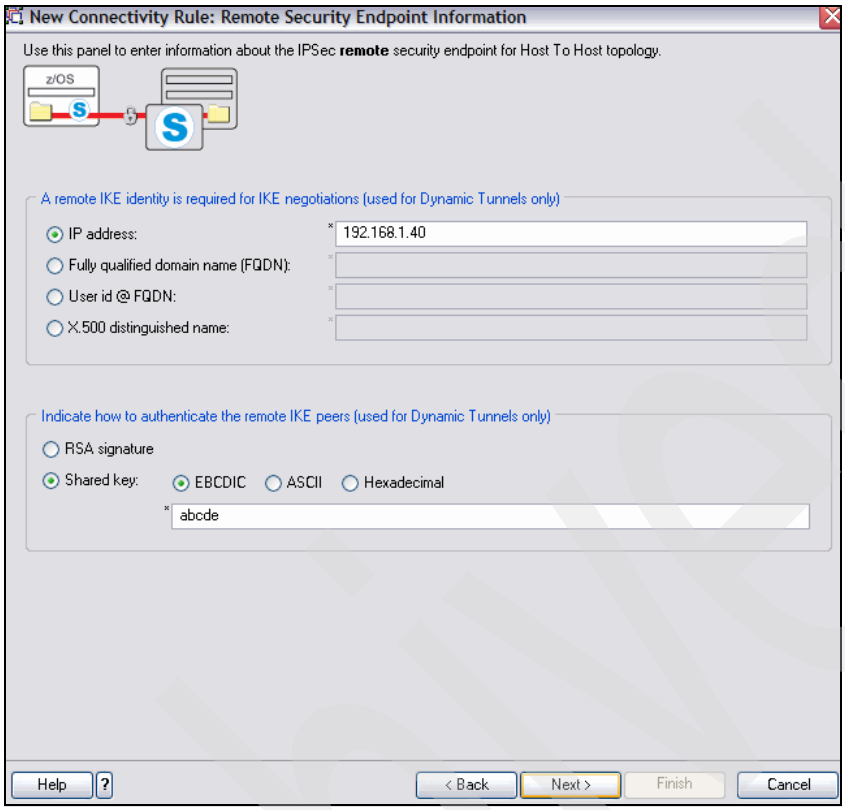


Figure 9-32 Connectivity Rule: Remote Security Endpoint Information window

Repeat the same procedure to add all the other objects that you want to select for the current stack.

Note: There is an importance to the order of the connectivity rules in a stack. When the Policy Agent is running and there is an attempt to send packets from or to the z/OS image, the first rule in the list is checked first and if there is a match with the definition of this rule, it will be used. If there is no match, the second rule will be checked and so on.

There are performance implications too, so if possible, place your “most used” rules at the top of the list.

10. Back at the IPSec Perspective panel, we click **Apply Changes** to save the configuration and **OK** to go back to the Main IPSec Perspective panel, finishing the configuration process for this z/OS image, as shown in Figure 9-33.

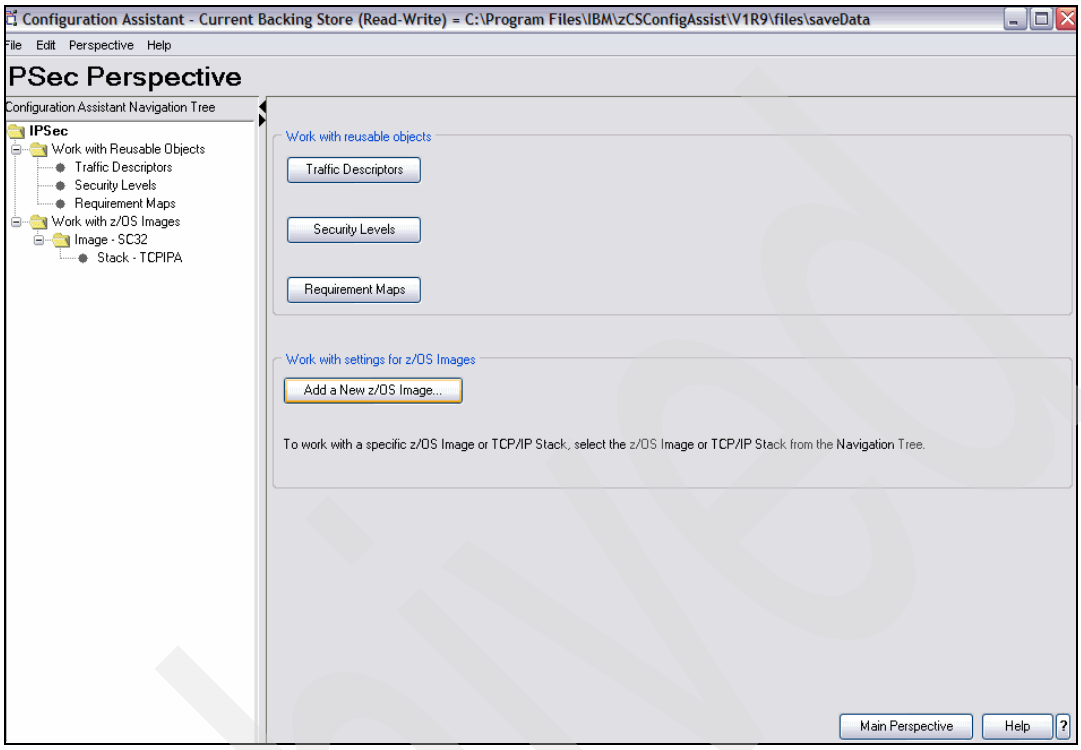


Figure 9-33 Configuration for image SC32 - Stack TCPIPA is finished, create the next Image

This completes the creation of the policy for one endpoint of the tunnel. We now proceed to create the other endpoint of our tunnel.

Adding another new z/OS image

Similarly, follow steps 2- 4 (see “Adding a new z/OS image” on page 399 through “Adding connectivity rules for the TCP/IP stack” on page 403) to create the second endpoint with TCP/IP stack and connectivity rules. In our case, we created another z/OS image called SC33 in the configuration file with TCP/IP stack named TCPIPE and implemented the same connectivity rules, as shown in Figure 9-34 on page 414.

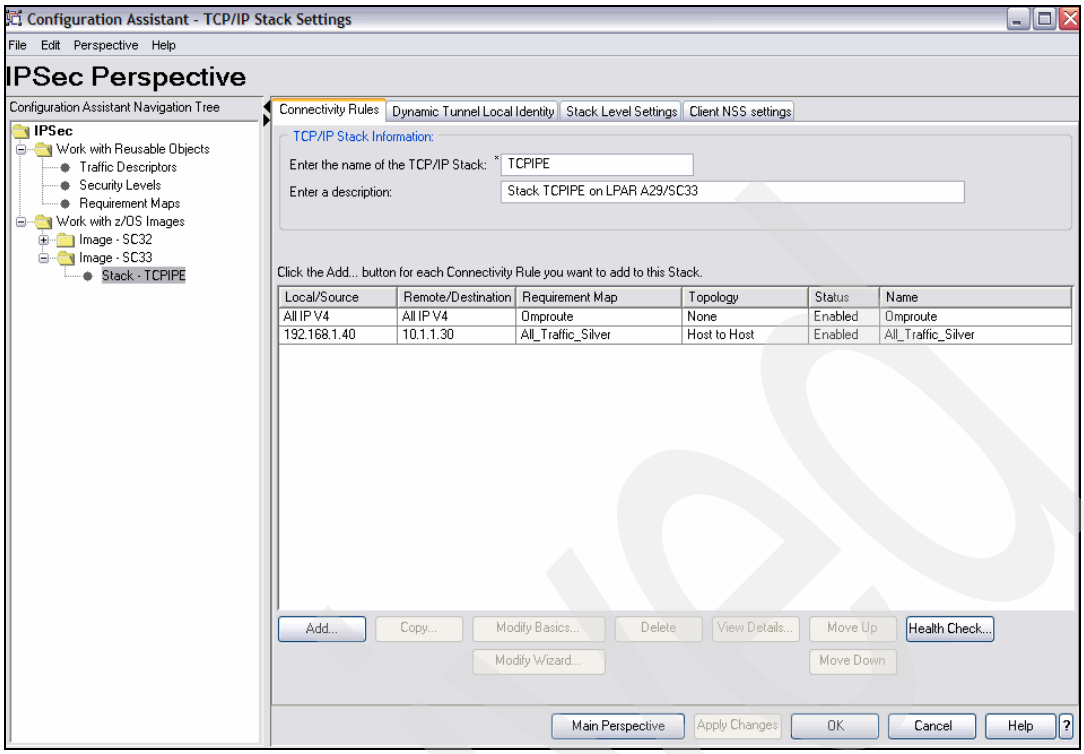


Figure 9-34 Stack TCPIPE settings

At this point, we had completed the configuration process of our scenario. The next step was to use IBM Configuration Assistant for z/OS Communication Server to install the configuration files.

Tip: Click the **Health Check...** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The Health Check reports on errors in your policy and we found it helpful.

9.5.2 Installing the configuration files

After saving the generated configuration for all stacks, we install these configuration files on each z/OS image, using FTP to send the policy files to the z/OS system, as follows:

1. Right-clicking **Image-SC32** in the Configuration Assistant Navigation Tree area gives you the option to install the configuration file. Click **Install Configuration Files.**
2. In the Installation - Stack= TCPIPA panel, select **TCPIPA - IPSec: Policy Agent Stack Configuration** in the configuration files list. To install the selected configuration file in the z/OS image, click **FTP,** as shown in Figure 9-35 on page 415.

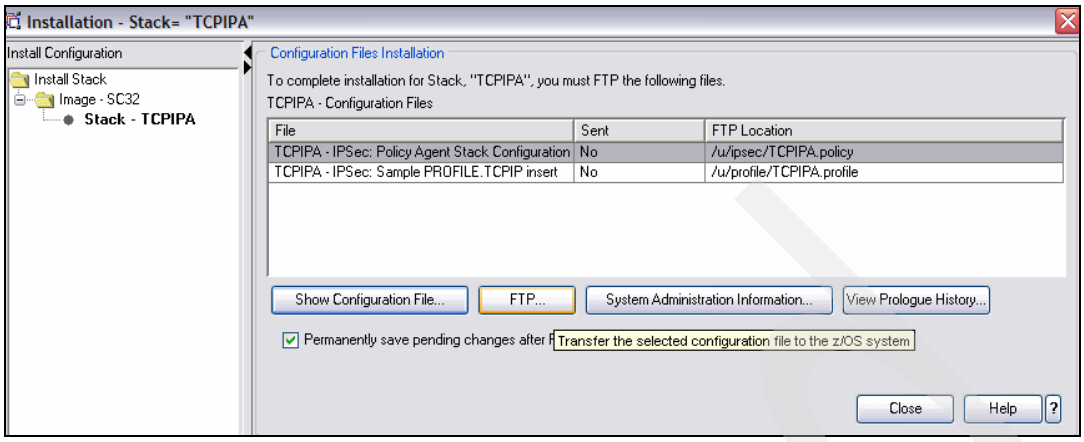


Figure 9-35 Installation - Stack- TCPIPA panel

3. A FTP Configuration File panel is provided. Enter the necessary information in each field, then click **Send,** as shown in Figure 9-36.

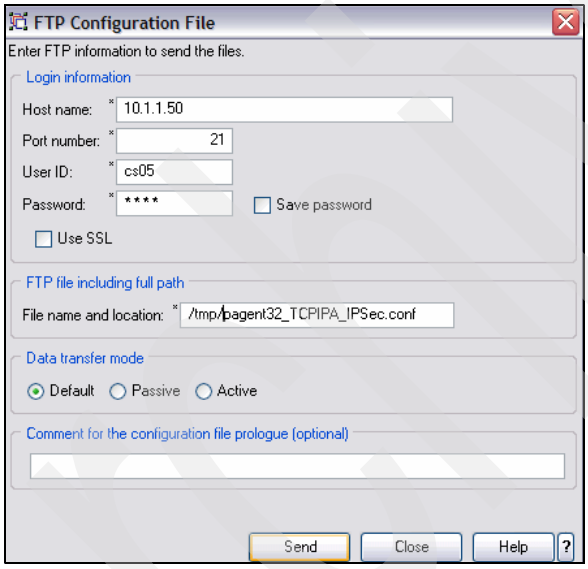


Figure 9-36 FTP configuration file

4. We repeated this procedure to install the generated configurations files for all stacks in our scenario.
5. After the files were transmitted to our z/OS image, the process was still not complete. So the next step is to customize the files to be used by PAGENT. In our scenario we sent our files to a HFS directory, so we used the UNIX shell to proceed with the customization. In the UNIX shell we moved the files to the /etc directory, and changed the pagent configuration files to point to our IPSec configuration file, as shown in part in Example 9-15.

Example 9-15 pagent32_TCPIPA.conf contents with IPSec

```
# ###################################################################
#                  IMAGE FILE FOR Stack TCPIPA on LPAR A25/SC32
# ###################################################################
#RoutingConfig /etc/pagent32_PBR.conf
IPSecConfig   /etc/pagent32_TCPIPA_IPSec.conf
```

6. Update the Policy Agent with the **modify pagent,update** command, as seen in Example 9-16. The IPSec tunnel will become active next time we generate traffic between the two endpoints.

Example 9-16 The modify pagent,update command results

```
F PAGENT,UPDATE
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IPSEC
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
```

After this command executed, our scenario was implemented and ready to be used.

9.5.3 Verification of IPSec between two z/OS images

This section describes the steps needed to verify that the IP security configuration is working properly. The steps are as follows:

- Check that IPSECURITY and SOURCEVIPA is configured to a TCP/IP stack in both z/OS images.
- Check that Policy Agent is updated with IPSEC policy.
- Display the active IP security policy.
- Try to run FTP between the z/OS images.
- Display the IKED tunnels and dynamic (or manual, depending on your configuration) tunnels.

Checking that the TCP/IP stack is configured for IP security

On the UNIX shell, run the following command:

```
netstat -p TCPIPA -f
```

The first parameter, -p, allows a specific stack to be queried. Check that the IpSecurity field under the IP Configuration Table is equal to Yes. Part of the output generated by the **netstat** command is shown in Example 9-17.

Example 9-17 netstat -f for stack TCPIPA, on SC32 z/OS image

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPA          13:47:20
TCP Configuration Table:
DefaultRcvBufSize:  00065536  DefaultSndBufSize: 00065536
DefltMaxRcvBufSize: 00262144
...
IP Configuration Table:
Forwarding: Yes    TimeToLive: 00064  RsmTimeOut:  00060
IpSecurity: Yes
ArpTimeout: 01200  MaxRsmSize: 65535  Format:      Long
IgRedirect: Yes    SysplxRout: Yes    DoubleNop:   No
StopClawEr: No     SourceVipa: Yes
MultiPath:  No     PathMtuDsc: No     DevRtryDur:  0000000090
DynamicXCF: Yes
SubNet: 255.255.255.0    Metric: 08
...
```

Checking that the Policy Agent is active with IP security policy

The first thing you need to check is that the Policy Agent has read in the current policies.

After you FTP the configuration file to the z/OS image, the following command is used to update the PAGENT (where pagent is the started task name for the Policy Agent):

```
MODIFY PAGENT,UPDATE
```

If no changes have been made since the last time the polices were read, you will receive the following message:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : NONE
```

If changes have been made, then you will receive the following message instead:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IPSEC
```

Displaying the active IP security policy

With TRMD running, syslogd is the repository for all Policy Agent and IKE daemon messages.

```
ipsec -p TCPIPA -f display -a YO
```

The first parameter, `-p`, allows specific stacks policies to be queried. The second parameter, `-f`, indicates filters are to be displayed. The third parameter, `-a`, indicates we want to show only a list of all dynamic anchor filters. In the resulting display, look at the `Source:` field. It shows `Stack Policy`, meaning the IP security policy is installed and active.

In our scenario the command shows two policy filters with type dynamic anchor, from a total of 26, as shown in Example 9-18.

Example 9-18 ipsec -f display for stack TCPIPA, on SC32 z/OS image

```
CS V1R9 ipsec  Stack Name: TCPIPA  Wed Sep 26 14:39:47 2007
Primary:  Filter          Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    Current           TotAvail: 26
Logging:  On              Predecap: Off               DVIPSec:  No
NatKeepAlive:  20

FilterName:                   All_Traffic_Silver~7
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     YO
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Outbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                10.1.1.50
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   n/a
```

```
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  192.168.1.40
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     n/a
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
CreateTime:                   2007/09/26 13:38:39
UpdateTime:                   2007/09/26 13:38:39
*****************************************************************************
FilterName:                   All_Traffic_Silver~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                192.168.1.40
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   n/a
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  10.1.1.50
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     n/a
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
RmtUdpEncapPort:              n/a
CreateTime:                   2007/09/26 13:38:39
UpdateTime:                   2007/09/26 13:38:39
*****************************************************************************
2 entries selected
```

In the output listed in Example 9-18 on page 417, filter `All_Traffic_Silver~7` is listed twice. The rule has been expanded into an inbound rule and an outbound rule. A quick glance down

the list of parameters provides information about the VPN action, source, and destination IP addresses and source and destination ports.

The **ipsec** display option does not provide complete details about the VPN's policies. More complete information can be displayed using the following command:

```
pasearch -p TCPIPA -s DynamicVpn
```

The beginning of the resulting display is shown in Example 9-19.

Example 9-19 pasearch -p TCPIPA -s DynamicVpn resulting display

```
TCP/IP pasearch CS V1R9                  Image Name: TCPIPA
  Date:                 09/26/2007       Time:  14:50:54
  IPSec Instance Id:    1190828319

policyRule:             All_Traffic_Silver~7
  Rule Type:            IpFilter
  Version:              3                Status:           Active
  Weight:               106              ForLoadDist:      False
  Priority:             6                Sequence Actions: Don't Care
  No. Policy Action:    2                ConditionListType: CNF
  IpSecType:            policyIpFilter
  policyAction:         IpSec~LogYes
   ActionType:          IpFilter GenericFilter
   Action Sequence:     0
  policyAction:         IPSec__Silver
   ActionType:          IpFilter DynamicVpn
   Action Sequence:     0
...
```

Another format of the **ipsec** command can be using `-t` for traffic test. This command returns all the rules in the current filter table that match the given traffic type; for example:

```
ipsec -p TCPIPA -t 10.1.1.50 192.168.1.40 tcp 21 0 out
```

Running this command from SC32 (IP: 10.1.1.50) results in the output of all matching rules in the current filter table of SC32 that match FTP traffic, as shown in Example 9-20.

Example 9-20 The output of ipsec -t command on the SC32 z/OS image

```
CS V1R9 ipsec  Stack Name: TCPIPA  Wed Sep 26 15:05:45 2007
Primary:  IP Traffic Test Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    n/a               TotAvail: 3
TestData: 10.1.1.50  192.168.1.40  tcp 21 0 out

FilterName:                 All_Traffic_Silver~7
FilterNameExtension:        1
GroupName:                  n/a
LocalStartActionName:       n/a
VpnActionName:              IPSec__Silver
TunnelID:                   Y0
Type:                       Dynamic Anchor
State:                      Active
Action:                     Permit
Scope:                      Local
Direction:                  Outbound
OnDemand:                   Yes
SecurityClass:              0
Logging:                    All
Protocol:                   All
```

```
ICMPType:                    n/a
ICMPType:                    n/a
ICMPCode:                    n/a
OSPFType:                    n/a
TCPQualifier:                n/a
ProtocolGranularity:         Rule
SourceAddress:               10.1.1.50
SourceAddressPrefix:         n/a
SourceAddressRange:          n/a
SourceAddressGranularity:    Packet
SourcePort:                  n/a
SourcePortRange:             n/a
SourcePortGranularity:       Rule
DestAddress:                 192.168.1.40
DestAddressPrefix:           n/a
DestAddressRange:            n/a
DestAddressGranularity:      Packet
DestPort:                    n/a
DestPortRange:               n/a
DestPortGranularity:         Rule
OrigRmtConnPort:             n/a
RmtIDPayload:                n/a
RmtUdpEncapPort:             n/a
CreateTime:                  2007/09/26 13:38:39
UpdateTime:                  2007/09/26 13:38:39
*************************************************************************
```

In the output shown in Example 9-20 on page 419, you can see that the first matching rule is a rule with action type `IPSec_Silver` (which means the tunnel will be secured), the Tunnel ID is `Y0` (which means a dynamic tunnel will be created), and that the traffic direction is `Outbound`.

Displaying IKED and dynamic tunnels

In order to check that communication is working under IP security, try to use **ftp** or **ping** commands. If you have more than one TCP/IP stack in your z/OS image, run **ftp** in the following format:

```
ftp -p TCPIPA 192.168.1.40
```

In this example, we run the command from SC32, using stack TCPIPA. If the **ftp** command completed successfully, use the following command to display the IKED tunnel:

```
ipsec -p TCPIPA -k display
```

The output from this command can be seen in Example 9-21.

Example 9-21 ipsec active IKED tunnels display in SC33, after ftp to SC30

```
CS V1R9 ipsec  Stack Name: TCPIPA  Wed Sep 26 15:15:02 2007
Primary:  IKE tunnel      Function: Display           Format:   Detail
Source:   IKED            Scope:    Current           TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          All_Traffic_Silver~5
KeyExchangeActionName:        All_Traffic_Silver
LocalEndPoint:                10.1.1.50
LocalIDType:                  IPV4
LocalID:                      10.1.1.50
RemoteEndPoint:               192.168.1.40
RemoteIDType:                 IPV4
RemoteID:                     192.168.1.40
```

```
ExchangeMode:                Main
State:                       DONE
AuthenticationAlgorithm:     Hmac_Sha
EncryptionAlgorithm:         DES
DiffieHellmanGroup:          1
AuthenticationMethod:        PresharedKey
InitiatorCookie:             0XF4753A05E16F3B35
ResponderCookie:             0XE015F06F925C2EB8
Lifesize:                    0K
CurrentByteCount:            312b
Lifetime:                    480m
LifetimeRefresh:             2007/09/26 21:10:38
LifetimeExpires:             2007/09/26 23:13:43
Role:                        Initiator
AssociatedDynamicTunnels:    1
NATTSupportLevel:            None
NATInFrntLclScEndPnt:        No
NATInFrntRmtScEndPnt:        No
zOSCanInitiateP1SA:          Yes
AllowNat:                    No
RmtNAPTDetected:             No
RmtUdpEncapPort:             n/a
*******************************************************************************
1 entries selected
```

Note: The **ipsec** command has three main display functions, which can be used to check the phases of IKED security association:

- To see the tunnels created in phase one of IKED security association (IKE tunnels), use **ipsec -k display** to see the IKED tunnels.
- To see the tunnels created in phase two of IKED security association (IPSec tunnels), use **ipsec -y display** to see dynamic tunnels, or use **ipsec -m display** to see manual tunnels.

For more information about the **ipsec** command, refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Use the following command to display the active tunnel:

```
ipsec -p TCPIPA -y display
```

The output from this command can be seen in Example 9-22.

Example 9-22 IPSec active dynamic tunnels display on SC32, after ftp to SC33

```
CS V1R9 ipsec  Stack Name: TCPIPA  Wed Sep 26 15:18:12 2007
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                    Y2
ParentIKETunnelID:           K1
VpnActionName:               IPSec__Silver
LocalDynVpnRule:             n/a
State:                       Active
HowToEncap:                  Transport
LocalEndPoint:               10.1.1.50
RemoteEndPoint:              192.168.1.40
LocalAddressBase:            10.1.1.50
LocalAddressPrefix:          n/a
```

```
LocalAddressRange:            n/a
RemoteAddressBase:            192.168.1.40
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              1188572426
 AuthOutboundSpi:             3684346584
HowToEncrypt:                 DES
 EncryptInboundSpi:           1188572426
 EncryptOutboundSpi:          3684346584
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              11
OutboundBytes:                388
InboundPackets:               9
InboundBytes:                 552
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2007/09/26 18:02:20
LifetimeExpires:              2007/09/26 19:13:43
CurrentTime:                  2007/09/26 15:18:12
VPNLifeExpires:               2007/09/27 15:13:43
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
*******************************************************************************

1 entries selected
```

9.5.4 Working with the z/OS CS Network Management Interface

The z/OS Communications Server includes a Network Management Interface API (NMI) that network management products can take advantage of to provide simple problem determination for complex areas. OMEGAMON® XE for Mainframe Networks represents one of these network management products.

The IKE daemon implements an AF_UNIX listening socket that accepts connections, and uses a request/response model for providing IPSec management data and control. Consequently, IKED must be running in order to make use of this NMI service. Figure 9-37 on page 423 shows the running Policy Agent environment that provides information about IPSec through the **ipsec** command. It also depicts the NMI that interfaces with the IP Security Monitor application. The network operations staff or the network management user uses this information by accessing a GUI at a workstation.

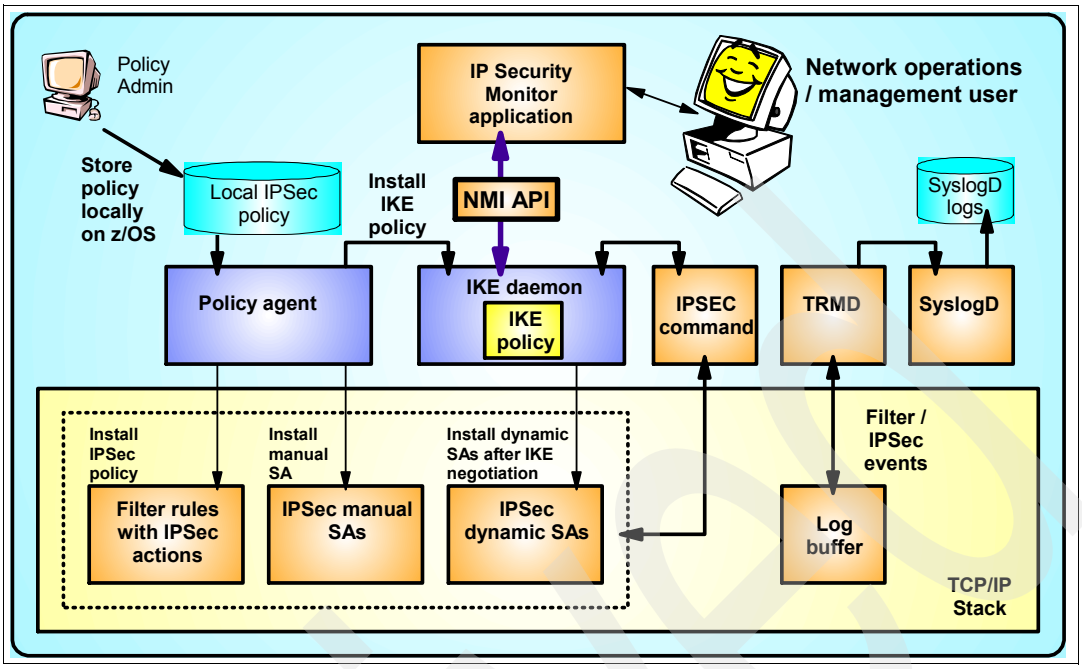


Figure 9-37 IPSec network management interface support

Data similar to what can be retrieved using the **ipsec** command is available over the IP Security NMI interface:

- IP filtering rules and statistics
- IKE phase 1 SA information and status
- IKE phase 2 SA information and status
- Manual SA information and status
- Port translation data

All IPSec NMI data and actions are grouped by individual TCP/IP stack. Almost all requests pertain only to a single TCP/IP stack. These requests are grouped into display requests and control requests. Display requests return various global settings or statistics, or particular information about the IP filters, IP tunnels, IKE tunnels, and so on.

For example, the display requests of the NMI provide the following information that a network management tool may choose to include:

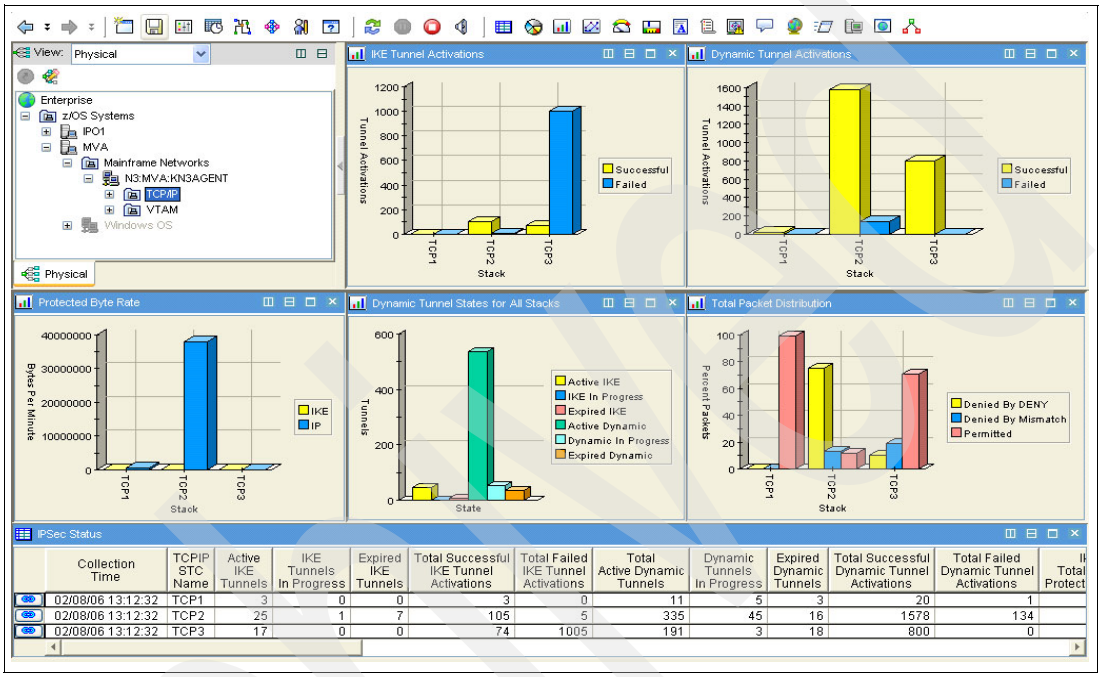
- Stack information (global IPSec configuration settings)
- Summary statistics (various counters and statistics for TCP/IP and IKED)
- Current IP filters (either default or policy filters)
- Default IP filters (defined in the TCP/IP profile)
- Policy IP filters (defined using the Policy Agent)
- Port translation information
- Manual IP tunnels
- Dynamic IP tunnels (information known to the TCP/IP stack)
- Dynamic IP tunnels (information known to IKED)
- IKE tunnels (with or without associated IP tunnels)
- List of IP interfaces

The control requests of the NMI provide the following information that a network management tool may choose to include:

- Activate or deactivate manual tunnels
- Activate, deactivate, or refresh IP tunnels

- Deactivate or refresh IKE tunnels
- Load default IP filters or policy IP filters

IPSec provides an NMI API that OMEGAMON XE exploits. Although the **`ipsec`** command is available to provide display output and to manage system information for Integrated IPSec, the NMI for IPSec delivers these capabilities to OMEGAMON XE. The system programmer retrieves or acts on this information using a GUI at a workstation. Figure 9-38 shows an IPSec management display from OMEGAMON XE for Mainframe Networks.



Collection Time	TCPIP STC Name	Active IKE Tunnels	IKE Tunnels In Progress	Expired IKE Tunnels	Total Successful IKE Tunnel Activations	Total Failed IKE Tunnel Activations	Total Active Dynamic Tunnels	Dynamic Tunnels In Progress	Expired Dynamic Tunnels	Total Successful Dynamic Tunnel Activations	Total Failed Dynamic Tunnel Activations	Total Protect
02/08/06 13:12:32	TCP1	3	0	0	3	0	11	5	3	20	1	
02/08/06 13:12:32	TCP2	25	1	7	105	5	335	45	16	1578	134	
02/08/06 13:12:32	TCP3	17	0	0	74	1005	191	3	18	800	0	

Figure 9-38 IPSec Status Workspace – the dashboard

Authorization to use the NMI

SAF authorization in the SERVAUTH class to the resource EZB.NETMGMT.system.stack.IPSEC.type is required for most request types. Type represents DISPLAY or CONTROL. If the authorization does not exist, then only superusers or users permitted to BPX.SUPERUSER in the FACILITY class are permitted to request data for a given stack.

9.6 Additional information

Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding IPSec configuration.

Refer to *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787, for additional information about authorization.

10

Network Address Translation traversal support

This chapter discusses the support for an IP Security (IPSec) tunnel that traverses a Network Address Translation (NAT) or Network Address Port Translation (NAPT) device.

We provide a high-level overview of NAT and NAPT, explain why NAT and NAPT pose problems for IPSec, and describe the support for IPSec tunnels that traverse a NAT or NAPT device.

We discuss the solution to these problems and demonstrate the support in our environment by showing configuration examples.

Note: z/OS Communications Server provides NAT traversal support for IPv4 only.

The following topics are discussed in this chapter.

Section	Topics
10.1, “Network Address Translation” on page 426	Basic functionality of NAT and Network Address Port Translation (NAPT)
10.2, “IPSec and NAT incompatibilities” on page 428	Brief description of the incompatibilities between IPSec and NATT
10.3, “NAPT traversal support for integrated IPSec/VPN” on page 428	The solution, testing environment, configuration examples, and problem determination suggestions

10.1 Network Address Translation

Network Address Translation (NAT) is a method defined in RFC 3022. NAT, as the name suggests, is a way of translating (or changing) a hosts source IP address into a different IP address on an outbound packet. In order to allow two-way communications, the same translation is performed on an inbound packet. Thus, a NAT device acts as a router between two networks with the ability to translate IP addresses and ports. When connecting two separate networks, there will be a designated *inside* and *outside* network. Generally, NAT devices will perform port translation as well.

With the increase of Internet addresses worldwide, it is becoming more difficult for enterprises to assign globally unique IP addresses to each host. There simply are not enough unique IP addresses to ensure secure and accurate data transmission from site to site or company to company. By using NAT, a single worldwide Internet address can be use to represent a very large private network. As a result, NAT devices are fast becoming a way for large, multisite enterprises to avoid IP address conflicts.

A second benefit is one of security: by using NAT, the real IP address assigned to a host is hidden from the Internet. By hiding the real IP address of a host, it becomes more difficult to access or harm the host behind the NAT device.

NAT itself provides a simple IP address conversion table between a private IP address and a public IP address. After the IP address conversion is done, IP checksum and TCP checksum are recalculated and updated.

NAT encompasses the following IP mapping techniques:

- One-to-one address translation (static or dynamic NAT)
- Network Address Port Translation (NAPT)

One-to-one NAT

An internal-external IP address mapping is maintained by the NAT device. IP addresses are translated, but ports are unchanged. The mapping can be static or dynamic. For a static mapping, there is a definition in the NAT that always translates IP address *x.x.x.x* to IP address *y.y.y.y*. An outbound packet is not needed to establish the mapping. For a dynamic mapping, the NAT has a pool of IP addresses that are assigned as needed, therefore IP address *x.x.x.x* might be mapped to IP address *y.y.y.y* one time, and to IP address *z.z.z.z* at another time. The mapping is established when an outbound packet is processed.

Figure 10-1 shows static NTAT.

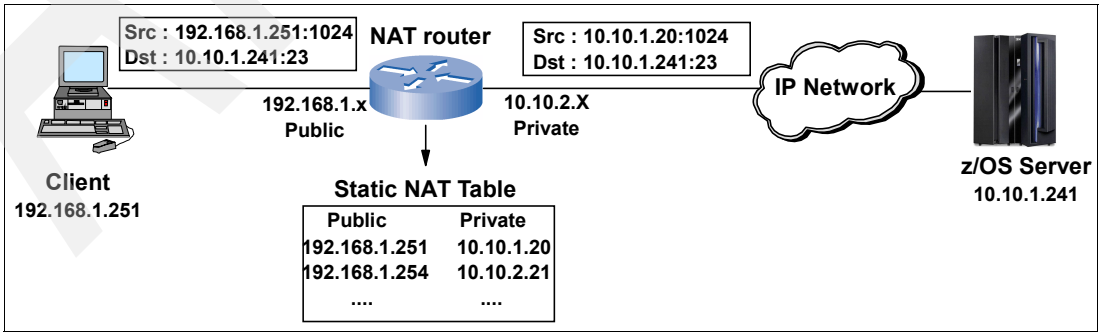


Figure 10-1 Static NAT

Network Address Port Translation

The Network Address Port Translation (NAPT) technique uses multiple internal IP addresses that are translated into a single public IP address. As part of this translation process, the TCP and UDP ports in the packets are translated. NAPT is sometimes referred to as *port address translation (PAT)* or *IP masquerade*.

The mapping is typically dynamic and established in the NAPT when an outbound packet is processed. For example, the NAPT might create a mapping for internal address *x.x.x.x* port *y* to external address *a.a.a.a* port *b*, and a mapping for internal address *z.z.z.z* port *v* to external address *a.a.a.a* port *c*. When only one client behind a NAPT has negotiated a security association, it is not always possible for the remote peer to detect whether the public address is being used for one-to-one address translation or for NAPT.

When multiple clients have active security associations, the remote peer can detect when port translation is being performed. The terms “in front of” and “behind” are used to convey which IP address a NAT is translating. When a NAT is said to be in front of a client, it means that the client’s address will be translated by the NAT. When a server is said to be behind a NAT, it means that server’s address will be translated by the NAT.

Note: NAPT translation is by far the most commonly found “NAT”. It is implemented in most home-use firewall and router devices. Quite often, the term NAT is used when in fact the term NAPT should be used. It is common enough that the term NAT is considered to be synonymous with NAPT in everyday usage.

In this chapter we distinguish these two terms, when appropriate.

In our NAPT configuration example shown in Figure 10-2, clients communicate with a TELNET server through a NAT router. All clients are communicating with only one address called the inside global address as displayed in the router’s NAT table. This registered IP address is used by users who come from the public network to reach the z/OS TELNET server in the private network.

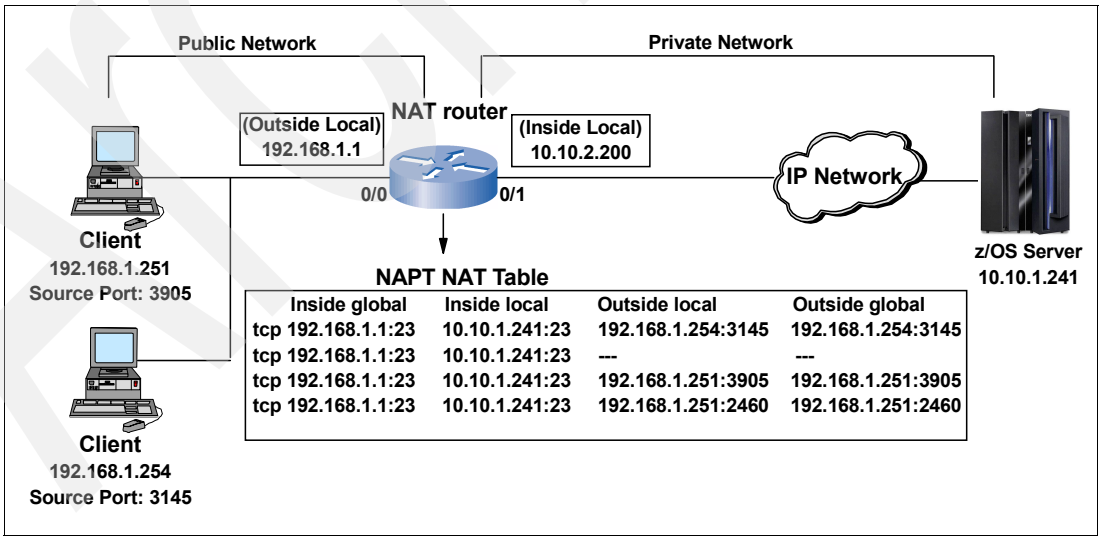


Figure 10-2 NAPT configuration example

Example 10-1 on page 428 shows the results from the corresponding z/OS “Netstat Telnet” display command for the client connections to port 23 on the z/OS TELNET server.

Example 10-1 z/OS Telnet display

```
Internal Telnet Server Status:
Conn     State    BytesIn    BytesOut   ApplName LuName
----     -----    -------    --------   -------- ------
00009902 Establsh 0000000032 0000000739          SC31TS02
  Foreign socket: ::ffff:192.168.1.251..3905
00009900 Establsh 0000000032 0000000739          SC31TS01
  Foreign socket: ::ffff:192.168.1.254..3145
```

10.2 IPSec and NAT incompatibilities

This section discusses the incompatibilities between IPSec and NAT functions.

Background

There are inherent incompatibilities between IPSec and NAT functions. RFC 3715, IPSec-Network Address Translation (NAT) Compatibility Requirements, provides a description of the problems that arise when IPSec is used to protect traffic that traverses a NAT.

One basic problem is that when IPSec security associations traverse a NAT, the NAT is unable to update IP addresses and checksums that are part of the encapsulated data (encrypted, authenticated, or both). RFCs 3947 and 3948 define mechanisms that enable specific uses of IPSec to traverse one or more NAT devices.

- RFC 3947, Negotiation of NAT-Traversal in the IKE, allows an Internet Key Exchange (IKE) daemon to detect when one or more NATs are being traversed.
- RFC 3948, UDP Encapsulation of IPSec ESP Packets, defines two IPSec encapsulation modes: UDP-Encapsulated-Tunnel mode, and UDP-Encapsulated-Transport mode. These modes facilitate the traversal of IPSec traffic through a NAT by encapsulating ESP packets within a UDP packet.

UDP-Encapsulated-Tunnel mode and UDP-Encapsulated-Transport mode security associations are negotiated by IKE when NAT traversal is enabled and a NAT is detected. Manually configured security associations do not support UDP-Encapsulated-Tunnel mode or UDP-Encapsulated-Transport mode.

It should be noted that the NAT traversal support, as defined by the Internet Engineering Task Force (IETF), transmits internal IP addresses to its IPSec peer. These internal addresses are not exposed on the external network. However, the internal addresses are available for display at the remote security endpoint. If you are considering using this support, evaluate whether transmitting internal IP addresses to an IPSec peer is acceptable from a security policy perspective.

Important: UDP encapsulation is *not* encapsulating a UDP packet. UDP encapsulation is inserting a UDP header between the IP header and the ESP header. The payload data can have a TCP, UDP, or other transport header.

10.3 NAPT traversal support for integrated IPSec/VPN

z/OS Communications Server extends IP security support to protect traffic that traverses a NAPT device. A NAPT device translates multiple internal IP addresses to a single public address, and translates the TCP or UDP port to make the connection unique. There are

inherent incompatibilities between IPSec and NAT functions. Problems occur when IPSec is used to protect traffic that traverses a NAT or NAPT device. NAT traversal support as defined by the IETF in RFC 3947 (Negotiation of NAT-traversal in the IKE) and RFC 3948 (UDP encapsulation of IPSec ESP packets) defines mechanisms that allow specific uses of IPSec to traverse one or more NAT or NAPT devices. IP security provides NAT traversal support for a defined group of configurations where an IPSec security association (SA) traverses a NAPT device.

Restrictions

In configurations where the remote security endpoint is behind a NAPT device, the following SA restrictions exist:

- The remote security endpoint must initiate the SA.
- The remote data endpoint must initiate the data.
- Only TCP, UDP, and Internet Control Message Protocol (ICMP) traffic is supported. ICMP traffic has limited support.
- In a sysplex-wide security association (SWSA) environment, the SA can be distributed only to targets at release level V1R8 or later that have IP security configured.
- In an SWSA environment, SWSA takeover functions are not available for the SA.

10.3.1 Enabling NAPT traversal support for IPSec

The objective was to successfully establish an IPSec dynamic tunnel from a Windows client, via a Cisco NAPT device to a TELNET server running on the z/OS mainframe. In order to test and verify the support, we made the following configuration changes:

- IBM Configuration Assistant: Enabled NAT traversal support for TCPIPA
- Cisco router: Configured NAT and Port Address Translation (PAT) for port 23
- Windows XP: Installed and configured IPSec

Figure 10-3 represents our test environment.

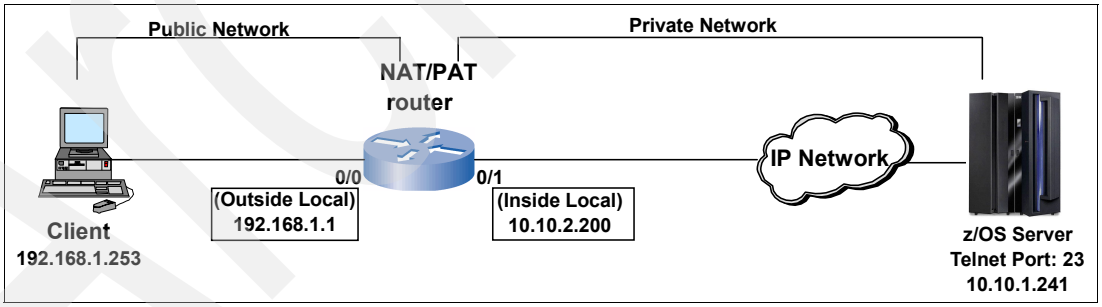


Figure 10-3 NAPT traversal diagram

Next, we explain the configuration changes we made to our test environment to successfully establish the IPSec dynamic tunnel via the NAPT traversal device.

IBM Configuration Assistant changes

In this section, we demonstrate the additional changes required on the Configuration Assistant graphical user interface (GUI) in order to enable the NAPT support.

Important: Before making these changes, refer to and complete the IPSec for Windows XP implementation in Appendix C, “Configuring IPSec between z/OS and Windows” on page 711.

1. From the main Configuration Assistant panel, we clicked **TCPIPA** -> **IPSec: Stack Level Settings**, as shown in Figure 10-4.

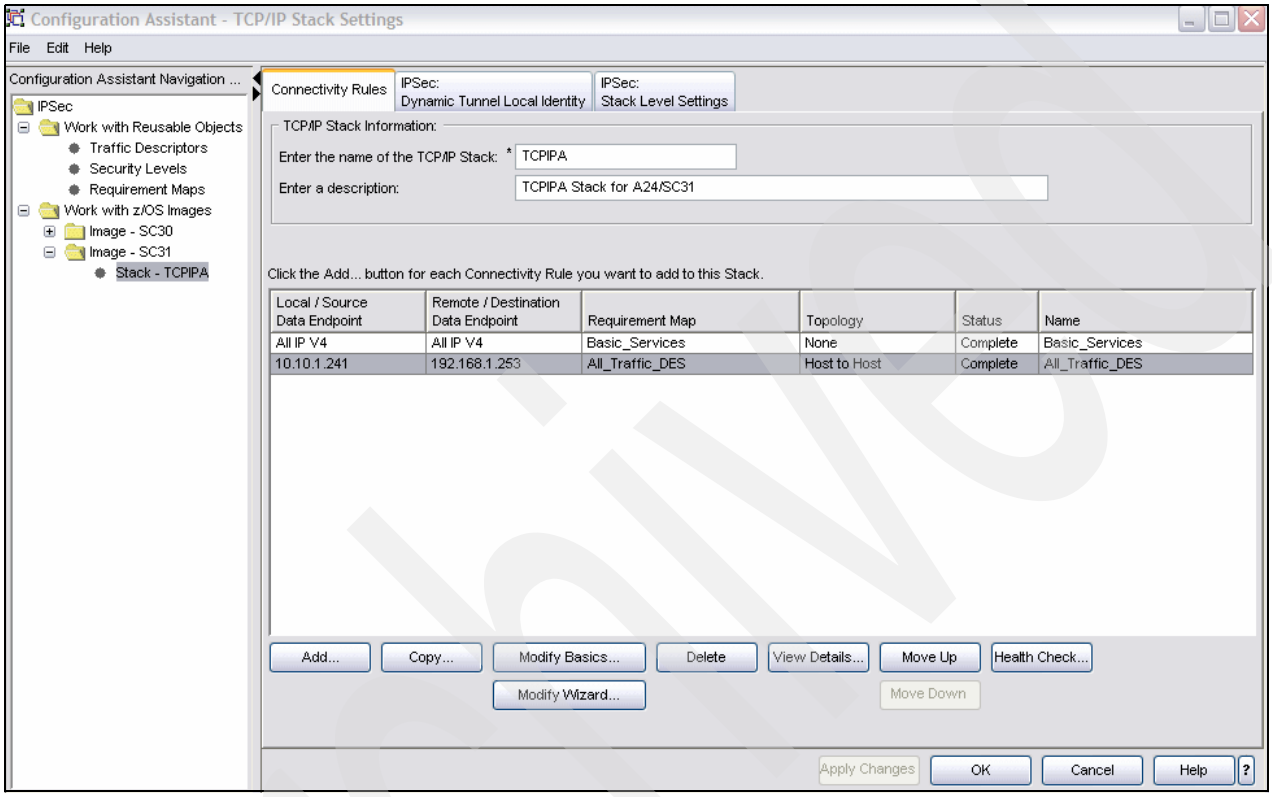


Figure 10-4 TCP/IP Stack Settings

2. From the IPSec: Stack Level Settings panel, we selected **NAT Defaults** -> **Allow** -> **Apply**, as shown in Figure 10-5.

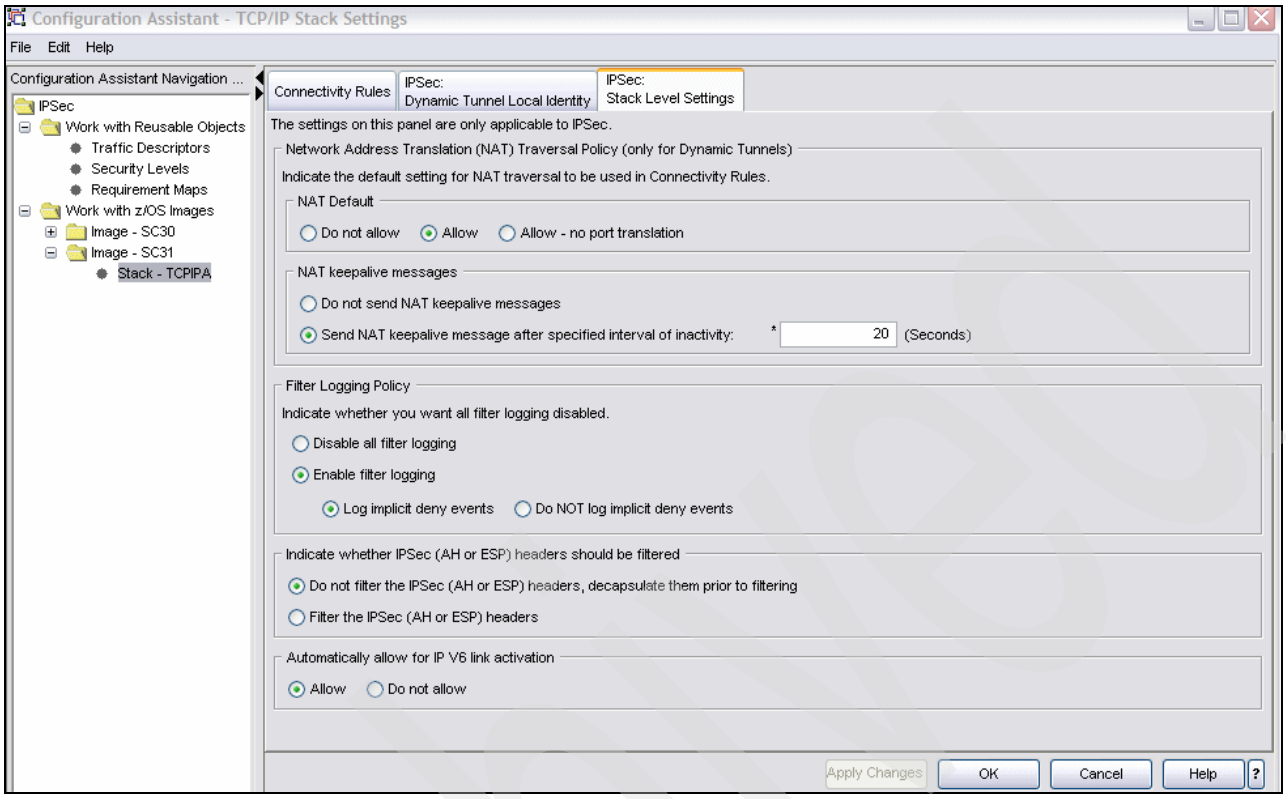


Figure 10-5 IPSec: Stack Level Settings

3. From the TCP/IP Stack Settings panel, we clicked rule -> **10.10.1.241 to 192.168.1.253** -> **Modify Basics**, as shown in Figure 10-6.

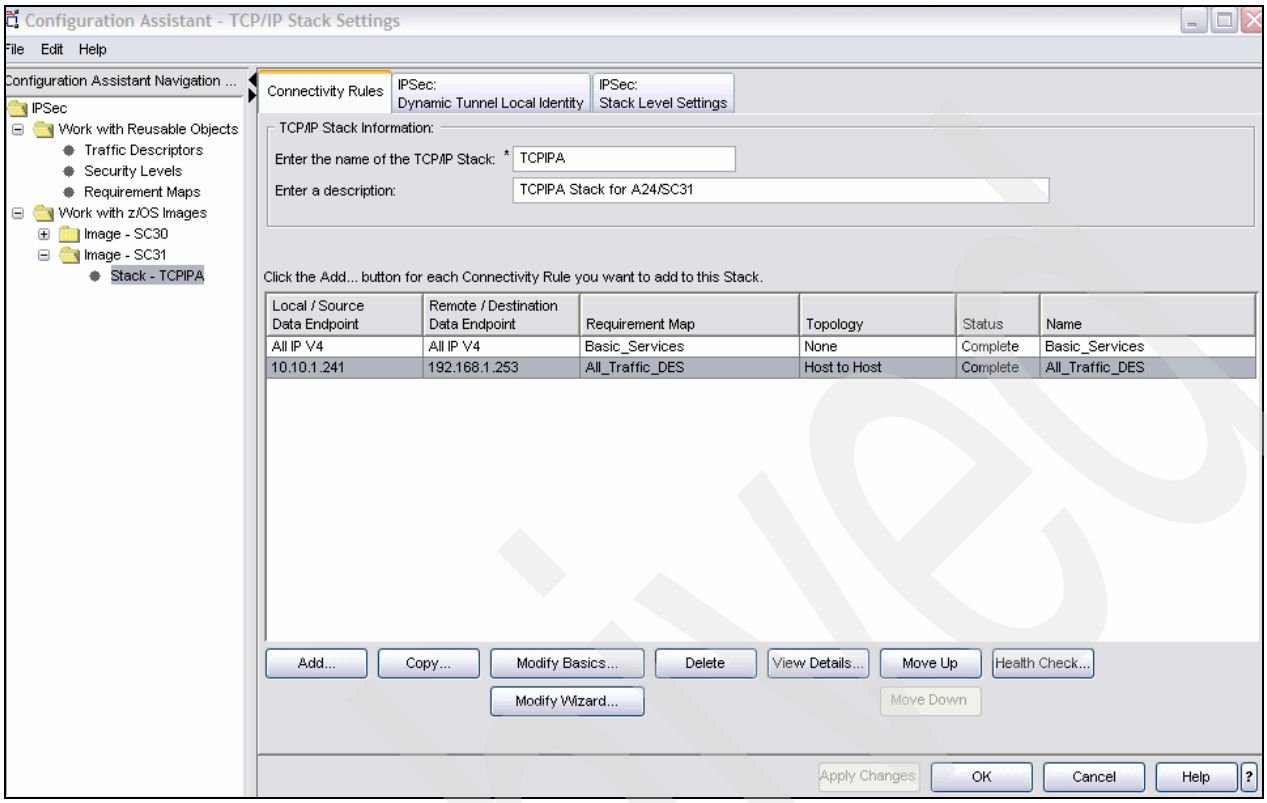


Figure 10-6 TCP/IP Stack Settings

4. From the Connectivity Rule panel (Figure 10-7), we clicked **Additional Settings**.

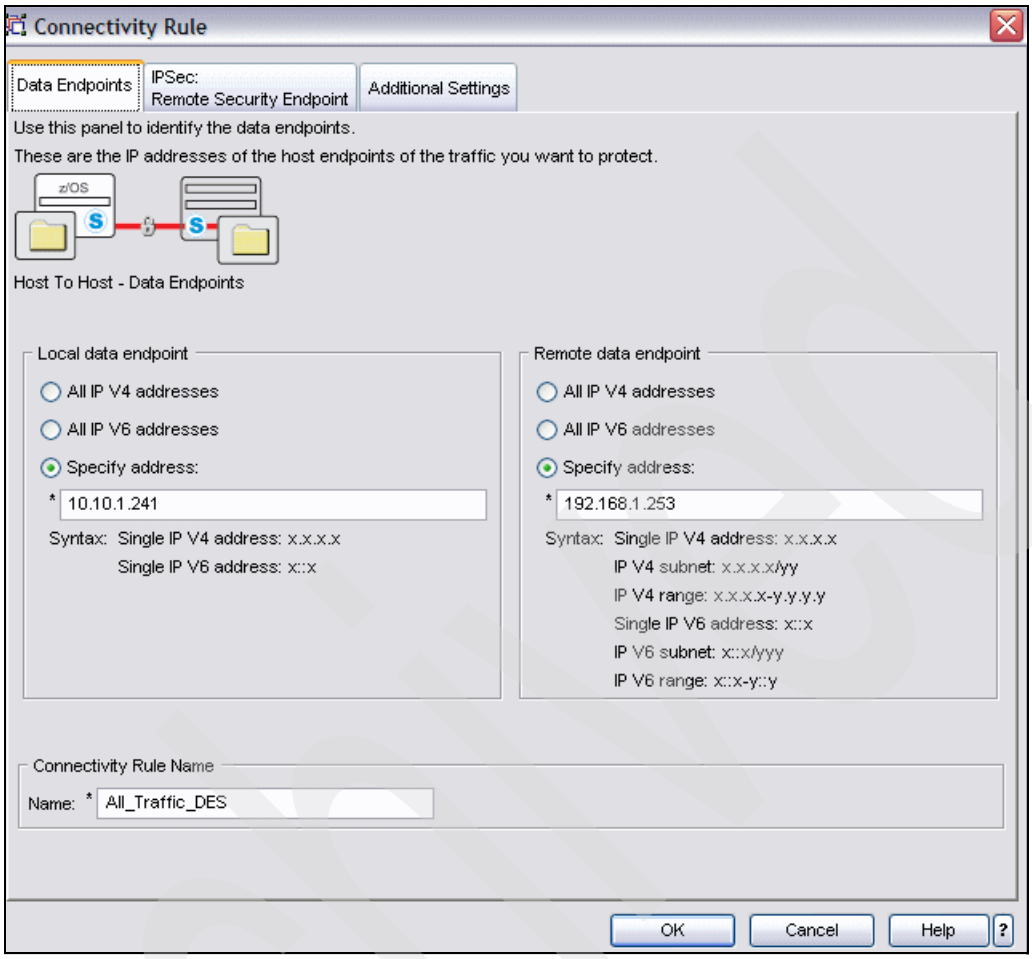


Figure 10-7 Connectivity Rule

5. From the Additional Settings panel, we clicked **Advanced**, as shown in Figure 10-8.

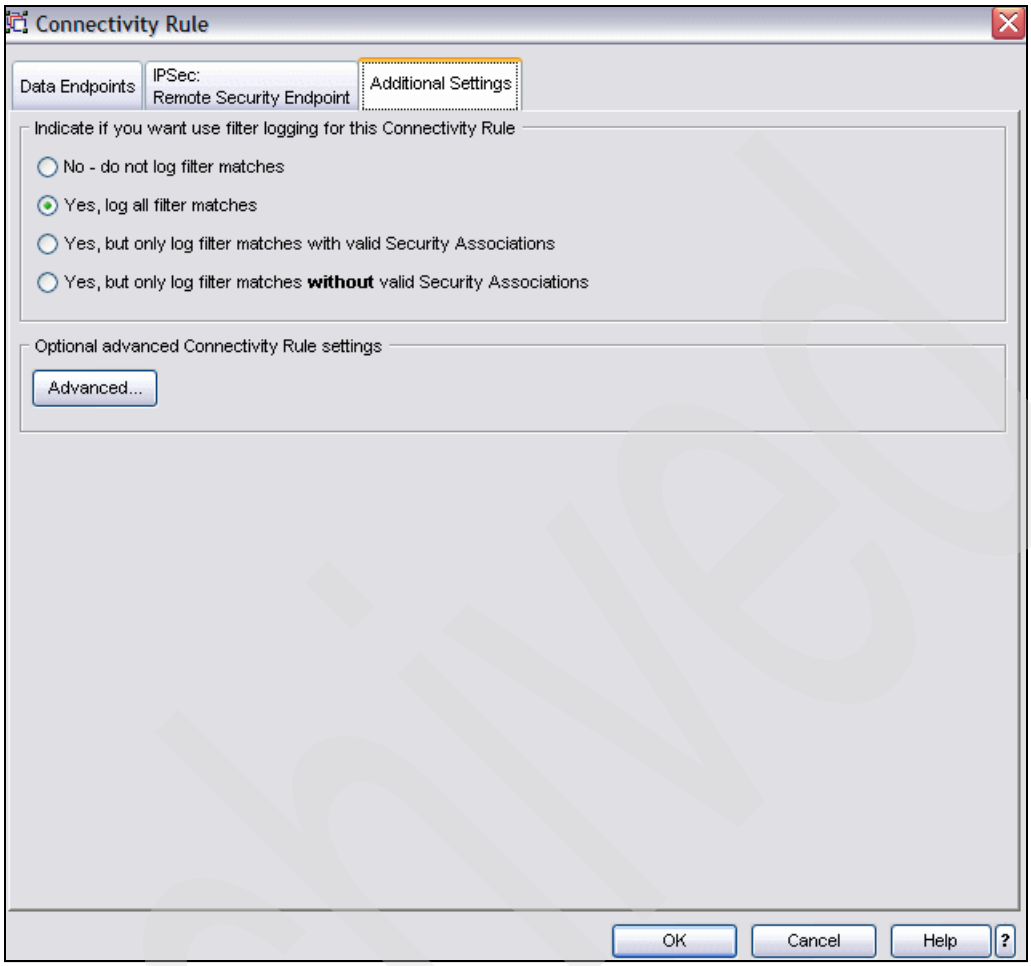


Figure 10-8 Connectivity Rule: Additional Settings

6. From the Advanced Connectivity Rule Settings panel (Figure 10-9), we clicked **IPSec: Dynamic Tunnels: Key Exchange Settings**.

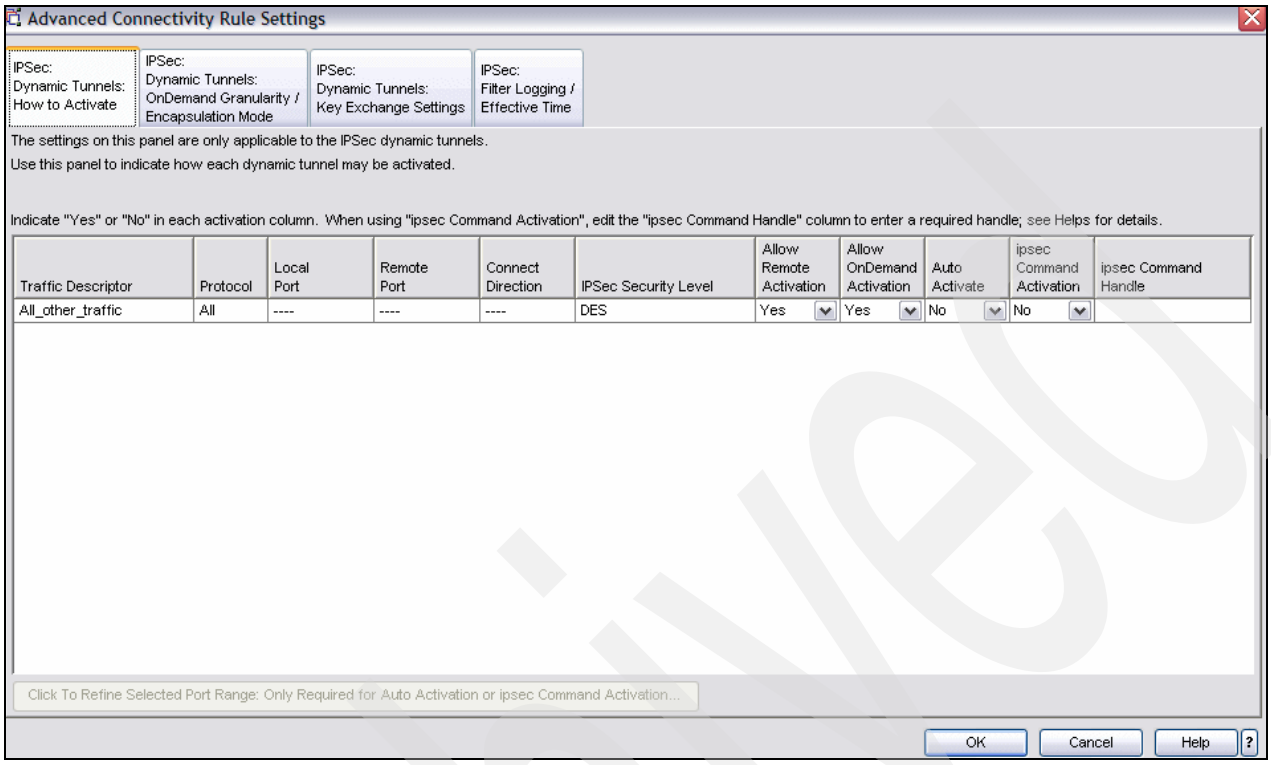


Figure 10-9 Advanced Connectivity Rule Settings

7. From the IPSec: Dynamic Tunnels: Key Exchange Settings for the Network Address Translation NAT Traversal field, we selected the **Allow** button, as shown in Figure 10-10.

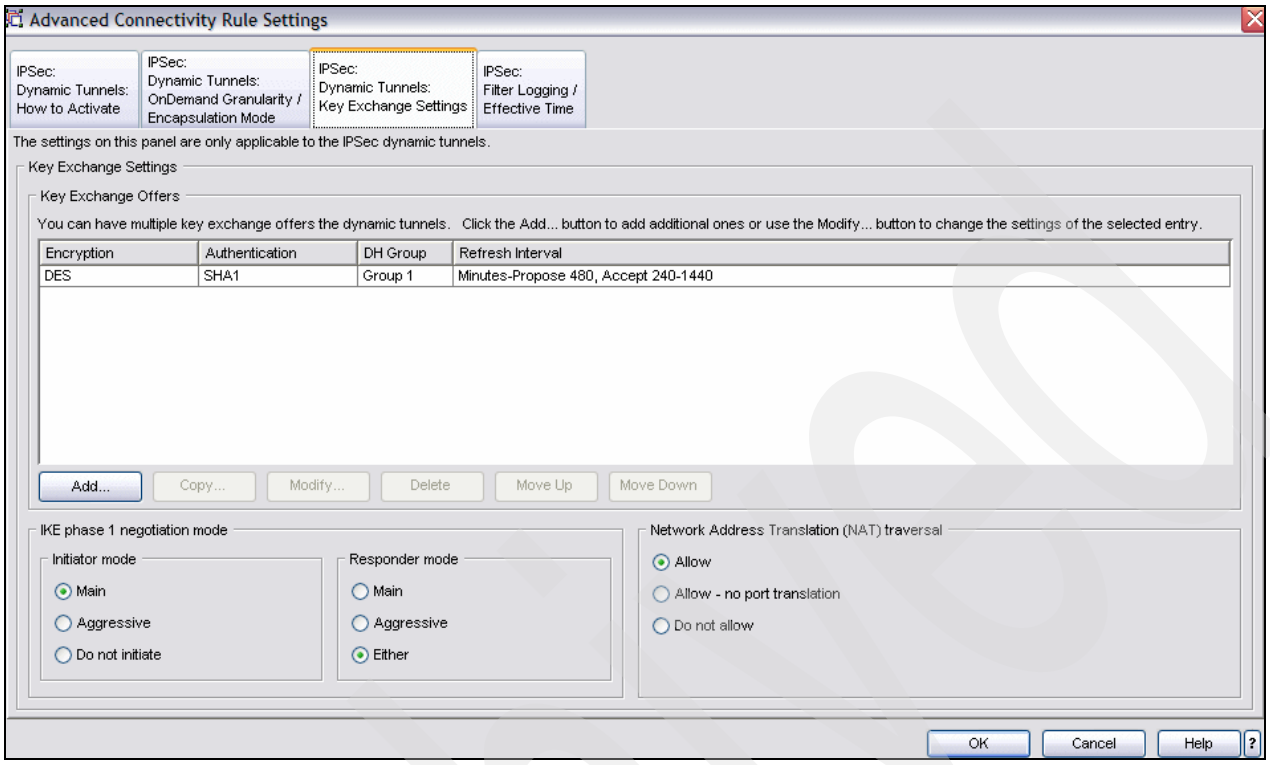


Figure 10-10 IPSec: Dynamic Tunnels: Key Exchange Settings

8. We transferred the new IPSec rules to the PAGENT server on z/OS, as shown in Figure 10-11.

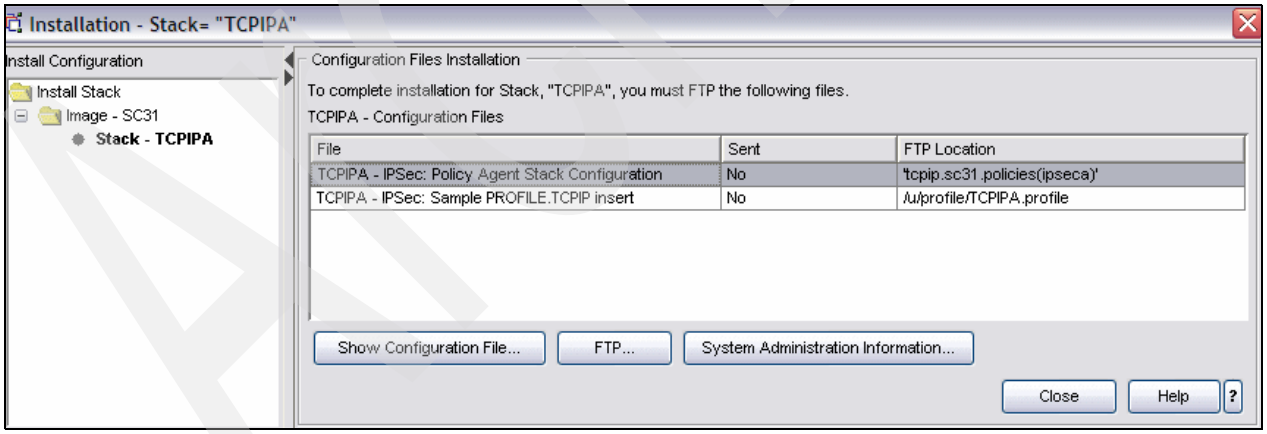


Figure 10-11 Configuration Files Installation

Cisco router configuration

For our NAPT device, we used a Cisco 2600 series router on which we configured a Static Port Address Translation (PAT) feature for ports 23, 500, and 4500. Apart from the normal Telnet port: 23, we also had to configure ports 500 and 4500 for the IKE and IPSec Dynamic tunnel. Our configuration is shown in Example 10-2.

Example 10-2 Cisco NAPT configuration

```
provrtr01#sh run
Building configuration...

Current configuration : 1409 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname provrtr01
!
boot system flash c2600-ik9s-mz.122-15.t17.bin
boot system flash c2600-is-mz.122-11.t3.bin
logging queue-limit 100
enable secret 5 $1$1WC1$Vhn3KRUVOba5gobLe2Jbb.
enable password itsc
!
ip subnet-zero
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 ip nat outside
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
!
interface FastEthernet0/1
 ip address 10.10.2.200 255.255.255.0
 ip nat inside
 duplex auto
 speed auto
!
ip nat inside source static tcp 10.10.1.241 23 192.168.1.1 23 extendable
ip nat inside source static udp 10.10.1.241 500 192.168.1.1 500 extendable
ip nat inside source static udp 10.10.1.241 4500 192.168.1.1 4500 extendable
ip nat outside source list 1 interface FastEthernet0/1
ip http server
no ip http secure-server
ip classless
ip route 10.10.0.0 255.255.0.0 10.10.2.1
!
access-list 1 permit 192.168.1.0 0.0.0.255
!
snmp-server community public RO
snmp-server enable traps tty
call rsvp-sync
!
```

```
line con 0
line aux 0
line vty 0 4
 password swj43r
 login
!
!
end
```

Windows XP configuration

We performed the steps in this section after completing the initial Windows XP IPSec configuration described in Appendix C, “Configuring IPSec between z/OS and Windows” on page 711. However, because our IPSec tunnel traversed a NAT device, we also had to enable NAT traversal support for Windows and therefore we discuss it here.

Important: If these actions are not executed, the IPSec NAT traversal tunnel will not establish successfully.

1. We had Windows XP Service Pack 2 installed and due to a change in the default behavior for Windows IPSec NAT traversal, we had to make a modification to the Windows Registry. We recommend that you do a search for “NAT Traversal” on the support.microsoft.com Web site to determine whether your system requires similar changes. The details and description for our change can be found at the following link:

   http://support.microsoft.com/kb/885407/en-us

2. We also changed the Destination address in “Set up a Windows IPSec policy” on page 721 to reflect our Inside global IP address 198.168.1.1. This address corresponds to the IP address configured on Interface FastEthernet0/0 of the Cisco device shown in Example 10-3 on page 439.

10.3.2 Testing and verification

To verify our configuration and the functionality of the IPSec NAT traversal support for NAPT, we established a Telnet connection from our Windows client 192.168.1.253 to the outside global address 192.168.1.1 port 23.

The outside global address in return gets translated by the PAT on the Cisco router to 10.10.1.241 port 23, which represent the z/OS server. This action created a dynamic IKE IPSec tunnel between the Windows client and the z/OS server, which confirmed the functionality of the IPSec NAPT traversal support.

In the following steps, we provide relevant examples and descriptions of the snapshots that were taken during this verification stage after we successfully established the Telnet connection from the Windows client to the z/OS server:

1. The MSG10 on the client (Figure 10-12) was the first indication that we managed to establish the IPSec tunnel.

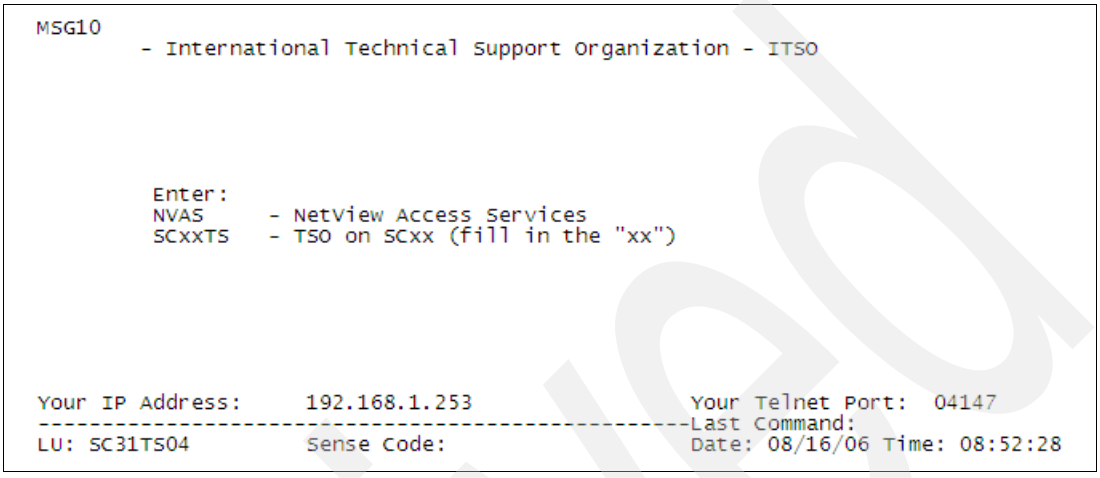


Figure 10-12 MSG10 on the Windows client

2. We logged on to the Cisco router and issued the **show ip nat translation** command. The output in Example 10-3 confirmed two very important points:
   - Our NAPT configuration was done correctly and performed as intended.
   - Further confirmation that the IPSec tunnel was successfully established over the NAPT device. Port 4500 represents the IPSec dynamic tunnel.

Example 10-3 Cisco NAT display

```
provrtr01#sh ip nat translation
Pro Inside global      Inside local       Outside local      Outside global
tcp 192.168.1.1:23     10.10.1.241:23     ---                ---
udp 192.168.1.1:4500   10.10.1.241:4500   ---                ---
udp 192.168.1.1:500    10.10.1.241:500    ---                ---
udp 192.168.1.1:4500   10.10.1.241:4500   192.168.1.253:4500 192.168.1.253:4500
provrtr01#
```

3. On the z/OS server, we issued a Netstat Telnet command, which confirmed that we had a connection established on port (04147) with the Windows client. The port number corresponds with the port number displayed on the MSG10; see Example 10-4.

Example 10-4 Netstat Telnet display on z/OS

```
Internal Telnet Server Status:
 Conn     State    BytesIn    BytesOut   ApplName LuName
 ----     -----    -------    --------   -------- ------
 00000130 Establsh 0000000032 0000000739          SC31TS01
   Foreign socket: ::ffff:192.168.1.253..04147
 ***
```

4. From the z/OS OMVS shell, we issued the **ipsec** command to verify both the IKE and Dynamic tunnel. First, we issued the **ipsec -p tcpipa -k** command and then the **ipsec -p tcpipa -y** command. The results from these commands are shown in Example 10-5 and Example 10-6, respectively.

Example 10-5 Output from the ipsec -p tcpipa -k display

```
Primary:  IKE tunnel      Function: Display           Format:   Detail
Source:   IKED            Scope:    Current           TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          All_Traffic_DES~5
KeyExchangeActionName:        All_Traffic_DES
LocalEndPoint:                10.10.1.241
LocalIDType:                  IPV4
LocalID:                      10.10.1.241
RemoteEndPoint:               192.168.1.253
RemoteIDType:                 IPV4
RemoteID:                     192.168.1.253
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup            1
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0XEBFAC6490364F62A
ResponderCookie:              0XDA471394239D2C08
Lifesize:                     0K
CurrentByteCount:             544b
Lifetime:                     480m
LifetimeRefresh:              2006/08/16 15:05:08
LifetimeExpires:              2006/08/16 16:48:49
Role:                         Initiator
AssociatedDynamicTunnels:     1
NATTSupportLevel:             D2RFC
NATInFrntLclScEndPnt:         Yes
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     Yes
RmtNAPTDetected:              No
RmtUdpEncapPort:              4500
```

Example 10-6 shows the output from the **ipsec -p tcpipa -y display** command.

Example 10-6 Output from the ipsec -p tcpipa -y display

```
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y2
ParentIKETunnelID:            K1
VpnActionName:                DES
LocalDynVpnRule:              n/a
State:                        Active
HowToEncap:                   Transport
LocalEndPoint:                10.10.1.241
RemoteEndPoint:               192.168.1.253
LocalAddressBase:             10.10.1.241
LocalAddressPrefix:           n/a
LocalAddressRange:            n/a
```

```
RemoteAddressBase:            192.168.1.253
RemoteAddressPrefix:          n/a
RemoteAddressRange:           n/a
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Sha
 AuthInboundSpi:              708956256
 AuthOutboundSpi:             2506164033
HowToEncrypt:                 DES
 EncryptInboundSpi:           708956256
 EncryptOutboundSpi:          2506164033
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              27
OutboundBytes:                3127
InboundPackets:               37
InboundBytes:                 986
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             0b
LifetimeRefresh:              2006/08/16 11:54:38
LifetimeExpires:              2006/08/16 12:48:49
CurrentTime:                  2006/08/16 08:53:27
VPNLifeExpires:               2006/08/17 08:48:22
NAT Traversal Topology:
  UdpEncapMode:               Yes
  LclNATDetected:             Yes
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    No
  RmtIsZOS:                   No
  zOSCanInitP2SA:             Yes
  RmtUdpEncapPort:            4500
  SrcNATOARcvd:               0.0.0.0
  DstNATOARcvd:               0.0.0.0
```

Debugging tools

We used the following commands and tools to gather debugging information, depending on what type of problem we worked on.

- TCP/IP Netstat, Ping, Tracerte and displays
- Omproute (OSPF) d tcpip,tcpipa,omp,ospf,xxxxx displays
- Log files, MVS console log, Syslogd
- 2600 Router, show ip nat translations, debug ip nat detailed

11

Application Transparent Transport Layer Security

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocol technology is used to protect data exchanges between client and server applications. Using TLS/SSL, the client and server can confirm the identity of each other, and their data flows can be encrypted to protect e-commerce transactions and other data as they navigate the network. In this chapter, the terms TLS and SSL are used interchangeably. The TLS/SSL service available under z/OS is called System SSL.

The use of TLS/SSL protocols used to require extensive programming changes to applications within the mainframe environment. With the availability of Application Transparent Transport Layer Security (AT-TLS), you can now deploy TLS encryption without the time and expense of re-coding your applications.

The following topics are discussed in this chapter.

Section	Topic
11.1, “Conceptual overview of AT-TLS” on page 444	The role of AT-TLS in securing socket-based applications
11.2, “REXX socket API support for AT-TLS” on page 448	How to implement AT-TLS for a general application A REXX™ API used as an example
11.3, “Problem determination for AT-TLS” on page 469	AT-TLS problem determination techniques
11.4, “Additional information sources for AT-TLS” on page 470	AT-TLS application restrictions

11.1 Conceptual overview of AT-TLS

Application Transparent TLS (AT-TLS) provides TLS support for existing clear-text (no encryption) socket applications without requiring any application changes; therefore, the term *Application Transparent* applies.

11.1.1 What is AT-TLS

The Transport Layer Security (TLS) protocol provides transport layer security (authentication, integrity, and confidentiality) for a secure connection between two applications. The TLS protocol begins with a handshake, in which the two applications agree on a cipher suite. A cipher suite is comprised of cryptographic algorithms to be used for authentication, session encryption and hashing (digital fingerprinting). After the client and server applications have negotiated a cipher suite, they authenticate each other and generate a session key. The session key is used to encrypt and decrypt all data traffic sent between the client and the server.

Implementing TLS protocols directly into applications (without using AT-TLS) requires modification to incorporate a TLS capable API toolkit. Only the z/OS System SSL toolkit supports RACF keyrings and the associated advantages (user ID mapping, SITE certificates and more).

Note: AT-TLS only supports TCP-based applications; it cannot be used to provide security for UDP-based applications. To provide security for UDP-based applications, consider taking advantage of IP Security (IPSec) support (discussed in Chapter 9, “IP Security” on page 373).

11.1.2 How AT-TLS works

AT-TLS support is policy-driven and is managed by a Policy Agent (PAGENT), as discussed in Chapter 5, “Policy Agent” on page 221. Socket applications continue to send and receive clear text over the socket, but data sent over the network is protected by system SSL. Support is provided for applications that require awareness of AT-TLS for status or to control the negotiation of security.

AT-TLS provides application-to-application security using policies. The policies are defined and loaded into the stack by Policy Agent. When AT-TLS is enabled and a newly established connection is first used, the TCP layer of the stack searches for a matching AT-TLS policy. If no policy is found, the connection is made without AT-TLS involvement. If a policy is found, a sequence like the one illustrated in Figure 11-1 on page 445 is followed.

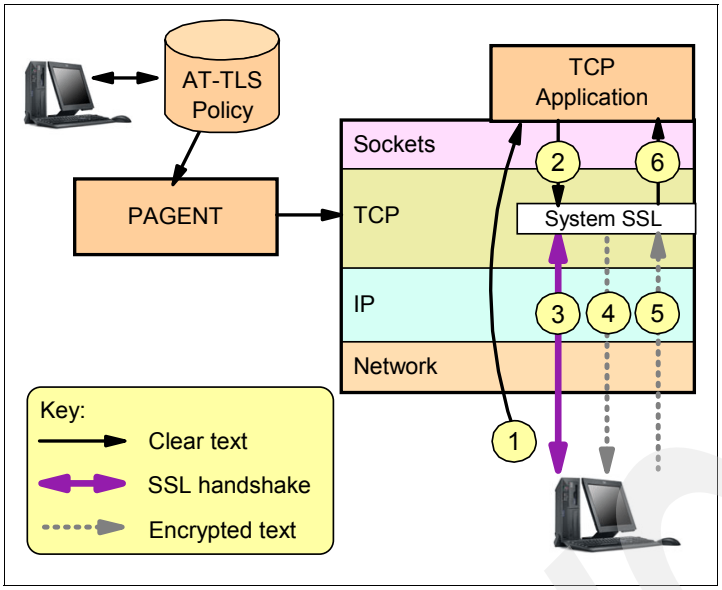


Figure 11-1 AT-TLS basic flow

The flow is:

1. The client connection to the server gets established in the clear (no security, TCP handshake only).
2. The server sends data in the clear and the TCP layer queues it.
3. The TCP layer invokes System SSL to perform an SSL handshake under the identity of the server, using policy information.
4. The TCP layer invokes System SSL to encrypt the queued data and sends it to the client.
5. The client then sends encrypted data and the TCP layer invokes System SSL to decrypt the data.
6. The server receives data in the clear.

All of this is performed transparently to the remote application, as it would have no way of knowing that the handshake and encryption were being done via AT-TLS instead of direct System SSL calls.

When AT-TLS is enabled, statements in the Policy Agent define the security attributes for connections that match AT-TLS rules. This policy-driven support can be deployed transparently underneath many existing sockets, leaving the application unaware of the encryption and decryption being done on its behalf. Support is also provided for applications that need more control over the negotiation of TLS or need to participate in client authentication; however, these applications must be aware of AT-TLS and use IOCTL support (see “Controlling applications” in “AT-TLS application types” on page 446). AT-TLS supports the TLS, SSLv3, and SSLv2 protocols.

IOCTL support is provided for applications that need to be aware of AT-TLS for status or to control the negotiation of security. TLS can be requested by applications where the application issues AT-TLS API calls to indicate that a connection should start or stop using TLS.

Client identification services are also available for applications where TLS API calls are used to receive user identity information based on X.509 client certificates. SIOCTTLSCTL is an IOCTL specifically available with AT-TLS for applications to control AT-TLS for a connection.

Applications can perform tasks such as initializing a connection (TTLS_INIT_CONNECTION), resetting a connection (TTLS_RESET_CONNECTION), and resetting ciphers, using the SIOCTTLSCTL IOCTL macros. Also, the SIOCTTLSCTL IOCTL macro can stop security on a connection (TTLS_STOP_CONNECTION) and allow both secure and non-secure connections on the same port.

11.1.3 How AT-TLS can be applied

Although AT-TLS can be used to provide security for the majority of applications transparently, some applications need to control the security functions being performed by TCP/IP. This communication between the application and AT-TLS is done through the transparent TLS API using the SIOCTTLSCTL Input/Output Control (IOCTL) macros.

Note: Be careful to coordinate your use of AT-TLS with other application-specific encryption implementations. Otherwise, you could end up encrypting the same data twice, first by the application and then by AT-TLS.

If possible, use AT-TLS as a consistent security solution for all of your TCP-based applications.

Planning considerations for applying general and specific functions of AT-TLS are covered in the following topics:

- AT-TLS application types
- FTP AT-TLS support
- TN3270 AT-TLS support
- Recommended AT-TLS implementations
- General restrictions with AT-TLS

AT-TLS application types

Applications have different requirements concerning security. Some applications need to be aware of when a secure connection is being used. Others may need to assume control if and when a TLS handshake occurs. For this reason, there are different application types supported by AT-TLS. These application types include:

- Not-enabled applications:
  - Pascal API and Web Fast Response Cache Accelerator (FRCA) applications are not supported at all by AT-TLS.
  - By default, when there are no AT-TLS policies in place and an application is considered to be not-AT-TLS enabled. This includes applications that start during the InitStack window and if the policy explicitly says `enabled off`.
  - A third category applies to FTP and Telnet. They can be not-enabled for AT-TLS in the policy agent, but function with their native TLS/SSL capabilities independent of AT-TLS.

  Note: The AT-TLS capabilities for TN3270 and FTP server is now functionally superior to the native TLS/SSL capabilities.

- Basic applications:
  - The AT-TLS policy says `enabled on`.
  - The application is unchanged and unaware of AT-TLS (no AT-TLS IOCTL calls).
- Aware applications
  - The AT-TLS policy says `enabled on`.

  - The application is changed to use the SIOCTTLSCTL IOCTL to extract AT-TLS information.
- Controlling applications
  - The application protocol may negotiate the use of TLS in cleartext prior to starting a secure session.
  - Where the policy says `enabled on` and `ApplicationControlled on`.
  - The application is changed to use SIOCTTLSCTL IOCTL to extract and control AT-TLS.

FTP AT-TLS support

In addition to native TLS support, the FTP server and client can use Application Transparent Transport Layer Security (AT-TLS) to manage TLS security. TLS managed by AT-TLS (TLSMECHANISM ATTLS) supports more security functions than TLS managed natively by the FTP (TLSMECHANISM FTP). Be aware of these AT-TLS capabilities and requirements when planning AT-TLS support for FTP:

- Specify the label of the certificate to be used for authentication instead of using the default.
- Support SSL session key refresh.
- Support SSL sysplex session ID caching.
- Trace decrypted SSL data for FTP in a data trace.
- Receive more detailed diagnostic messages in syslogd.
- There are no restrictions for this function.
- Policy Agent must be active.
- TLS security defined natively to FTP will continue to be available in addition to AT-TLS.

Refer to Chapter 15, “Secure File Transfer Protocol” on page 579, for complete details about implementing TLS and AT-TLS security for the FTP server and client.

TN3270 AT-TLS support

In addition to native TLS support, the TN3270 server can use Application Transparent Transport Layer Security (AT-TLS) to manage secure connections. TLS managed by AT-TLS (TTLSPORT) supports more security functions than TLS managed by the TN3270 (SECUREPORT).

Note: If you have security parameters in a PARMSGROUP statement mapped to host names, you will not be able to emulate that mapping with AT-TLS. If you are not using PARMSGROUP to map to host names, we urge you to migrate to the use of AT-TLS as a consistent solution for all of your TCP applications.

Be aware of these AT-TLS capabilities and requirements when planning AT-TLS support for TN3270:

- Dynamically refresh a key ring
- Support new or multiple key rings
- Specify the label of the certificate to be used for authentication instead of using the default
- Support SSL session key refresh
- Support SSL session reuse
- Support SSL sysplex session ID caching
- Trace decrypted SSL data for Telnet in a data trace
- Receive more granular error messages in syslogd for easier debugging.
- Policy Agent must be active.
- TLS security defined within the TN3270 profile will continue to be available and can be implemented concurrently with AT-TLS.

Refer to Chapter 14, “Telnet security” on page 539, for complete details about implementing TLS and AT-TLS security for the TN3270 server.

Note: The TN3270 client, activated through the TSO TELNET command, does not support any SSL or TLS security protocols. Basic mode is the only method supported by the TN3270 client.

Recommended AT-TLS implementations

AT-TLS is the preferred method of implementing TLS support in order to achieve a consistent solution for *all* of your TCP applications. AT-TLS provides security for your TCP-based applications without the development costs of implementing security directly into your applications. You can use AT-TLS for existing CICS or DB2 applications, for example. Because AT-TLS can be used as a consistent solution across *all* of your TCP-based applications, systems administrators reap the benefits of improved productivity and infrastructure simplification with streamlined management.

Note: Use of application-negotiated SSL/TLS (AUTH command) is recommended by the IETF over the use of *implicit* SSL/TLS (TLSPORT). Implicit TLS only existed as a temporary standard during the evolution of TLS for FTP. Even though z/OS FTP supports implicit SSL/TLS under *both* native TLS and AT-TLS implementations, we recommend avoiding the use of implicit SSL/TLS (TLSPORT) so that FTP can be consistent in how it negotiates the use of SSL/TLS during session setup.

General restrictions with AT-TLS

The following applications will not map to AT-TLS policies and are not supported by AT-TLS.

- Applications using the Pascal API to access TCP/IP
  - Line Print daemon and commands LPD, LPQ, LPRM
  - Simple Mail Transfer Protocol (JES Spool Server)
  - TSO Telnet client
- Web servers using Fast Response Cache Accelerator
- Network administration applications permitted to the EZB.INITSTACK RACF profile. (If you have activated stack initialization protection, then anything that you have permitted to EZB.INITSTACK is presumably something you do not want AT-TLS for anyway.
  - Connections established and mapped prior to the installation of the AT-TLS policy will proceed in cleartext.
  - Connections established and mapped after installation of the AT-TLS policy are subject to the installed policy.

Those applications that are not supported by AT-TLS will be permitted to proceed in cleartext.

Tip: AT-TLS functions at a different level from IP filtering and VPN policies. There is thus no need to integrate the AT-TLS policies into the rules used for VPN and filtering. In other words, AT-TLS and VPN rules can function together, effectively independent of each other.

11.2 REXX socket API support for AT-TLS

This section shows how to implement general AT-TLS support using a REXX socket server application running on SC30/A23 using stack TCPIPA, connecting to a REXX socket client application running on SC31/A24 using stack TCPIPB. It is beyond the scope for this book to

show the source of the client and server REXX programs and their JCL. We are simply using them to show the general functionality of AT-TLS.

The AT-TLS policies are provided to the stack by the Policy Agent (PAGENT). The Policy Agent thus has to be set up prior to activating AT-TLS. Setting up the Policy Agent and its associated security aspects is discussed in detail in Chapter 5, “Policy Agent” on page 221.

Figure 11-2 shows the relationship of the Policy Agent configuration files to one another.

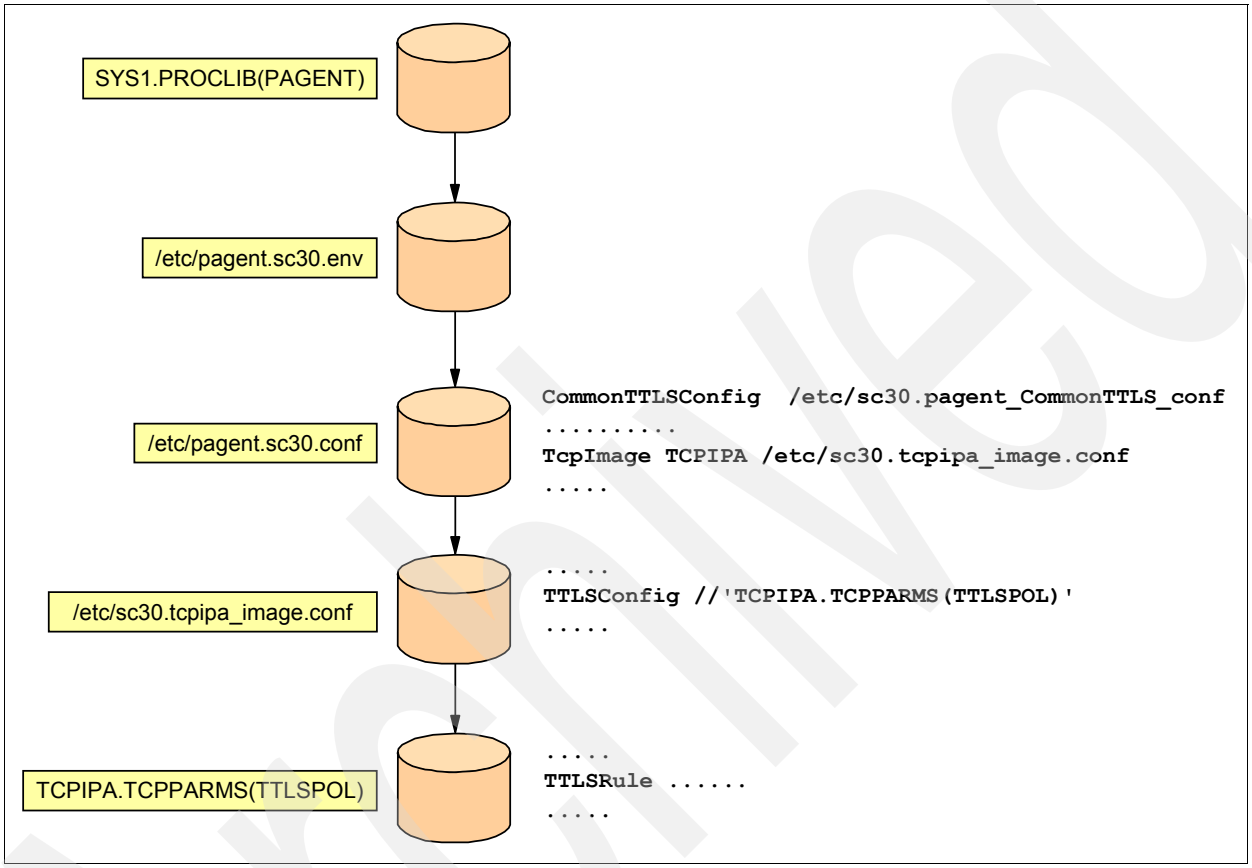


Figure 11-2 AT-TLS PAGENT configuration file relationship

Figure 11-2 is explained as follows:

We set up our AT-TLS related Policy Agent configurations files by coding a CommonTTLSConfig statement, pointing to file /etc/sc30.pagent_CommonTTLS_conf, containing AT-TLS objects shared across our TCP/IP stacks. This file contains the following policy statements:

- TTLSRule statements
- TTLSGroupAction statements
- TTLSEnvironmentAction statements
- TTLSConnectionAction statements

We also coded TcpImage statements pointing to a file /etc/sc30.tcpipa_image.conf for each TCP/IP stack. Each one of these stack TTLSConfig files contains a TTLSConfig statement that points to TCPIP.SC30.POLICIES(TTLSPOL). This file contains the following policy statements:

- TTLSRule statements
- TTLSGroupAction statements

- TTLSEnvironmentAction statements
- TTLSConnectionAction statements

The following RACF aspects are important for the successful implementation of AT-TLS:

- Setting up TTLS stack initialization access control
- Enabling CSFSERV resources
- Creating digital certificates and key rings

11.2.1 Description of REXX AT-TLS support

We set up a REXX socket client and server application on our two z/OS machines, SC30 and SC31, as shown in Figure 11-3, in order to demonstrate how this application can make use of the TLS protocol without requiring changes.

The server application runs under the job name of APISERV, and repeatedly accepts connections, writes out socket end-point information, and returns this data to the client application.The server binds to port 7000. The client application, APICLN, runs as a batch job on SC31, sends data to the server on SC30, and receives returned data.

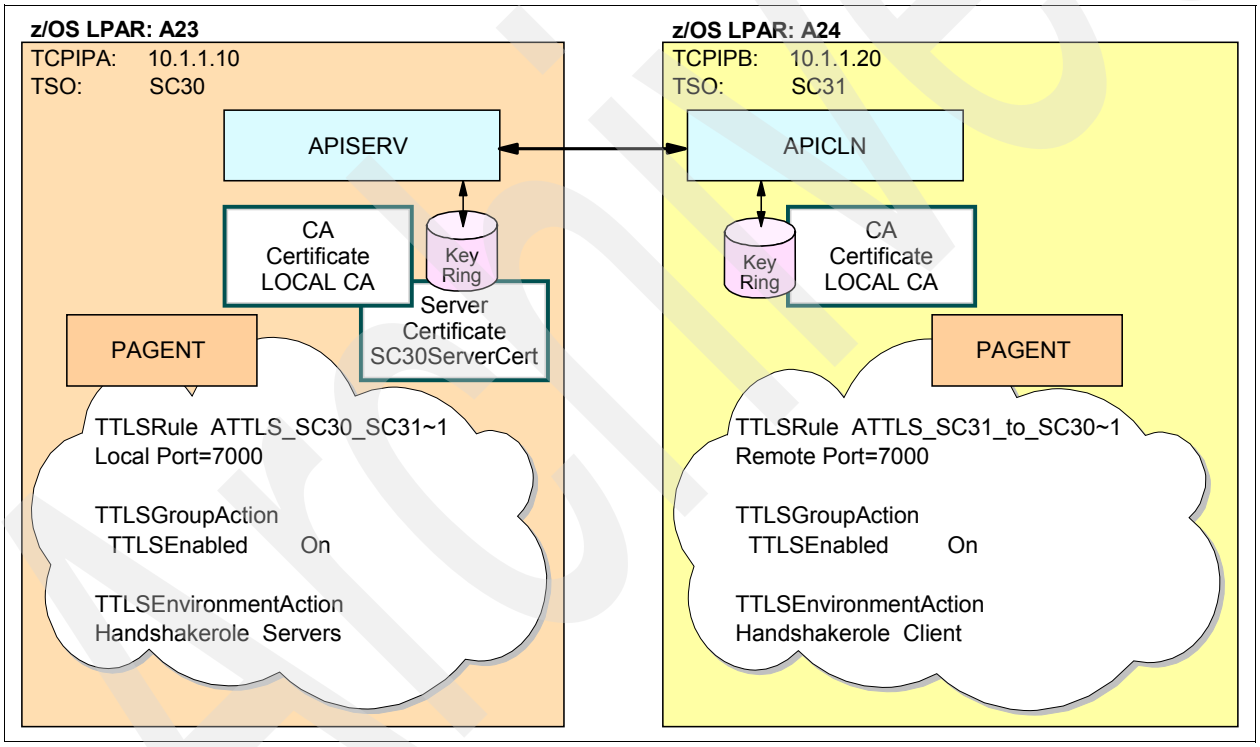


Figure 11-3 AT-TLS REXX socket APPLs

11.2.2 Configuration of REXX AT-TLS support

Setting up AT-TLS for our REXX client/server socket applications involved the following activities:

- Configuring the server policies
- Configuring the client policies
- Defining the digital certificates and key rings
- Enabling CSFSERV resources
- Controlling access during the window period
- Enabling AT-TLS in the TCP/IP profile

Configuring the server policies

We used the IBM Configuration Assistant for z/OS Communication Server to create our AT-TLS policy file for our server on SC30.

We performed the following steps to define the server policies:

1. Open the Configuration Assistant GUI and initiate an AT-TLS only session
2. Add a traffic descriptor object for REXX server.
3. Add a requirement map object for REXX server.
4. Add a z/OS image for the REXX server.
5. Add a TCP/IP stack to the z/OS image.
6. Add a connectivity rule.

Open the Configuration Assistant GUI and initiate an AT-TLS only session

1. Open the IBM Configuration Assistant for z/OS Communication Server. Select **File** → **Open** → **Create a New Backing Store**. A Backing Store is the file where your configuration file and backups will be saved. The Backing Store creates a new file each time the following occurs:
   - A file is saved.
   - A file is saved under a different name.
   - A file is imported.
   - A file from an older version of this tool is migrated to this version.
   - A stack configuration is installed on a host; that is, a configuration file is sent to the host via FTP. It does not need to be a local file.
2. In the Information panel, click **OK**.
3. In the panel Indicate a Directory and File Name for the New Backing Store, enter a file name for the configuration file and click **Open**. This will set your default file for a **File** → **Save** operation. Make sure that your Backup Store is backed up on a regular basis.

4. In the Main Perspective panel, select **AT-TLS Application Transparent - Transport Layer Security**, as shown in Figure 11-4. Click **Configure**.

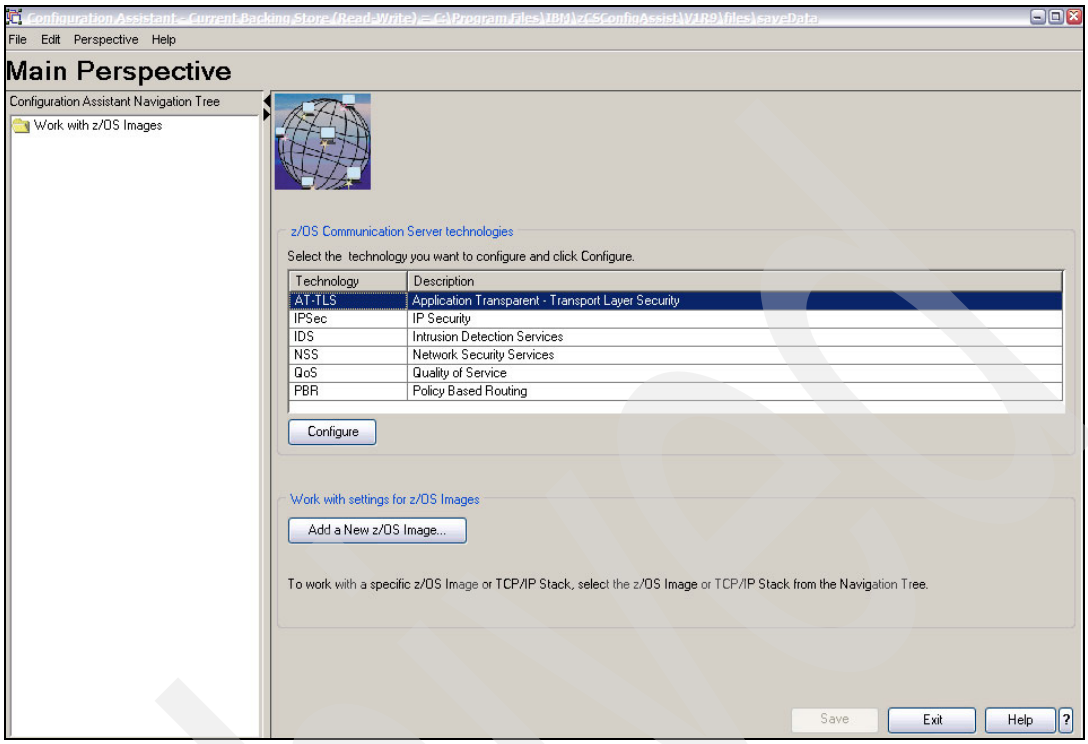


Figure 11-4 Welcome panel - selecting AT-TLS only policy

Before defining the z/OS images, TCP/IP stacks and the connectivity rules, define the traffic descriptors and the requirement maps that you will use for the connectivity rules.

Add a traffic descriptor object for REXX server

To add a traffic descriptor object for REXX server:

1. Select **AT-TLS Application Transparent - Transport Layer Security** and then click **Configure**.
2. You should now be viewing the AT-TLS Perspective panel. Under Work with reusable objects, click **Traffic Descriptors** and then **Add**.
3. In the New Traffic Descriptor panel, enter a name and description for the new traffic descriptor and click **Add**.

4. In the New Traffic Type panel, enter the details for the traffic descriptor. The server traffic descriptor shown in Figure 11-5 contains the AT-TLS handshake role button, which we set to `Server`. We are also using the key ring as defined on our z/OS image. We define our connection direction as incoming from all ephemeral ports to local port 7000.

   Click **OK**.

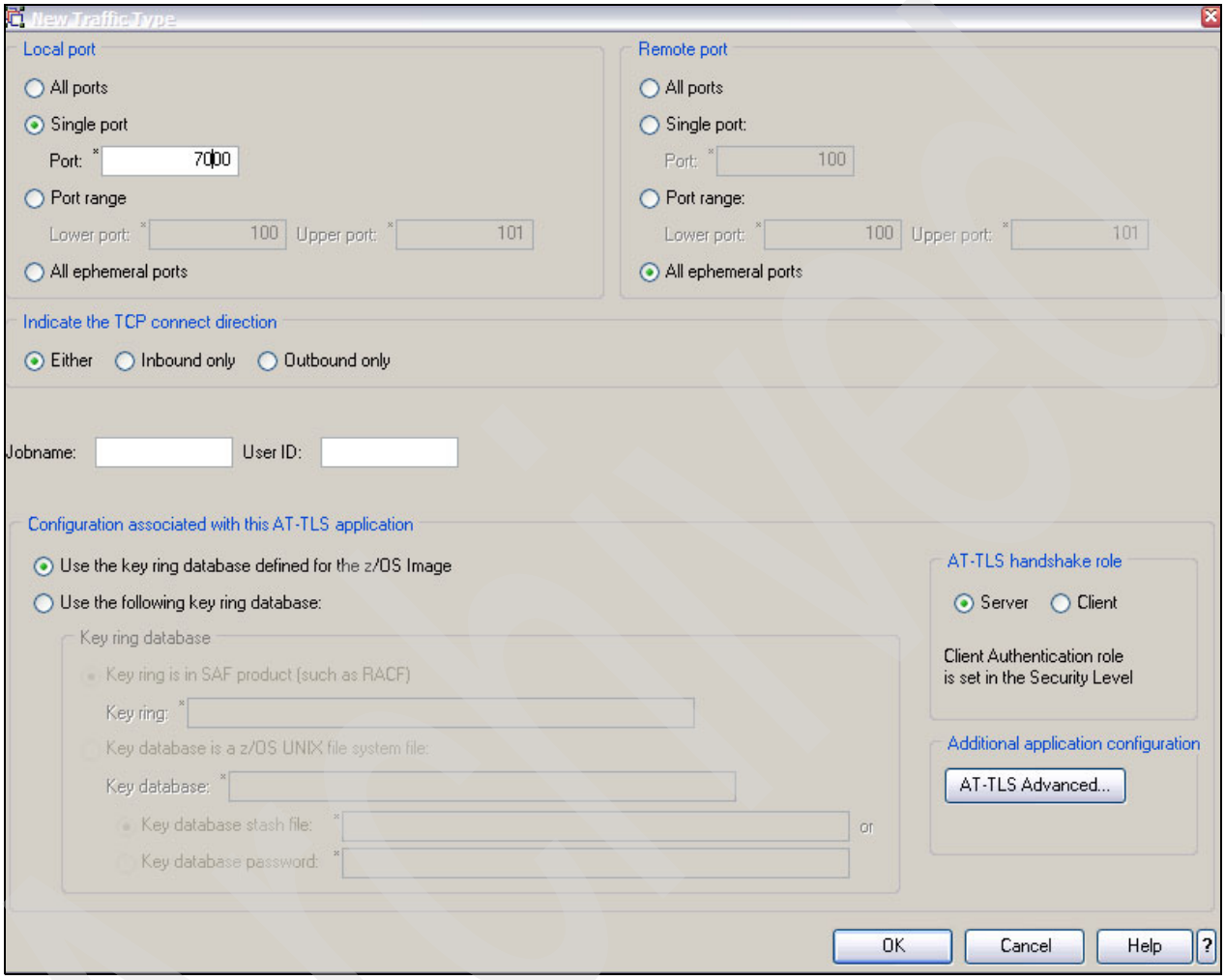


Figure 11-5 Server traffic type descriptor

For details regarding certificates, key rings, and handshake roles, refer to Chapter 3, “Certificate management in z/OS” on page 31.

5. The New Traffic Descriptor panel should now look like the panel shown in Figure 11-6. Click **OK**.

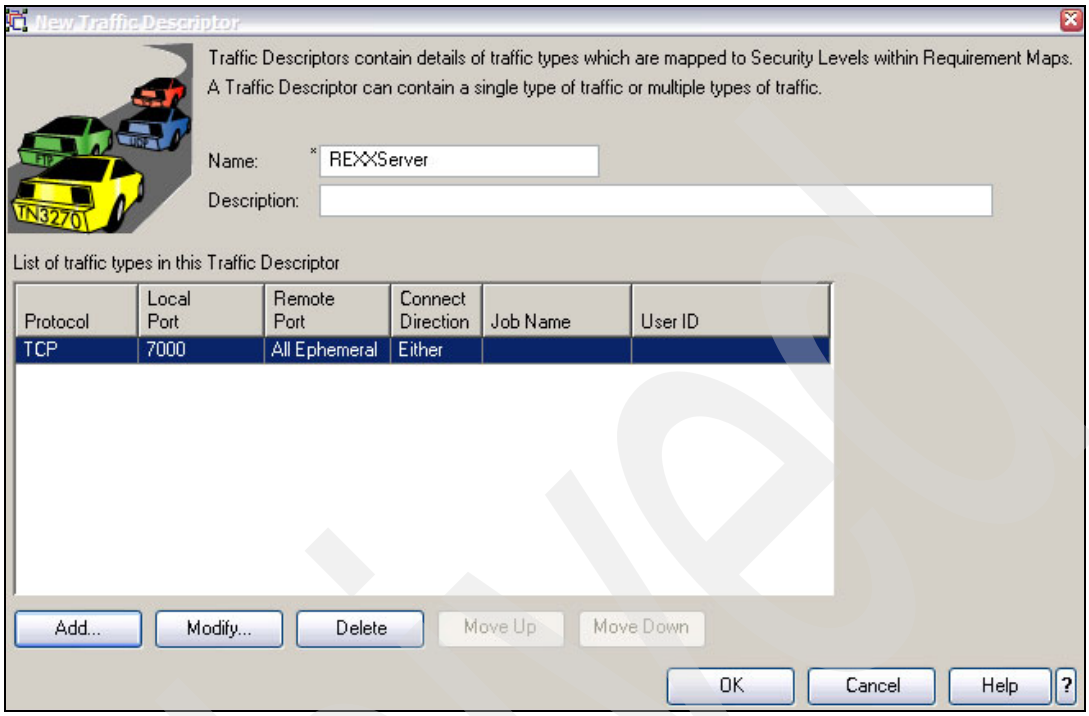


Figure 11-6 Traffic Descriptor Object panel: Adding a traffic descriptor for REXX server

Add a requirement map object for REXX server

To add a requirement map object for REXX server:

1. Select **AT-TLS** -> **Work with Reusable Objects** -> **Requirement Maps** from the Configuration Assistant Navigation Tree (the Navigation Tree is the left-most window pane in the panel).
2. You will see a window pane called List of all defined Requirement Map Objects. Click **Add**.
3. In the New Requirement Map panel, enter a name and a description for the requirement map. In addition, select the traffic descriptor you want to configure in this requirement map object (REXXServer) and click **Add**. Use the pull-down menu to select the right `AT-TLS - Security Level` for the traffic descriptor that you added by clicking the drop-down list.

   For example, we defined our server traffic descriptor with the supplied AT-TLS GOLD security level, which uses the following ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F and TLS_RSA_WITH_AES_128_CBC_SHA, as shown in Figure 11-7 on page 455. Click **OK**.

   At this point, you may be prompted with a “tip” on requirement map creation. Choose the **Proceed** button.

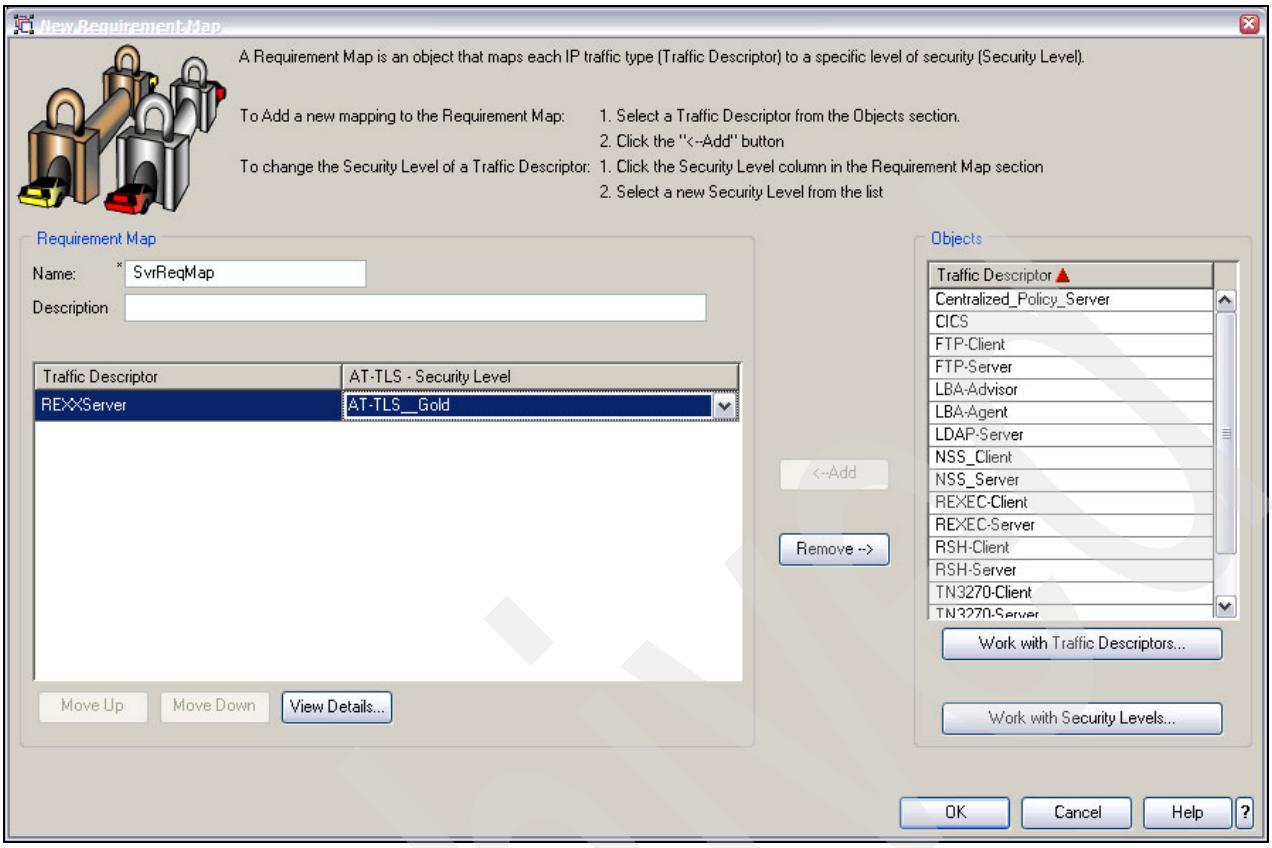


Figure 11-7 Server requirement map: adding a requirement map for REXX server

Add a z/OS image for the REXX server

To add a z/OS image for the REXX server:

1. If you have not already done so, select **Main Perspective** to return to the Main Perspective panel. Click **Add a New z/OS Image...**.
2. In New z/OS Image: Welcome panel, click **Next**.

3. In the New z/OS Image: Information panel, enter a name for z/OS image and description and click **Next**, as shown in Figure 11-8.

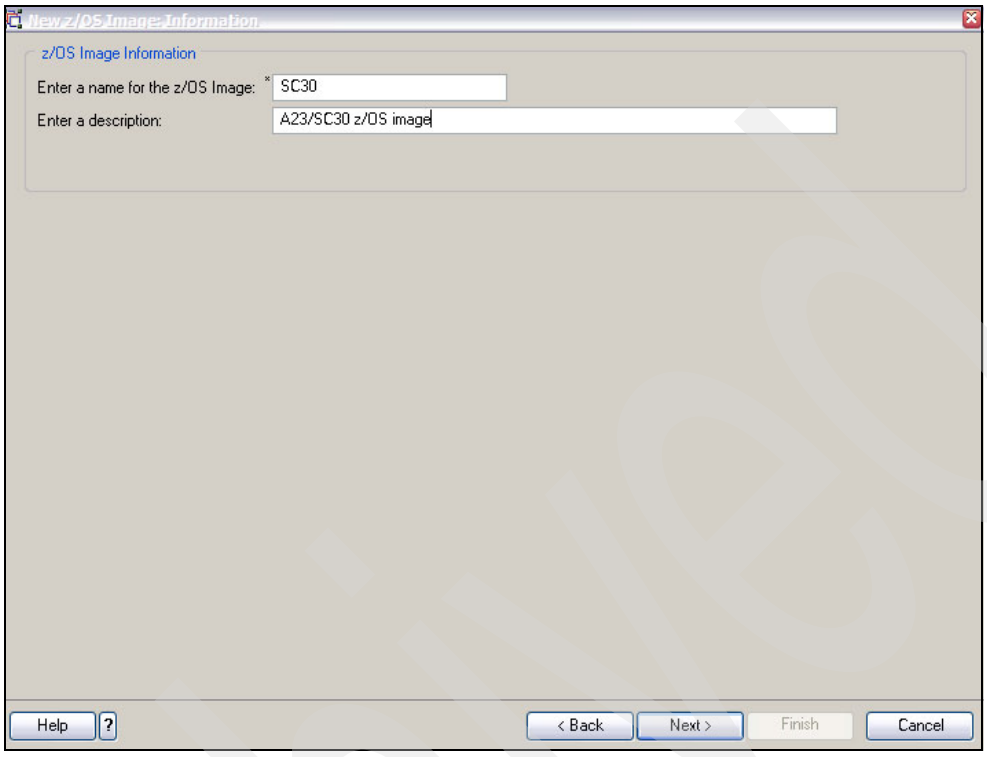


Figure 11-8 New z/OS Image: Information panel - adding SC30

4. In the New z/OS Image: AT-TLS Image Level Settings panel, enter the settings for the key ring database. We selected **Key ring is in SAF product** and entered the name of the key ring that we created in RACF, ATTLS_keyring. We selected all the trace levels available: **Level 1 - Errors (to TCP/IP Joblog)**, **Level 2 - Errors (to Syslog)**, and **Level 4 - Information (to Syslog)**, as shown in Figure 11-9. Click **Next**.

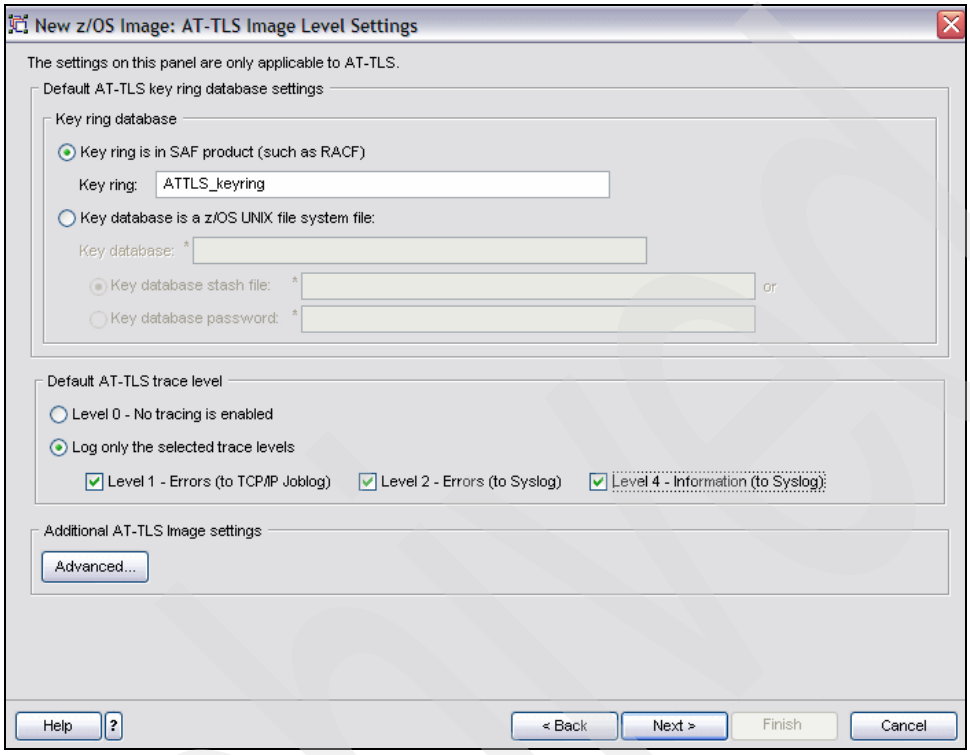


Figure 11-9 New z/OS Image: AT-TLS Image Level Settings panel

5. In the New z/OS Image: Finish panel, click **Finish**.
6. In the Proceed to the next step? panel, click **Yes**.

Add a TCP/IP stack to the z/OS image

To add a TCP/IP stack to the z/OS image:

1. In the New TCP/IP Stack Wizard: Welcome panel, click **Next**.
2. In the New TCP/IP Stack: Name panel, enter the TCP/IP stack name and description (in our case, TCPIPA), as shown in Figure 11-10. Click **Next**.

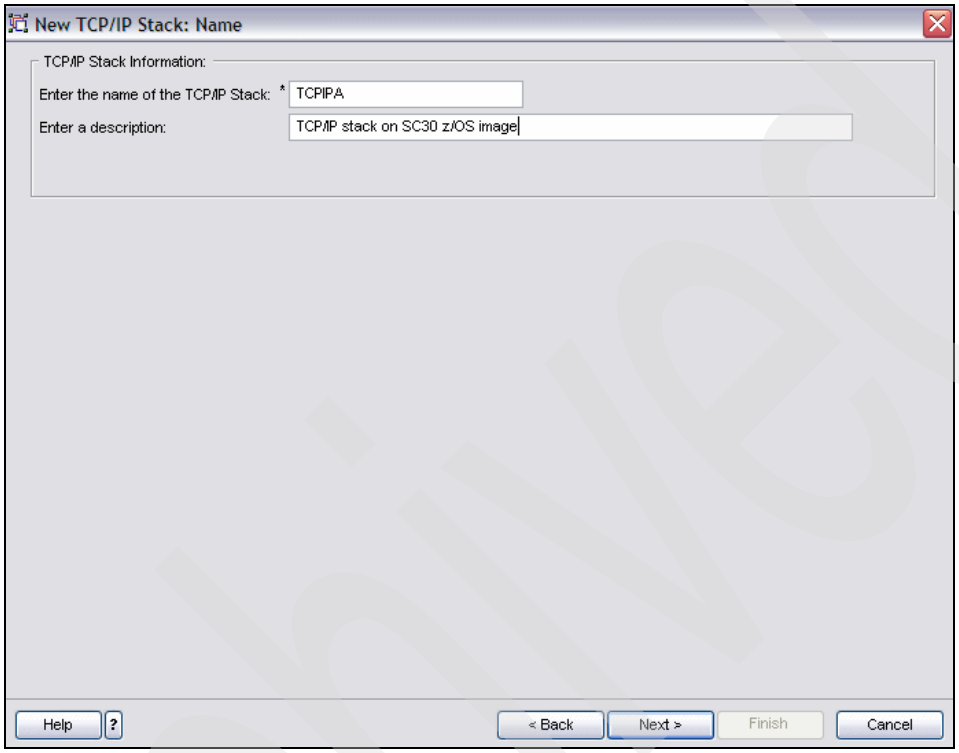


Figure 11-10 New TCP/IP Stack: Name panel - adding TCPIPA

3. In the New TCP/IP Stack: Finish panel, click **Finish**.
4. In the Proceed to the next step? panel, click **Yes**.

Add a connectivity rule

To add a connectivity rule:

1. In the New Connectivity Rule: Welcome panel, click **Next**.
2. In the New Connectivity Rule: Data Endpoints panel, enter the endpoints data and a rule name, as shown in Figure 11-11. Click **Next**.

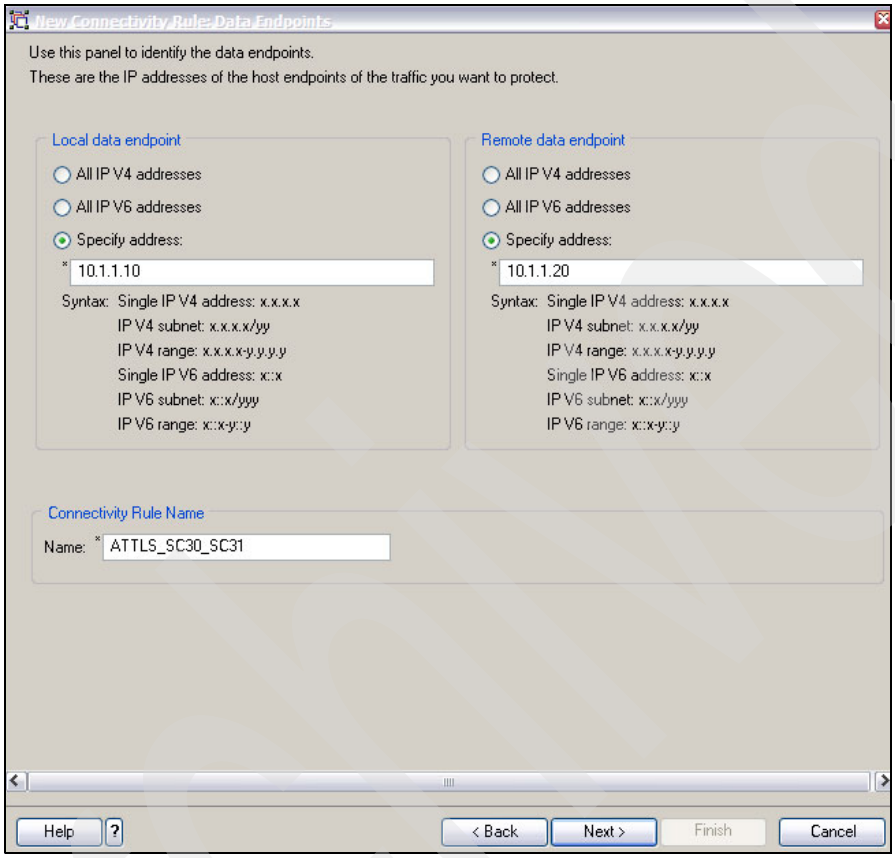


Figure 11-11 Connectivity Rule: Data Endpoints panel - creating a rule between SC30 and SC31

3. In the New Connectivity Rule: Select Requirement Map panel, select the requirement map that you created, as shown in Figure 11-12. Click **Next**.

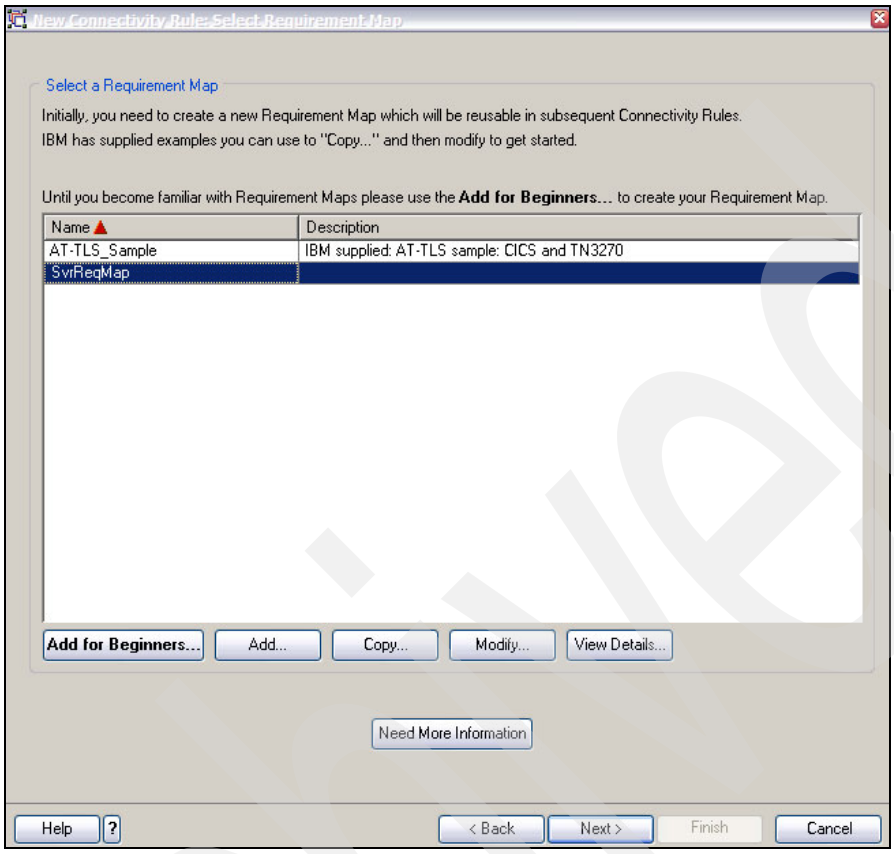


Figure 11-12 Connectivity Rule: Select Requirement Map panel

4. In the New Connectivity Rule: Finish panel, click **Finish**.

The resultant policy file for SC30 is displayed in Example 11-1. An explanation of these policies, along with the changes, follows.

Example 11-1 Server AT-TLS policy for TCPIPA on SC30

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC30
##    Stack: TCPIPA
. . .
##
TTLSRule                          ATTLS_SC30_SC31~1 1
{
  LocalAddrRef                    addr1
  RemoteAddrRef                   addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Direction                       Both
  Priority                        255
  TTLSGroupActionRef              gAct1~REXXServer
  TTLSEnvironmentActionRef        eAct1~REXXServer
  TTLSConnectionActionRef         cAct1~REXXServer
}
TTLSGroupAction                   gAct1~REXXServer
```

```
{
  TTLSEnabled                     On 2
  Trace                           7
}
TTLSEnvironmentAction             eAct1~REXXServer
{
  HandshakeRole                   Server 3
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~A23
}
TTLSConnectionAction              cAct1~REXXServer
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~AT-TLS__Gold
  Trace                           7
}
TTLSKeyringParms                  keyR~A23
{
  Keyring                         ATTLS_keyring 4
}
TTLSCipherParms                   cipher1~AT-TLS__Gold 5
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            10.1.1.10 6
}
IpAddr                            addr2
{
  Addr                            10.1.1.20 7
}
PortRange                         portR1
{
  Port                            7000 8
}
PortRange                         portR2
{
  Port                            1024-655359
}
```

1 The TTLSRule statement is used to define an AT-TLS rule. This policy is defined for inbound connections only and has been given the highest possible priority of 255 for the duration of our testing.

2 TTLSEnabled is the statement that turns on the AT-TLS function.

3 This statement controls who initiates the handshake. In the server role, AT-TLS will wait for an inbound hello from the client SSL handshake, which is performed like a server's handshake.

4 The key ring used was permitted to the user ID under which the REXX server started task was running. Even though the TCP/IP stack itself does the SSL calls, the security environment under which the calls execute is that of the application.

5 The AT-TLS__Gold cipher was selected, including the cipher specifications for this AT-TLS session.

6 This is the SC30 VIPA address.

7 This is the client source IP address; in our case, it is SC31 VIPA address.

8 The destination port used for testing was 7000.

9 The source port used for testing was all ephemeral ports.

Configuring the client policies

We used the IBM Configuration Assistant for z/OS Communication Server to create our AT-TLS policy file for our client on SC31.

We performed the following steps:

1. Add a traffic descriptor object for the REXX client.
2. Add a requirement map object for the REXX client.
3. Add a z/OS image for the REXX client (in our case, SC31).
4. Add a TCP/IP stack to the z/OS image (in our case, TCPIPA).
5. Add a connectivity rule.

As you can see, the steps are very similar to the steps for configuring the server policies.

Again, because the REXX client will run on z/OS, we used the IBM Configuration Assistant for z/OS Communication Server. The AT-TLS key ring was defined in the AT-TLS New z/OS Image: AT-TLS Image Level Settings on our SC31 z/OS image exactly as defined for SC30 in Figure 11-9 on page 457. We used a shared RACF database, which results in the ATTLS_keyring also being shared between the two systems.

The New Traffic Type panel shown in Figure 11-13 on page 463 contains the AT-TLS handshake role button, which we set to Client. We also used the key ring as defined on our z/OS image. We define our connection direction as outbound from all ephemeral ports to remote port 7000.

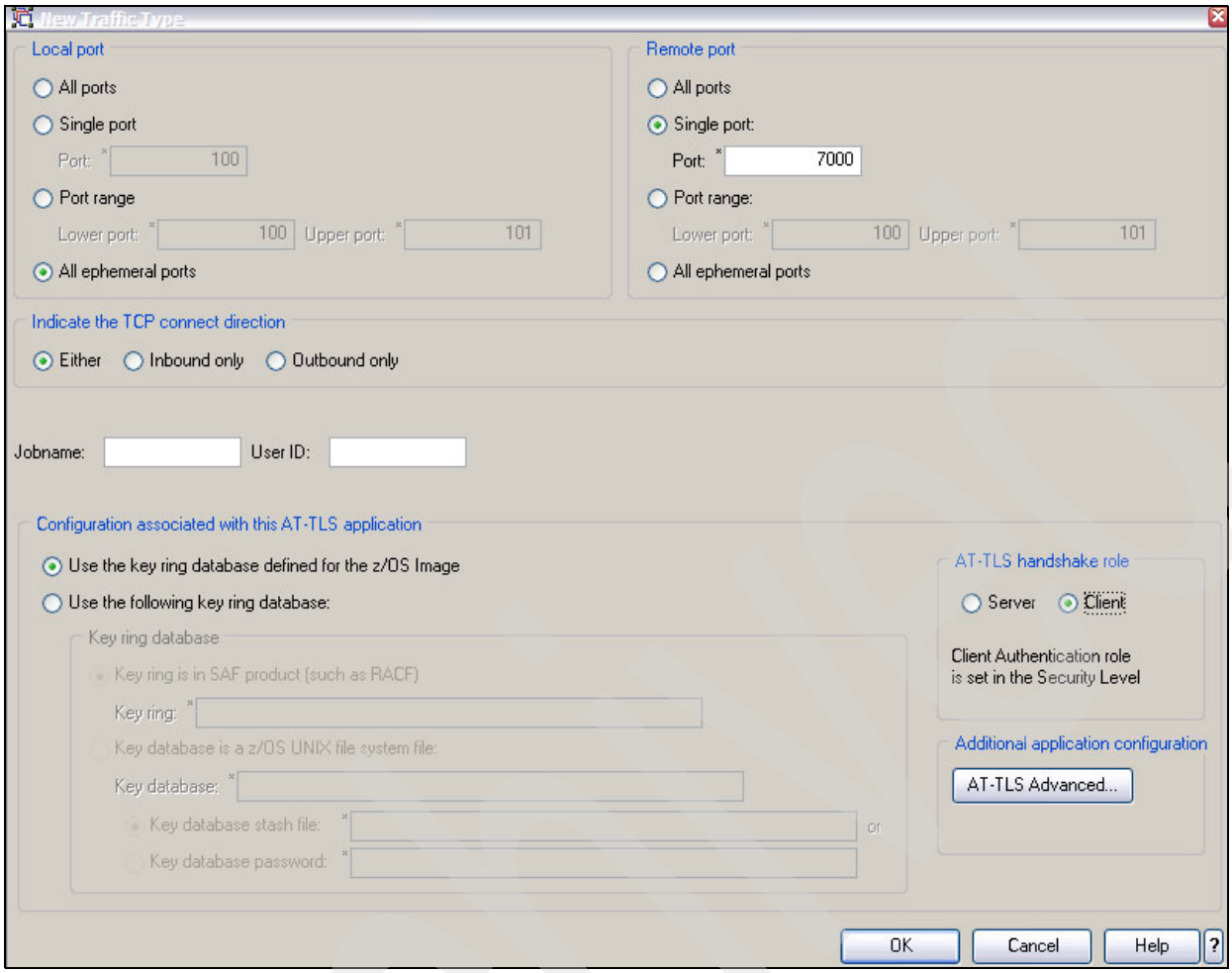


Figure 11-13 Client traffic descriptor

We defined our Client traffic descriptor with the supplied AT-TLS GOLD security level, which uses the following ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F and TLS_RSA_WITH_AES_128_CBC_SHA, as shown in Figure 11-14 on page 464.

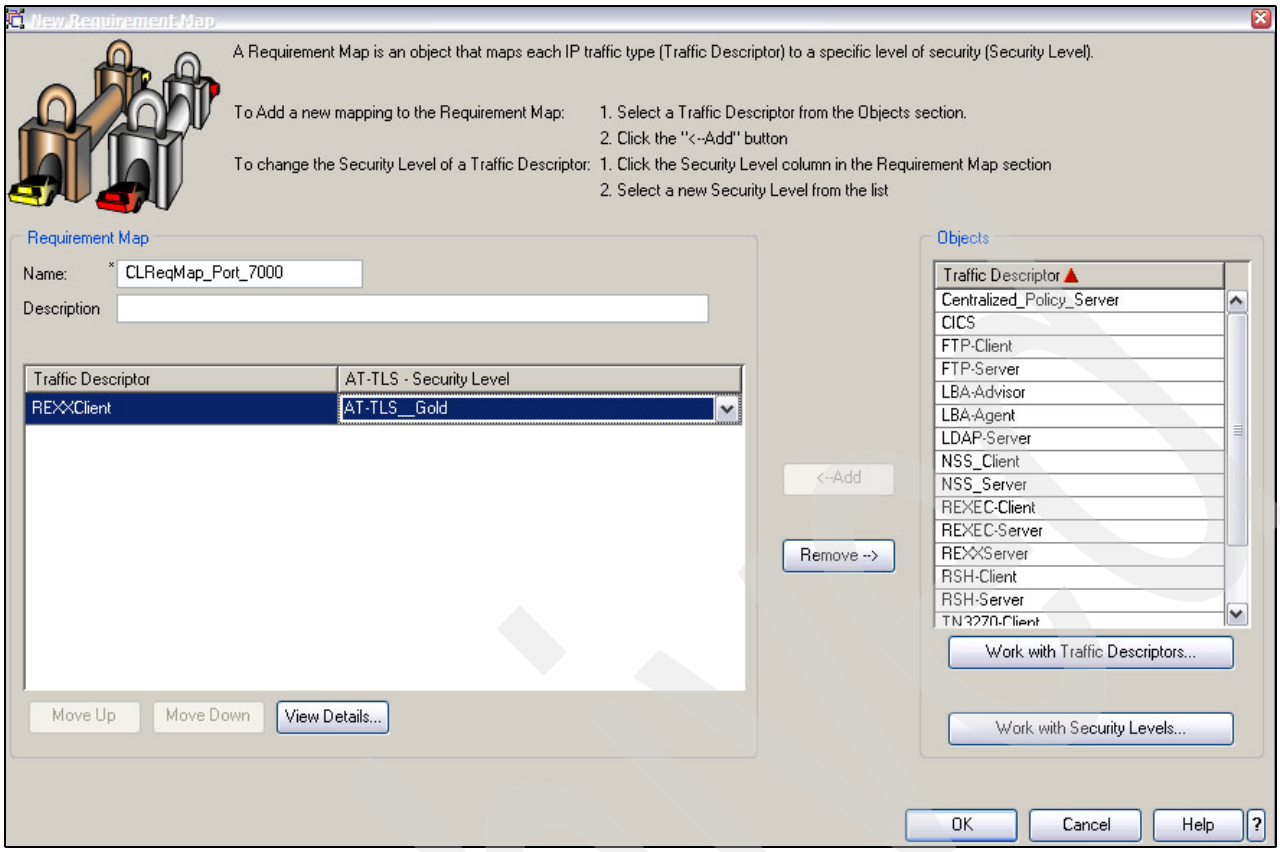


Figure 11-14 Client requirement map

The resultant policy file for SC31 is displayed in Example 11-2. The outbound connection direction for this client is reflected in this policy.

Example 11-2 Client AT-TLS policy for TCPIPA on SC31

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC31
##    Stack: TCPIPA
. . .
##
TTLSRule                          ATTLS_SC31_SC30~1
{
  LocalAddrRef                    addr1
  RemoteAddrRef                   addr2
  LocalPortRangeRef               portR1 1
  RemotePortRangeRef              portR2 2
  Direction                       Outbound
  Priority                        255
  TTLSGroupActionRef              gAct1~REXXClient
  TTLSEnvironmentActionRef        eAct1~REXXClient
  TTLSConnectionActionRef         cAct1~REXXClient
}
TTLSGroupAction                   gAct1~REXXClient
{
  TTLSEnabled                     On
  Trace                           7
}
TTLSEnvironmentAction             eAct1~REXXClient
```

```
{
  HandshakeRole                   Client
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~A24
}
TTLSConnectionAction              cAct1~REXXClient
{
  HandshakeRole                   Client
  TTLSCipherParmsRef              cipher1~AT-TLS__Gold
  Trace                           7
}
TTLSKeyringParms                  keyR~A24
{
  Keyring                         tlsKeyring
}
TTLSCipherParms                   cipher1~AT-TLS__Gold
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            10.1.1.20
}
IpAddr                            addr2
{
  Addr                            10.1.1.10
}
PortRange                         portR1 1
{
  Port                            1024-65535
}
PortRange                         portR2 2
{
  Port                            7000
}
```

Note that the local 1 and remote 2 port ranges have been reversed: from the client end, the target or remote port is 7000. Locally, the client will use any ephemeral port.

Defining the digital certificates and key rings

We created a key ring called ATTLS_keyring and connected a root CA certificate and personal server certificate to it, as shown in Example 11-3. We used a shared RACF database and therefore a shared key ring, so we were not required to export our CA certificate to a client key ring. The root CA certificate was defined as TRUSTED.

Example 11-3 Defining our digital certificates and key ring

```
SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

RACDCERT ID(CSO9) addring(ATTLS_keyring)

RACDCERT ID(cs09) CERTAUTH GENCERT                    -
         SUBJECTSDN( O('I.B.M Corporation')                    -
            CN('itso.ibm.com')                        -
            C('US')) TRUST                            -
            WITHLABEL('LOCALCA')                      -
            KEYUSAGE(certsign)
```

```
RACDCERT ID(CS09) GENCERT                                  -
         SUBJECTSDN  (CN('SC30ServerCert')                 -
                      OU('ITSO')                           -
                      C('US'))                             -
                      WITHLABEL('SC30ServerCert')          -
                      SIGNWITH(CERTAUTH                    -
                      label('LOCALCA'))

RACDCERT ID(CS09) CONNECT(ID(CS09)                         -
                     LABEL('SC30ServerCert')               -
                     RING(ATTLS_keyring)                   -
                     USAGE(personal))
RACDCERT ID(CS09) CONNECT(ID(CS09) CERTAUTH                -
                     LABEL('LOCALCA')                      -
                     RING(ATTLS_keyring)                   -
                     USAGE(certauth))

SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
```

Enabling CSFSERV resources

If you are using cryptographic hardware in conjunction with TLS security, and you have defined resources in the CSFSERV classes to protect cryptographic services, you should permit the user ID associated with the server to these resources.

With AT-TLS, the system SSL verifies that the user ID associated with the server is permitted to use CSFSERV resources. We defined the CSFDSV and CSFPKE services and permitted the RACF user ID CS09 to use the CSFSERV resource class, as shown in Example 11-4.

Example 11-4 Enabling CSFSERV resources

```
//RACFDEF EXEC PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
RDEFINE CSFSERV CSFDSV UACC(NONE)
RDEFINE CSFSERV CSFPKE UACC(NONE)
SETROPTS RACLIST(CSFSERV) REFRESH
PERMIT CSFDSV CLASS(CSFSERV) ID(CS09) ACCESS(READ)
PERMIT CSFPKE CLASS(CSFSERV) ID(CS09) ACCESS(READ)
SETROPTS RACLIST(CSFSERV) REFRESH
/*
```

Controlling access during the window period

When AT-TLS is enabled, the INITSTACK profile must be defined. The Policy Agent and any socket-based programs it requires must be permitted to this resource. Other programs or users that do not need to wait for the TTLS policy to be installed in the stack may be permitted to this resource. Users who are not permitted to this resource will not be able to open sockets on this stack until the TTLS policy is installed. You may receive a message like one of these when issuing a NETSTAT or FTP command:

```
EZZ2382I Unable to open UDP socket to TCPIPE  : TCPIPE   is not active.
```

or

```
EZA2590E socket error from initIPv4Connection - EDC5112I Resource temporarily
         unavailable. (errno2=0x12CA00B6)
```

However, after the policy agent has established its policies, you should no longer receive such messages.

When the resource is not defined, no stack access is permitted. We defined this profile for SC30 and SC31, as shown in Example 11-5.

Example 11-5 Setup TTLS stack initialization access control for SC30 and SC31

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.SC30.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC30.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
RDEFINE  SERVAUTH EZB.INITSTACK.SC31.TCPIPA UACC(NONE)
PERMIT   EZB.INITSTACK.SC31.TCPIPA CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

Enabling AT-TLS in the TCP/IP profile

To activate AT-TLS, the TTLS parameter has to be added to the TCPCONFIG profile statement in each stack that is to support AT-TLS, as shown in Example 11-6.

Example 11-6 Profile statement to enable AT-TLS

```
TCPCONFIG TTLS
```

11.2.3 Activation and verification of REXX AT-TLS support

We installed our policies on the client and server side. We started PAGENT as shown in Example 11-7, started our APISERV application on SC30, and started our APICLN application on SC31.

Example 11-7 PAGENT startup on SC30

```
S PAGENT

000090  $HASP373 PAGENT   STARTED
000090  EZZ8431I PAGENT STARTING
000090  EZZ8432I PAGENT INITIALIZATION COMPLETE
000090  EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : TTLS
```

We issued a NETSTAT TTLS display on the client side, as shown in Example 11-8, to determine whether the stack mapped a connection to our client AT-TLS policy and, if so, to which policy it was mapped.

Example 11-8 Display result of NETSTAT TTLS on client

```
. . .
TTLSGrpAction                          Group ID           Conns
 ------------------------------------  -----------------  -----
 gAct1~REXXClient                       00000002               0
```

We issued a NETSTAT TTLS display on the server side, as shown in Example 11-9 on page 468, to determine whether the stack mapped a connection to our server AT-TLS policy and, if so, to which policy it was mapped.

Example 11-9 Display result of NETSTAT TTLS on server

```
. . .
TTLSGrpAction                          Group ID          Conns
-------------------------------------- ----------------  -----
gAct1~REXXServer                       00000002              0
```

Our NETSTAT ALLCONN command on SC30 showed that the APISERV application was listening on port 7000 (see Example 11-10).

Example 11-10 NETSTAT ALLCONN on server

```
. . .
User Id  Conn     State
 -------  ----     -----
 APISERV  00004765 Listen
   Local Socket:   0.0.0.0..7000
   Foreign Socket: 0.0.0.0..0
:
```

Our NETSTAT ALL command in Example 11-11 on SC31 showed that our client application APICLN connected to IP address 10.1.1.10 and port 7000 from the client IP address 10.1.1.20 ephemeral port 1041.

Example 11-11 Display results of NETSTAT ALL on client

```
. . .
Client Name: APICLN                 Client Id: 00004C0B
   Local Socket: 10.1.1.20..1041
   Foreign Socket: 10.1.1.10..7000
 BytesIn:            0000000000000000031
 BytesOut:           0000000000000000031
 SegmentsIn:         0000000000000000011
 SegmentsOut:        0000000000000000013
 Last Touched:       07:38:13          State:              TimeWait
 RcvNxt:             3475413650        SndNxt:             3489619133
 ClientRcvNxt:       3475412176        ClientSndNxt:       3489618831
 InitRcvSeqNum:      3475412144        InitSndSeqNum:      3489618799
 CongestionWindow:   0000065120        SlowStartThreshold: 0000065535
 IncomingWindowNum:  3475446389        OutgoingWindowNum:  3489651872
 SndWl1:             3475413650        SndWl2:             3489619133
 SndWnd:             0000032739        MaxSndWnd:          0000032768
 SndUna:             3489619133        rtt_seq:            3489619103
 MaximumSegmentSize: 0000008140        DSField:            00
 Round-trip information:
   Smooth trip time: 0.000             SmoothTripVariance: 201.000
 ReXmt:              0000000000        ReXmtCount:         0000000000
 DupACKs:            0000000000
 SockOpt:            8000              TcpTimer:           0C
 TcpSig:             00                TcpSel:             C0
 TcpDet:             E0                TcpPol:             02
 QOSPolicyRuleName:
 TTLSPolicy:         Yes
   TTLSRule:         ATTLS_SC31_to_SC30~1
   TTLSGrpAction:    gAct1~REXXClient
   TTLSEnvAction:    eAct1~REXXClient
   TTLSConnAction:   cAct1~REXXClient
 ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
```

Our NETSTAT ALL command on SC30 showed that our server application APISERV was listening on port 7000; see Example 11-12.

Example 11-12 Display results of NETSTAT ALL

```
. . .
Client Name: APISERV                  Client Id: 00000037
   Local Socket: 0.0.0.0..7000
Foreign Socket: 0.0.0.0..0
  BytesIn:            0000000000000000000
  BytesOut:           0000000000000000000
  SegmentsIn:         0000000000000000000
  SegmentsOut:        0000000000000000000
  Last Touched:       07:40:43          State:              Listen
  RcvNxt:             0000000000        SndNxt:             0000000000
  ClientRcvNxt:       0000000000        ClientSndNxt:       0000000000
  InitRcvSeqNum:      0000000000        InitSndSeqNum:      0000000000
  CongestionWindow:   0000000000        SlowStartThreshold: 0000000000
  IncomingWindowNum:  0000000000        OutgoingWindowNum:  0000000000
  SndWl1:             0000000000        SndWl2:             0000000000
  SndWnd:             0000000000        MaxSndWnd:          0000000000
  SndUna:             0000000000        rtt_seq:            0000000000
  MaximumSegmentSize: 0000000536        DSField:            00
  Round-trip information:
    Smooth trip time: 0.000             SmoothTripVariance: 1500.000
  ReXmt:              0000000000        ReXmtCount:         0000000000
  DupACKs:            0000000000
  SockOpt:            8000              TcpTimer:           00
  TcpSig:             00                TcpSel:             00
  TcpDet:             C0                TcpPol:             00
  QOSPolicyRuleName:
ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
    ConnectionsIn:      0000000001        ConnectionsDropped: 0000000000
    CurrentBacklog:     0000000000        MaximumBacklog:     0000000010
    CurrentConnections: 0000000000        SEF:                100
    Quiesced: No
```

11.3 Problem determination for AT-TLS

The NETSTAT command can aid in problem determination and assist in checking the status of your connections. The following functions that pertain to AT-TLS are available:

- NETSTAT ALL
- NETSTAT ALLCONN
- NETSTAT TTLS
- pasearch -t

Other useful problem determination aids are:

- Reviewing SYSLOGD
- Running a CTRACE with option TCP or a packet trace
- Setting debug traces using the **`TTLSConnectionAction`** statement

The trace value is interpreted by AT-TLS as a bit map. Each of the options is assigned a value that is a power of 2, as shown in Table 11-1 on page 470. Add together the values of each option that you want to activate.

The default trace value is 2, which provides error messages to syslogd. When you are deploying a new policy, you might find it beneficial to specify a trace value of 6 or 7. This provides connection information messages, in addition to error messages in syslogd. The information messages provide positive feedback that connections are mapping to the intended policy.

Trace options event (8), flow (16), and data (32) are intended primarily for diagnosing problems. Trace values larger than 7 can cause a large number of trace records to be dropped instead of being sent to syslogd.

Table 11-1 Trace values and descriptions

Trace value	Description
0	No tracing is enabled.
1	Errors are traced to the TCP/IP joblog.
2	Errors are traced to syslogd.
4	Tracing of when a connection is mapped to an AT-TLS rule and when a secure connection is successfully initiated is enabled.
8	(Event) Tracing of major events is enabled.
16	(Flow) Tracing of system SSL calls is enabled.
32	(Data) Tracing of encrypted negotiation and headers is enabled.
64	Reserved.
128	Reserved.
255	All Tracing is enabled.

For information about AT-TLS codes that are above 5000, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, or to the appropriate IP messages manual.

For information about codes below 5000, refer to *z/OS Cryptographic Services System SSL Programming*, SC24-5901.

11.4 Additional information sources for AT-TLS

For additional information, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Cryptographic Services System SSL Programming*, SC24-5901

Tip: For descriptions of security considerations that affect specific servers or components, see "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

12

Intrusion Detection Services

“Intrusion” is a term describing undesirable activities. The objective of an intrusion may be to acquire information that a person is not authorized to have. It may be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. It may also be to cause business harm by rendering a network, host system, or application unusable. Most intrusions follow a pattern of information gathering, attempted access, and then destructive attacks. Intrusion Detection Services (IDS) thus guards against these intrusions, thereby providing protection against potential hackers.

The following topics are discussed in this chapter.

Section	Topic
12.1, “What is Intrusion Detection Services” on page 472	The different types of intrusions, and how policies are used to fend them off
12.2, “Basic concepts” on page 473	Scan detection, attack detection, and traffic regulation
12.3, “How IDS is implemented” on page 481	Using eServer™ IDS Configuration Manager to create IDS policies

12.1 What is Intrusion Detection Services

Intrusion Detection Services (IDS) is a z/OS Communications Server security protection mechanism that inspects inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. IDS can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. It can also provide a reactive system whereby IDS responds to the suspicious activity by taking policy-based actions.

Figure 12-1 illustrates the IDS architecture. As shown, a flat file for IDS policies has been implemented that will allow all policies to be stored in flat files, removing any requirements for an LDAP server environment. Policies may be stored on an LDAP server and are downloaded to the Policy Agent (PAGENT), or they can be stored directly in the Policy Agent from a flat file. The Policy Agent in turn installs the policies in the stack. When an attack is identified, any of the following resultant policy actions can be taken:

- Event logging
- Statistics gathering
- Packet tracing
- Discarding of the attack packets

Some IDS policies log events and statistics in syslogd and the system console via the Traffic Regulation Management Daemon (TRMD).

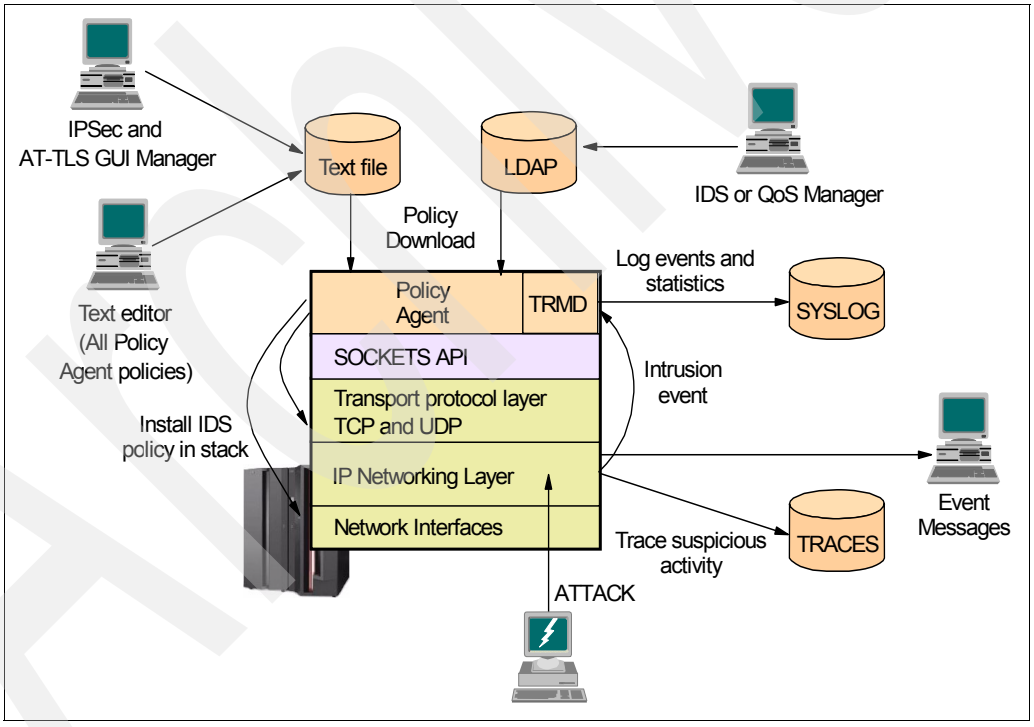


Figure 12-1 IDS architecture

12.2 Basic concepts

IDS functions can be subdivided into three areas:

- Scan detection
- Attack detection
- Traffic regulation

IDS is managed through policies. The policy is designed by the network administrator and based on preconceived events. The policy must include factors such as who, what, where, when, and how:

- *Who* is allowed to connect to the host?
- *What* applications or ports are clients allowed to use?
- *Where* is the attack, intruder, or traffic emanating from?
- *When* should I consider something to be an attack or scan?
- *How* is my system affected by the attack, scan, or traffic?

In z/OS V1R8 Communications Server, IDS policies are supported in a Policy Agent configuration file as well as an LDAP server. This solution provides an IDS policy solution that is consistent with other policy types for installations that do not have an LDAP infrastructure in place or that favor using configuration files instead of an LDAP configuration.

Note: The IBM Configuration Assistant for z/OS Communication Server is intended to replace the zIDS Manager for configuring IDS.

In this chapter, we cover only the Policy Agent configuration file that is created with the IBM Configuration Assistant for z/OS Communication Server.

The policy information is loaded into the Policy Agent application during PAGENT startup. All IDS policies allow the logging events to a specified message level in syslogd or the system console. Most IDS policies support discarding packets when a specified limit is reached. Most IDS policies support writing statistics records to the INFO message level of syslogd on a specified time interval, or if exception events have occurred.

All IDS policies support tracing all or part of the triggering packet to an IDS-specific CTRACE facility, SYSTCPIS. IDS assigns a correlator value to each event. Messages written to the system console and syslogd, and records written to the IDS CTRACE facility, all use this correlator. A single detected event may involve multiple packets. The correlator value helps to identify which message and packets are related to each other.

The type of policy written can be a *scan policy*, *attack policy*, or *traffic regulation policy*. The following sections give detailed descriptions of each of these policies.

12.2.1 Scan policies

Scans are detected because of multiple information gathering events from a single source IP within a defined time frame. Scanning is not harmful and may be part of normal operation, but many serious attacks, especially access violation attacks, are preceded by information gathering scans. Due to the fact that most scans use consistent source IP addresses, they can be monitored and the data processed to help prevent an attack or determine the origins of a previous attack.

The scanner is defined as a source host that accesses multiple unique resources (ports or interfaces) over a specified period of time. The number of unique resources (threshold) and

the time period (interval) can be specified via policy. Two categories of scans are supported: fast scans and slow scans.

Fast scan

During a fast scan, many resources are rapidly accessed in a short time period (they are usually program-driven and take less than five minutes), as shown in Figure 12-2.

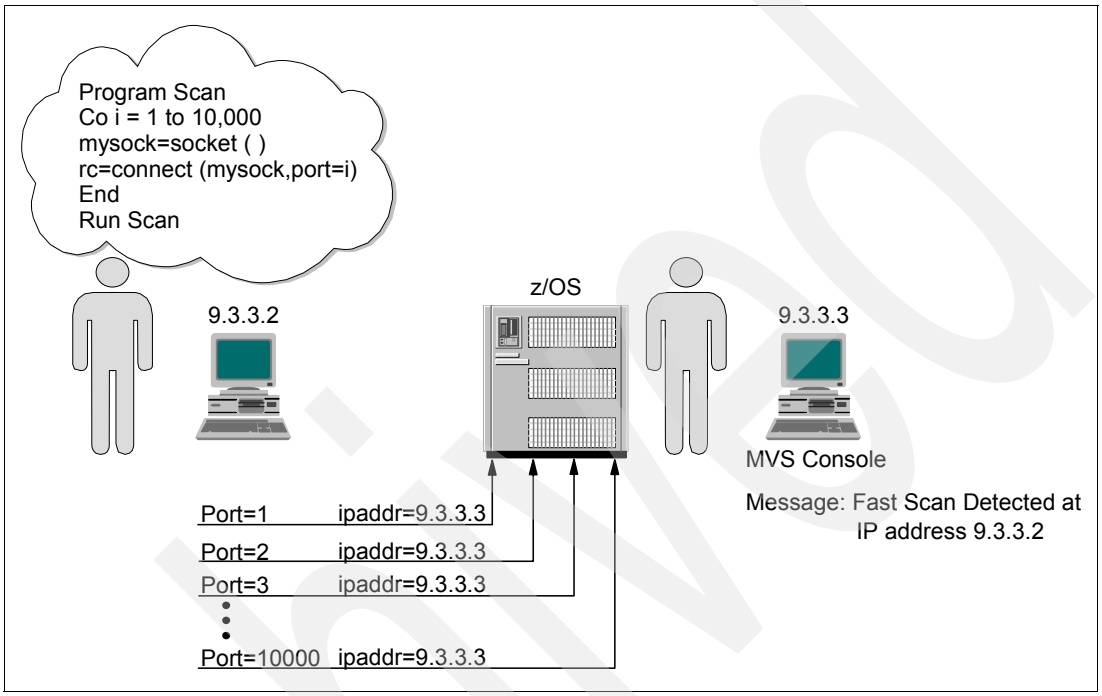


Figure 12-2 Fast scan

Slow scan

During a slow scan, different resources are intermittently accessed over a longer period of time (many hours). This could be a scanner trying to avoid detection, as shown in Figure 12-3.

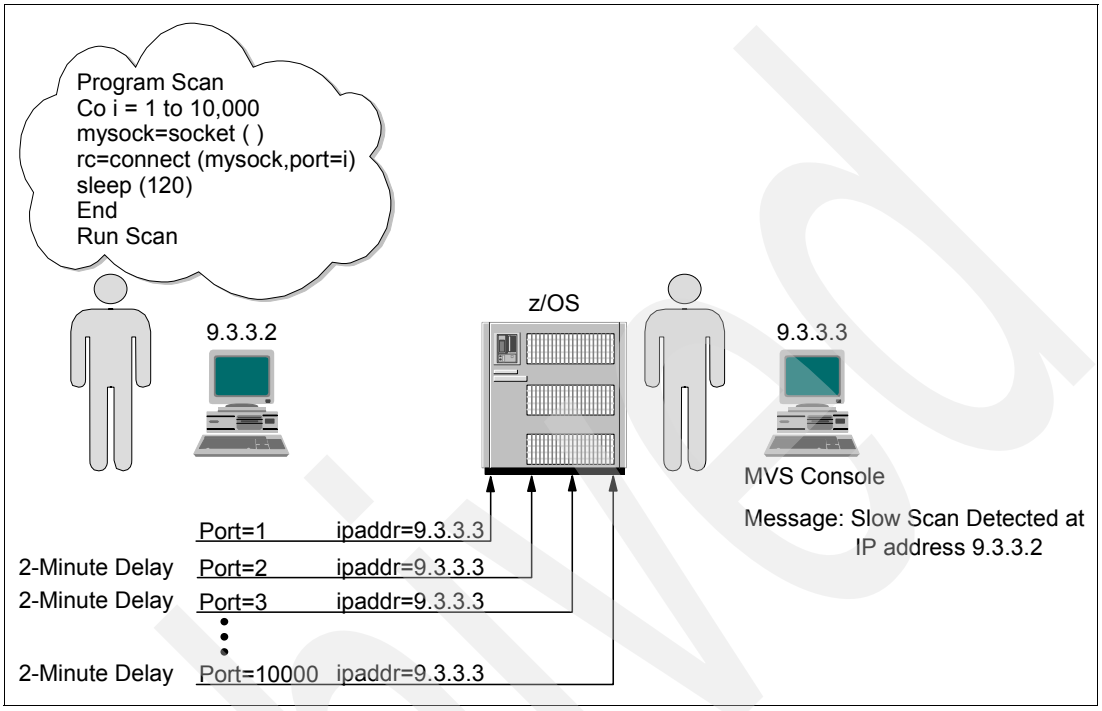


Figure 12-3 Slow scan

A fast scan scenario may be one in which an attack is based on the information provided through a program that loops through ports 1 - 1025 (normally the ports used by the server for listening ports), determining which ports have active listeners. This information may be the basis for a future attack.

A slow attack is more deliberate; occasional packets may be sent out to different ports over a long period of time with the same fundamental purpose: obtaining host information.

The same port being accessed will not generate multiple event records, for example, if a client from the same source IP address generates 20 connections to port 23 (TN3270 server). This is not considered a scan because only one unique resource has been accessed.

Note: Scan policies do not provide the ability to reject a connection. The actual rejection of the connection based on the source IP address must be configured in the Traffic Regulation policy or firewall.

Scan policy parameters

A scan policy provides the ability to control the following parameters that define a scan:

- Fast scan time interval
- Slow scan time interval
- Fast scan threshold
- Slow scan threshold
- Exclude well-known legitimate scanners via an exclusion list
- Specify a sensitivity level by port or port range (to reduce performance impacts)

- Notify the installation of a detected scan via console message or syslogd message
- Trace potential scan packets

The policy allows the user to set a sensitivity level. This is known as *policy-specified sensitivity*. This is used in parallel with the categorization of the individual packets to determine whether a packet should be counted as a scan event. The event classification is a normal, possibly suspicious, or very suspicious event. This logic is used to control the performance impact and analysis load of scan monitoring by only counting those individual packets where the chart indicates a count value. This value is then added with the current count total of scan events and compared with the threshold value to determine if we have met or exceeded the threshold in a specified time interval.

Scan events

Scan events are classified into Internet Control Message Protocol (ICMP), UDP port, and TCP port scans categories. The scan categories are described here:

- ICMP scan

  ICMP requests (echo, information, time stamp, and subnet mask) are used to obtain or map network information. The type of ICMP request determines the event classification.

- TCP port scans

  TCP is a stateful protocol. There are many different events that may be classified as normal, possibly suspicious, or highly suspicious.

- UDP port scans

  UDP is stateless. The stack is unable to differentiate between a client port and a server port. A scanner sending messages to many ephemeral ports looks very similar to a DNS server sending replies to many clients on ephemeral ports. TCP/IP configuration allows UDP ports to be RESERVED, therefore restricting a port so that it cannot be used.

Any countable scan event will count against an origin source IP address. The total number of countable events from all categories is compared to the policy thresholds. When an origin source IP address has exceeded the policy-defined fast or slow threshold, an event may be sent to the TRMD for logging to syslogd, a console message may be issued, and optionally a packet trace record issued. This is all dependent upon the notification actions set in the action of the policy. After a scan event is logged for a particular source IP address, no further scan events will be reportable within the specified fast interval. The intervals and thresholds for fast and slow scan are global. Only one definition of them is allowed across all event categories at a given point in time.

False positive scans

IDS attempts to reduce the recording of false scan events. This can be manually coded in the policy by excluding a source IP address, port, or subnet. This is useful if you have a particular client that probes the TCP/IP stack for general statistical information. Also, only unique events from a source IP address are counted as a scan event. An event is considered unique if the four-tuple, client IP address, client port, server IP address, and server port are unique, as well as the IP protocol for this scan interval. In the case of ICMP, a packet is unique if the type has not been seen before within this scan interval.

12.2.2 Attack policies

An *attack* is defined as an assault on system security. An attack is usually an intelligent act that includes a deliberate attempt to evade security services and violate the security policy of a system. In practice, however, an attack can even be inadvertent. For example, a malfunctioning host or application could generate a flood of SYN packets to a host. The security policies will not differentiate between a malicious intent and an accident. And, they should not differentiate, because attack policies exist to protect the z/OS host.

In fact, one of the difficulties of attack policies is how to differentiate between a valid user application repeatedly attempting to make a connection, and a hacking program trying to do damage.

An attack may be in the form of a single packet or multiple packets. There are two types of attacks, active and passive:

- An *active attack* is designed to alter system resources or affect their operation.
- A *passive attack* is designed to learn or make use of system information, but not affect system performance.

For more information about passive and active attacks, refer to RFC 4949.

The attack policies designed for IDS are based on active attacks. One may consider scanning to be more of a passive attack.

IDS attack policy allows the network administrator to provide network detection for one or more categories of attacks independently of each other. In general, the types of actions that can be specified for an attack policy are notifications (that is, event logging, statistics gathering, packet tracing), and discarding the attack packets.

IDS attack categories

The IDS categories of attacks are described in Table 12-1.

Table 12-1 Attack categories

Category	Attack description	Actions
Malformed packets	There are numerous attacks designed to crash a system's protocol stack by providing incorrect partial header information. The source IP address is rarely reliable for this type of attack.	► TCP/IP stack: Always discards malformed packets. ► IDS policy: May provide notification.
Inbound fragment restrictions	Many attacks are the result of fragment overlays in the IP or transport header. This support allows you to protect your system against future attacks by detecting fragmentation in the first 256 bytes of a packet.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
IP protocol restrictions	There are 256 valid IP protocols. Only a few are in common usage today. This support allows you to protect your system against future attacks by prohibiting those protocols that you are not actively supporting.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
IP option restriction	There are 256 valid IP options, with only a small number currently in use. This support allows you to prevent misuse of options that you are not intentionally using. Checking for restricted IP options is performed on all inbound packets, even those forwarded to another system.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.

Category	Attack description	Actions
UDP perpetual echo	Some UDP applications unconditionally respond to every datagram received. In some cases, such as Echo, CharGen, or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases, it may be polite application behavior to send error messages in response to incorrectly formed requests. If a datagram is inserted into the network with one of these applications as the destination and another of these applications spoofed as the source, the two applications will respond to each other continually. Each inserted datagram will result in another perpetual echo conversation between them. This support allows you to identify the application ports that exhibit this behavior.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause packet to be discarded.
ICMP redirect restrictions	ICMP redirect packets can be used to modify your routing tables.	► TCP/IP stack: Will discard ICMP redirects if IGNOREREDIRECT is coded in the tcpip.profile. ► IDS policy: May provide notification and disable redirects (this can optionally be coded as a parameter in the tcpip.profile).
Outbound raw restrictions	Most network attacks require the ability to craft packets that would not normally be built by a proper protocol stack implementation. This support allows you to detect and prevent many of these crafting attempts so that your system is not used as the source of attacks. As part of this checking, you can restrict the IP protocols allowed in an outbound RAW packet. It is recommended that you restrict the TCP protocol on the outbound raw rule.	► TCP/IP stack: No default action. ► IDS policy: May provide notification and cause the packet to be discarded.
TCP SYNflood	One common denial of service attack is to flood a server with connection requests from invalid or nonexistent source IP addresses. The intent is to use up the available slots for connection requests and thereby deny legitimate access from completing.	► TCP/IP stack: Provides internal protection against SYN attack. ► IDS policy: May provide notification.

Attack policy notification

The IDS attack policy (object class name ibm-idsNotification) notification allows attack events to be logged to syslogd and the system console. For all attack categories except flood, a single packet triggers an event.

To prevent message flooding to the system console, you can specify the maximum number of console messages to be logged per attack category within a five-minute interval (ibm-idsMaxEventMessage). There is no default, therefore it is recommended that you code a maximum number of event messages that are to be written to the console. To prevent message flooding to syslogd, a maximum of 100 event messages per attack category will be logged to syslogd within a five-minute interval.

Note: The console messages provide a subset of the information provided in the syslogd messages.

Attack policy statistics

The IDS attack policy statistics action provides a count of the number of attack events detected during the statistics interval. The count of attacks is kept separately for each category of attack (for example, malformed), and a separate statistics record is generated for each.

If you want to turn on statistics for attacks, it is recommended that you specify exception statistics (ibm-idsTypeActions:EXCEPTSTATS). With exception statistics, a statistics record will only be generated for the category of attack if the count of attacks is non-zero. If statistics are requested (ibm-idsTypeActions:STATISTICS), a record will be generated every statistics interval regardless of whether an attack has been detected during that interval.

12.2.3 Attack policy tracing

IDS attack policy tracing uses the component trace facility SYSTCPIS. The attack policy tracing attributes are ibm-idsTraceData and ibm-idsTraceRecordSize, which indicate whether packets associated with the attack events are to be traced. For all attack categories except flood, a single packet triggers an event and the packet is traced. In the case of a flood, a maximum of 100 attack packets per attack category will be traced during a five-minute interval.

Note: In order to use the attack policy tracing via the ctrace component SYSTCPIS, the component must be started. See *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, for more information about this topic.

12.2.4 Traffic Regulation policies

The IDS Traffic Regulation (TR) policies are used to limit:

- Memory usage
- Queue delay time

There are two types of TR policies: TCP and UDP Traffic Regulation policies.

TR TCP policy information

The IDS TR policies for TCP ports limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as otelnetd. The TR TCP terminology is very important when coding the policy to ensure the desired goal is achieved. The following section describes TR TCP terminology.

Connections

Connections can be separated into two groups: total connections, and number of available connections.

Total connections

This is the total number of connections that are coded in your policy. This number can never be exceeded for a particular port.

Number of available connections

This is the total number of available connections, which is equal to the connections in use subtracted from the total connections. This value is used in the fair share algorithm.

Fair share algorithm

The fair share algorithm is designed to limit the number of connections available to any source IP address. The algorithm is based on the percentage of the available remaining connections for a particular port compared with the total connections already held by the source IP address for that port. The fair share equation and logic statements are shown in Example 12-1.

Example 12-1 Fair share logic

```
Equation Statement :
% SourceIPAddr = Num. of Conn. held by SourceIPAddr / Currently Available Sessions x 100

Logic Statement:
If %SourceIPAddr < Policy Percentage Then Allow the Session
Else Reject the
Session
```

Note: If a host does not currently have any connections open on the port and connections are available, a host will always be allowed at least one connection.

Quality of Service policy

Multi-user source IP addresses may be allowed a larger number of connections by specifying a Quality of Service (QoS) policy with a higher number of connections (MaxConnections) than allowed by the TR policy. TR will honor the QoS Differentiated Services Policy if the port is *not* in a constrained state. A QoS exception is made only when QoS Differentiated Services Policy is applied for the specific source server port and specific outbound client destination IP address.

Constrained state

TR TCP generates a constrained event when a port reaches approximately 90% of its connection limit (total connections). An unconstrained event is generated when the port falls below approximately 88% of its limit. This 2% deviance is designed to avoid message flooding.

TR UDP policy information

Traffic Regulation for UDP connections can be done in two ways: through the UDPQUEUELIMIT parameter in the TCPIP.PROFILE, or by coding a TR UDP policy. If both are in effect, the TR UDP policy takes priority.

UDPQUEUELIMIT

Traffic Regulation for UDP-based applications can be provided through the TCPIP.PROFILE statement of UDPQUEUELIMIT. This statement relates to inbound packets for bound UDP ports. Packets are queued until the queue limit is reached or buffer memory is exhausted. If NOUDPQUEUELIMIT is coded, any single bound port under a flood attack or with a stalled application could consume all available buffer storage. It is recommended that UDPQUEUELIMIT always be set to active. This limits the amount of storage that can be consumed by inbound datagrams for any single bound port. Sockets that use the Pascal API have a limit of 160 KB in any number of datagrams. Sockets that use other APIs have a limit of 2000 datagrams or 2880 KB.

Note: If a policy is in effect for a UDP port, the queue limit size is controlled by the policy for that port.

TR UDP policies

IDS TR policies for UDP ports specify one of four abstract queue sizes for specified bound IP addresses and ports. The four abstract sizes are VERY_SHORT, SHORT, LONG, and VERY_LONG. The abstract size comprises two values, the number of packets and the total number of bytes on the queue. If either one of these values is exceeded, inbound data is discarded. See Table 12-2 for the internal values.

Table 12-2 TR UDP abstract queue information

Abstract size	Number of packets	Queue limit
VERY_SHORT	16	32 KB
SHORT	256	512 KB
LONG	2048	4 MB
VERY_LONG	8192	16 MB

Most UDP applications have time-out values based on human perceptions of responsiveness. These values tend to stay constant while system processing speeds and network delivery speeds continue to advance rapidly. This may require the physical sizes of these queues to change over time. For performance reasons, sockets that use the Pascal API will only enforce the byte limit. Sockets that use other APIs will enforce both limits. Sockets without a policy specified for their port can use the existing UDPQUEUELIMIT mechanism.

For applications that can process datagrams at a rate faster than the average arrival rate, the queue acts as a speed-matching buffer that shifts temporary peak workloads into following valleys. The more the application processing rate exceeds the average arrival rate and the larger the queue, the greater the variation in arrival rates that can be absorbed without losing work. Very fast applications with very bursty traffic patterns may benefit from LONG or VERY_LONG queue sizes.

For applications that consistently receive datagrams at a higher rate than they are able to process, the queue acts to limit the effective arrival rate to the processing rate by discarding excess datagrams. In this case, the queue size only influences the average wait time of datagrams in the queue and not the percentage of work lost. In fact, if the wait time gets too large, the peer application may have given up or retransmitted the datagram before it is processed.

Slow applications with consistently high traffic rates may benefit from SHORT queue sizes. In general, client-side applications will tend to have lower system priority, giving them lower datagram processing rates. They also tend to have much lower datagram arrival rates. Giving them SHORT or VERY_SHORT queue sizes may reduce the risk to system buffer storage under random port flood attacks, with little impact on percentage of datagrams lost.

12.3 How IDS is implemented

IDS is implemented through policies. The Policy Agent is an integral part of setting up the environment for IDS to execute these policies. See Chapter 5, “Policy Agent” on page 221, for discussion and implementation examples for PAGENT.

The IBM Configuration Assistant for z/OS Communication Server graphical user interface (GUI) tool is the new flat file alternative for storing IDS policies. You may also change the IDS policy configuration file manually.

IDS comes with a set definition examples in the following file:

- /usr/lpp/tcpip/samples/pagent_IDS.conf

Using these samples as a base is a useful way to start your IDS implementation. To learn more about modifying these definitions to create customized policies, consult *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

12.3.1 Installing the Policy Agent

PAGENT reads the configuration files that contain the IDS policy configuration statements, checks them for errors, and installs them into the TCP/IP stack. Setting up the PAGENT is described in Chapter 5, “Policy Agent” on page 221.

Note: You need superuser authority to start PAGENT, and the PAGENT executable modules must be in an APF-authorized library.

After setting it up, you need to define the IDSConfig statement to specify the path of the policy file that contains stack-specific IDS policy statements to PAGENT. Example 12-2 shows the IDS statements in the TCP/IP stack configuration file used by Policy Agent.

Example 12-2 The /etc/sc30.tcpipa_image.conf file with IDS configured

```
# This is a file that contains statements for TCP/IP stack TCPIPA in z/OS image SC30
# This file is used by Policy Agent.
#
# IDSConfig Statements
IDSConfig //'TCPIP.SC30.POLICIES(IDSA)'
```

12.3.2 The IBM Configuration Assistant for z/OS Communication Server

The IBM Configuration Assistant for z/OS Communication Server enables centralized configuration of intrusion detection policies for z/OS V1R9. This solution provides an IDS policy solution that is consistent with other policy types for those installations that do not have an LDAP infrastructure in place, or that favor using configuration files instead of an LDAP configuration.

The IBM Configuration Assistant for z/OS Communication Server is a tool designed to allow a network administrator to produce a configuration file for the Policy Agent (PAGENT).

The IBM Configuration Assistant for z/OS Communication Server’GUI provides a user-friendly front-end for the entry of policy information. It also produces a configuration file with the information required by PAGENT. This file can be sent via FTP, or moved to and placed in the PAGENT configuration file manually.

The IBM Configuration Assistant for z/OS Communication Server is a tool for network administrators. Therefore, before you begin you should:

- Read the chapter on policy-based networking in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.
- Be familiar with your particular environment so that you can make decisions about what events are to be detected under what circumstances and what action to take.

12.3.3 Requirements and download instructions

This section outlines the requirements and support of the IBM Configuration Assistant for z/OS Communication Server.

Download and installation

The download and installation instructions are written for Windows. The following information and executables are located at:

http://www.ibm.com/support/search.wss?rs=852&tc=SSSN3L&rankprofile=8&dc=D410&dtm

12.3.4 Configuring IDS policy using the GUI

This section will help the network administrator manage and understand the GUI provided. Each first-level directory is discussed and captures are provided to assist in the education. The sections are:

- Creating a new IDS policy
- Adding a z/OS image to the policy
- Adding a TCP/IP stack under the z/OS image
- The default IDS requirement map settings
- Manually creating IDS objects and rules
- Creating reusable objects

Upon completion of this chapter, you will have created:

- Reusable objects
- A scan global policy
- An attack, scan event, and TR TCP (condition, action, and policy)

Creating a new IDS policy

To create a new IDS policy:

1. From IBM Configuration Assistant for z/OS Communication Server, click **File** -> **Open** -> **Create a New Configuration**.
2. In the information panel, click **OK**.
3. Enter a file name for the new configuration and click **Save**.
4. In the Welcome panel, select **Intrusion Detection Services (IDS)**, as shown in Figure 12-4 on page 484. Click **OK**.

> **Note:** IBM Configuration Assistant for z/OS Communication Server help is available via the Help menu option. If detailed information is needed for a particular field, click the **?** button and then click the specific panel or specific field in the panel in which you want to get help.

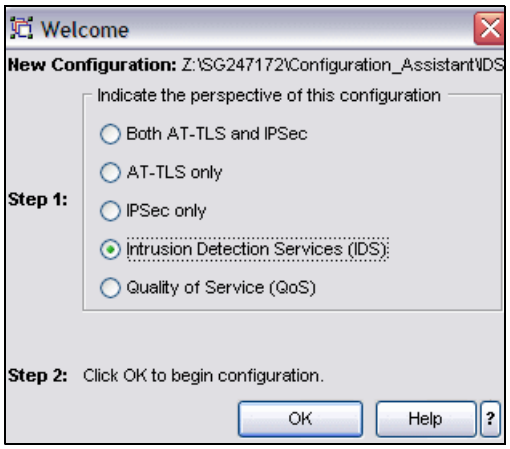


Figure 12-4 Welcome panel: Choosing to create an IDS configuration file

This brings you to the basic panel for configuring IDS policy, which is shown in Figure 12-5.

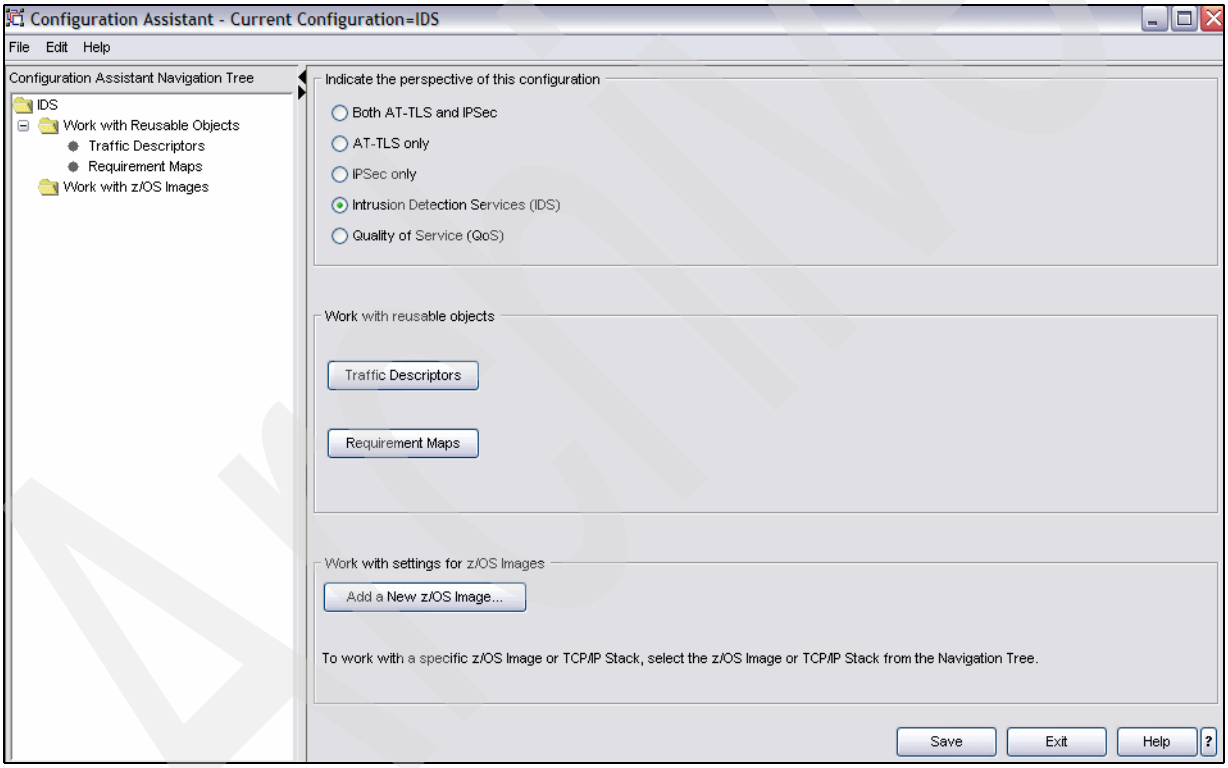


Figure 12-5 IBM Configuration Assistant for z/OS Communication Server first panel when choosing IDS

Adding a z/OS image to the policy

To add a z/OS image to the policy:

1. Click **IDS** -> **Work with z/OS Images** from the Configuration Assistant Navigation Tree. Click **Add a New z/OS Image**.
2. In the New z/OS Image: Information panel, enter the name of the z/OS image and a description, as shown in Figure 12-6. Click **OK**. Then click **Yes** to proceed to the next step.

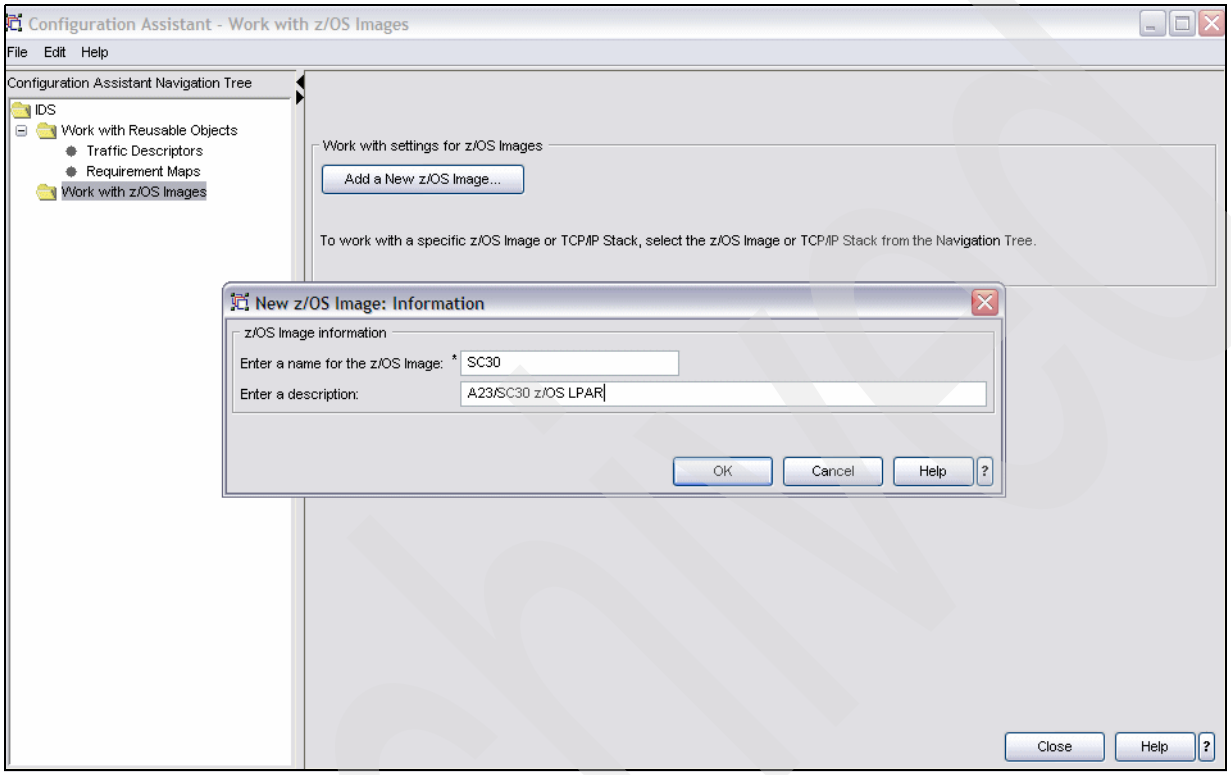


Figure 12-6 New z/OS Image: Information panel: adding SC30 z/OS image

Adding a TCP/IP stack under the z/OS image

In the New TCP/IP Stack: Name panel, enter the TCP/IP stack name and definition, as shown in Figure 12-7. Click **OK**.

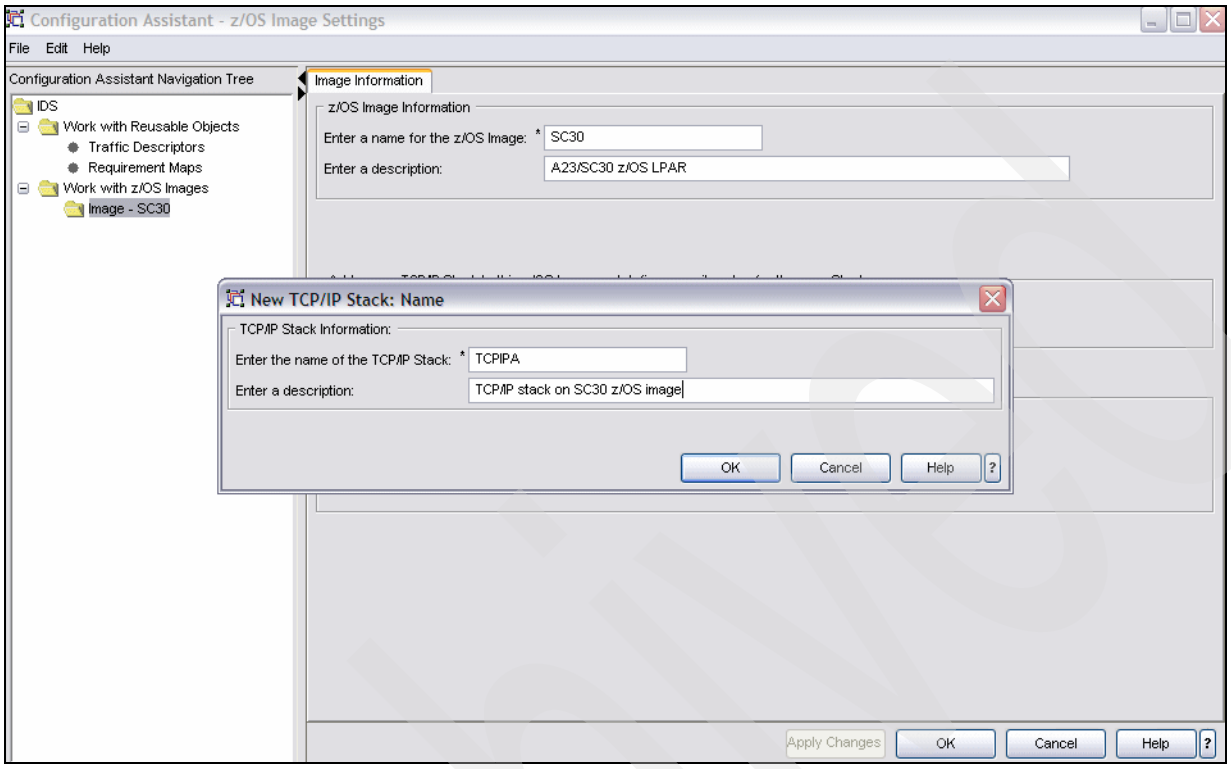


Figure 12-7 Adding a TCP/IP stack to the SC30 z/OS image

The default IDS requirement map settings

In the Proceed to the next step? panel, you have three options:

- Accept the default requirement map settings
- Get a tutorial about requirement maps
- Create a new requirement map with a wizard

We chose to use the default requirement map settings as a start. To see the default settings, click **View Details**.

Select **Accept the default Requirement Map settings**, as shown in Figure 12-8 on page 487. Click **OK**.

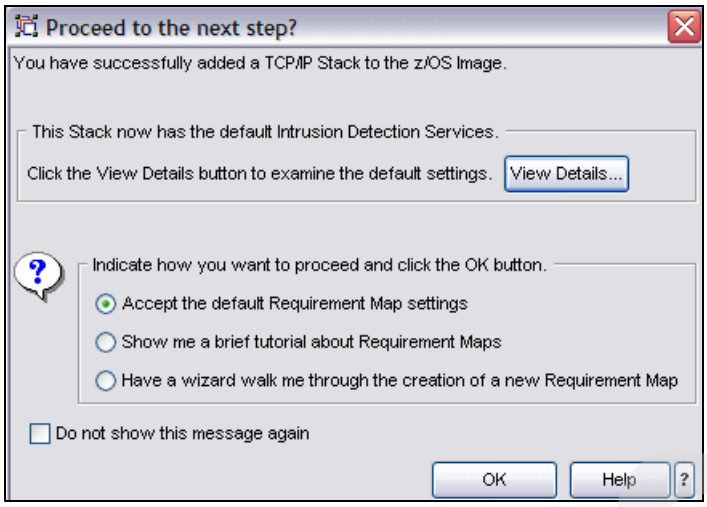


Figure 12-8 Proceed to the next step? panel: choosing the default settings

This should bring you to the panel shown in Figure 12-9.

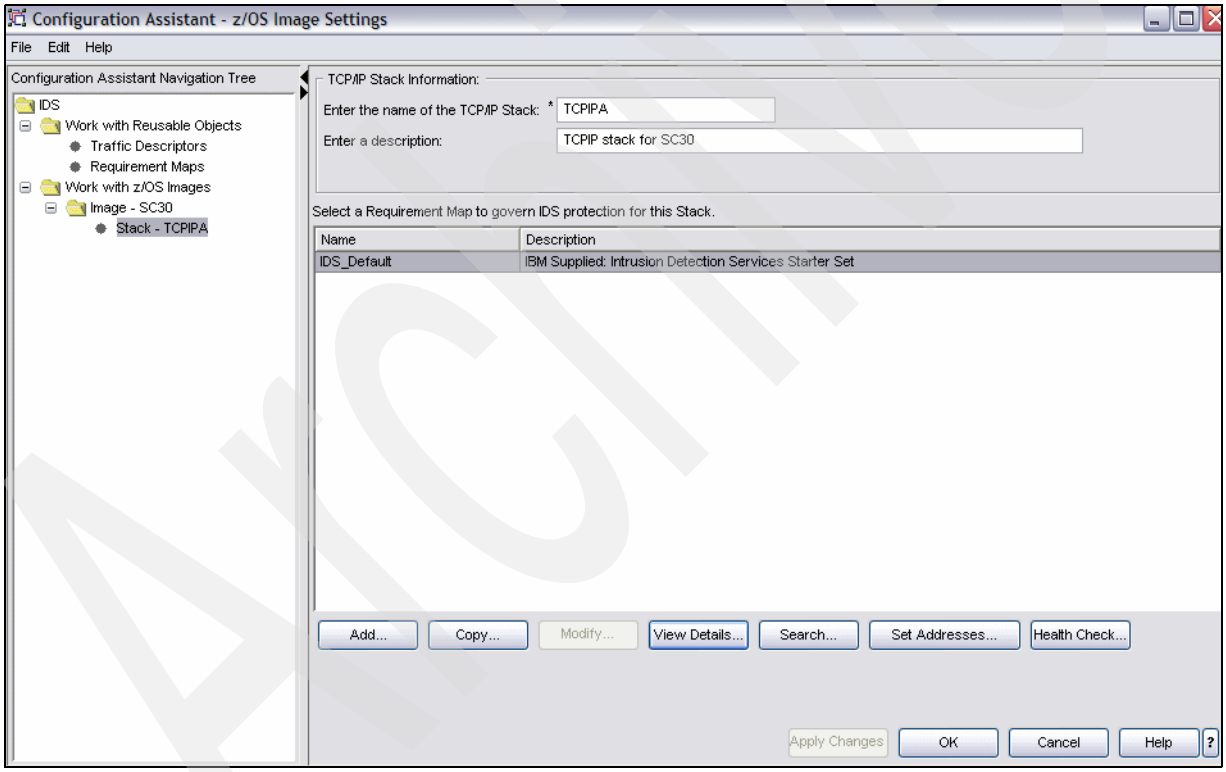


Figure 12-9 z/OS Image Settings panel: after defining the default settings for IDS policy

Manually creating IDS objects and rules

We have now arrived at the most critical task: defining the IDS policy rules. This task is typically done iteratively until the final policy rules are defined.

- You are required to establish one condition set and one action set in at least one policy rule.
- You may optionally specify that those rules apply only during validity periods.

Use this section to specify IDS policy rules, which can include condition sets, actions, policy keyword sets, or validity periods. Only one policy rule and associated actions can be applied to a particular packet.

Use the following steps to create an IDS policy rule:

1. Create reusable objects that will be used in your action and condition sets.
2. Create the actions and conditions based on those reuseable objects.
3. Build your policy rule from the available condition and action sets.

The following sections walk you through this process.

When you finish specifying IDS policy rules, click **IDS** -> **Work with z/OS images** -> **Image -** ***image_name*** -> **Stack -** ***stack_name*** from the Configuration Assistant Navigation Tree to store the policy information into z/OS image.

Creating reusable objects

The reuseable objects are designed to be incorporated into multiple policies, conditions, or actions, depending on the type of object.

There are two types of reusable objects:

Traffic Descriptors Reusable objects which describe properties of network traffic such as protocols and ports.

Requirement Maps Reusable objects which are entire sets of intrusion detection requirements. They contain, by default, only the default requirement map settings object.

Figure 12-10 illustrates some of the traffic descriptors that are available in the Work with Reusable Objects - Traffic Descriptors panel.

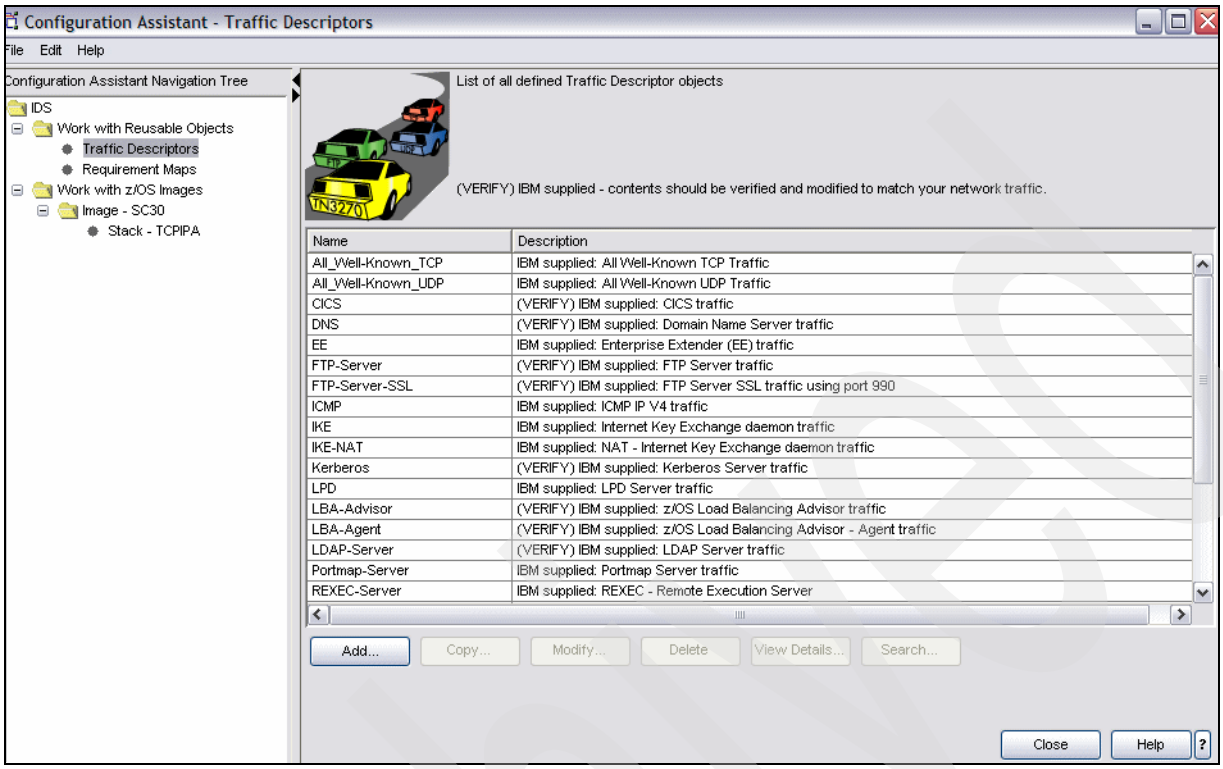


Figure 12-10 Traffic Descriptors for IDS policy

Creating a new IDS policy

The next step is to configure an attack protection.

1. Select **IDS** -> **Work with Reusable Objects** -> **Requirement Maps** from the Configuration Assistant Navigation Tree.
2. In the Configuration Assistant - Requirement Maps panel, click **Add**.
3. In the Add Requirement Map: Name panel, enter a name and description, as shown in Figure 12-11 on page 490. Click **Next**.

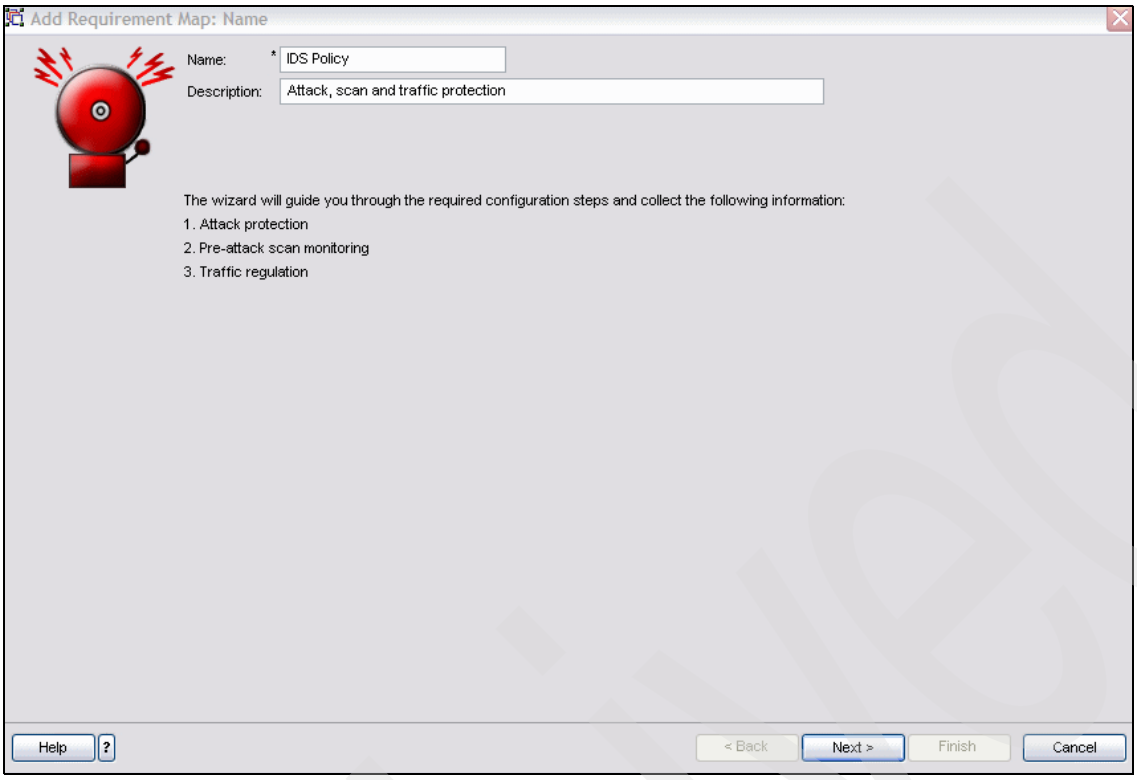


Figure 12-11 Add Requirement Map: Name - Adding an attack protection rule

Attacks

An attack can be a single packet designed to crash or hang a system, or multiple packets designed to consume a limited resource causing a network, host system, or application to be unavailable to its intended users (a denial of service). The IDS attack policy lets you turn on attack detection for one or more categories of attacks independently of each other. In general, the types of actions that you can specify for an attack policy are event logging, statistics gathering, packet tracing, and discarding of attack packets.

In the Add Requirement Map: Attacks panel, verify that the **Enable attack protection** check box is selected.

There are eight attack types that you can enable a protection from: flood, perpetual echo, unwanted IP protocols, unwanted IP options, ICMP redirect, malformed packet, outbound raw, and IP fragment.

You may modify and change the specific values of only five of these attack types: flood, perpetual echo, unwanted IP protocols, unwanted IP options, and outbound raw.

We chose to leave the default configuration supplied with the IBM Configuration Assistant for z/OS Communication Server, as shown in Figure 12-12. Click **Next**.

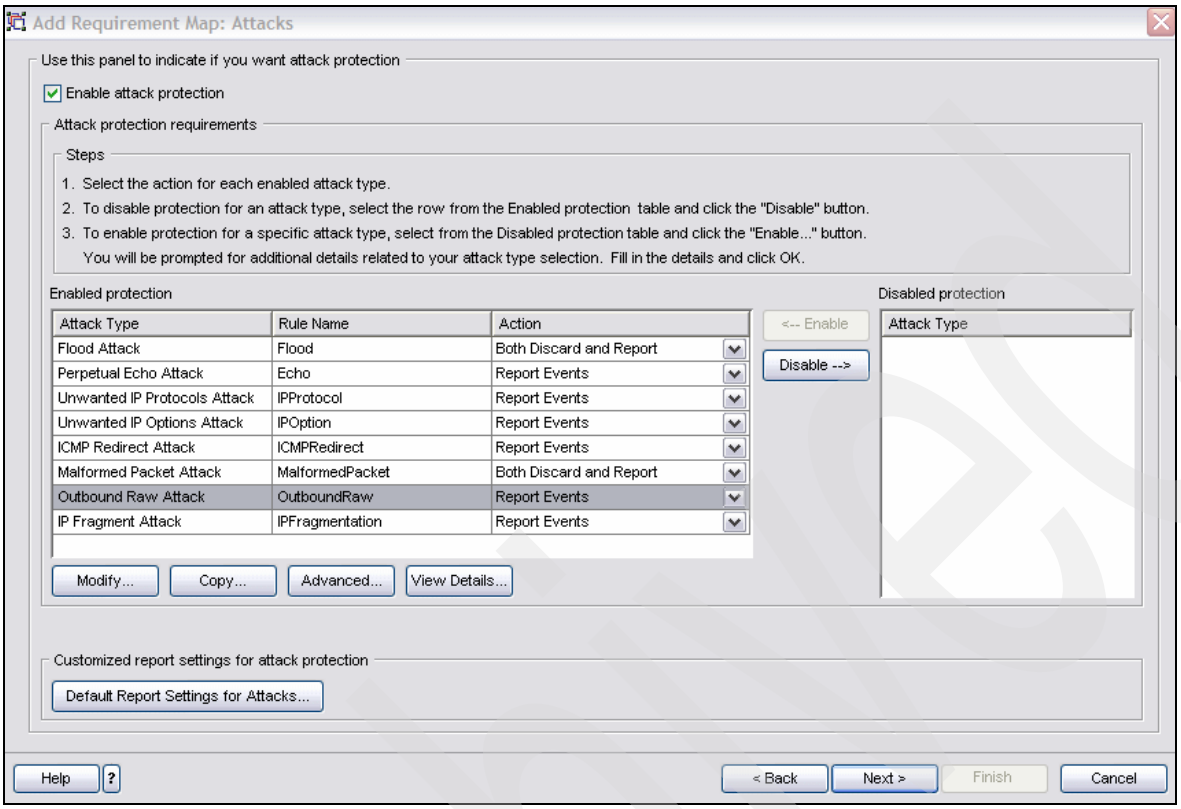


Figure 12-12 Add Requirement Map: Attacks - configuring attack protection

For information about the attack types, click the **?** button and then place the cursor in this field and click it.

Scans

You can specify sets of global scan detection parameters (threshold and interval for fast and slow scans). These attributes apply to all scan events. If you configure a certain category of scan events, the action will be triggered if the number of those events received from one IP address exceeds the slow scan threshold during the slow scan interval.

Similarly, if you configure a certain category of scan event, the action will be triggered if the number of those events received from one IP address exceeds the fast scan threshold during the fast scan interval. The slow scan threshold must be greater than the fast scan threshold. The slow scan interval must be greater than the fast scan interval.

1. In the Add Requirement Map: Scans panel, select **Enable scan**.

   Notice that there are default values for the Fast Scan Interval, Fast Scan Threshold, Slow Scan Interval, and Slow Scan Threshold fields. You can see the default values by clicking the **Modify Fast and Slow Scan Settings** button and then clicking **OK**. We accepted the default values.

Scan events come from the following categories:

- ICMP scans: ICMP requests (echo, information, time stamp, and subnet mask) are used to map network topology. Any request sent to a subnet base or broadcast address will be treated as very suspicious. Echo requests (PING) and time stamp requests are normal, unless they include the Record Route or Record Timestamp option, in which case they are possibly suspicious.
- TCP port scans: Because TCP is a stateful protocol, many different events may be classified as normal, suspicious, or highly suspicious. For more details, refer to “Scan policies” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.
- UDP port scans: A datagram received for a restricted port is very suspicious; one received for an unreserved but unbound port is possibly suspicious; and one received for a bound port is normal.

The individual packets used in a scan can be categorized as normal, possibly suspicious, or very suspicious. To control the performance impact and analysis load of scan monitoring, you can adjust your interest level in potential scan events. If you set the sensitivity level to:

High	Normal, possibly suspicious, and very suspicious events will be counted.
Medium	Possibly suspicious and very suspicious events will be counted.
Low	Only very suspicious events will be counted.
None	No events will be counted.

For more information about the sensitivity, click the **?** button and then click the **Sensitivity** field.

We accept the default configuration of enabled scans, as shown in Figure 12-13 on page 493.

Note: There is an importance to the order of the enabled scans list. Packets are checked against the rules in the order they appear in the table.

Click **Default Report Settings for Scans**.

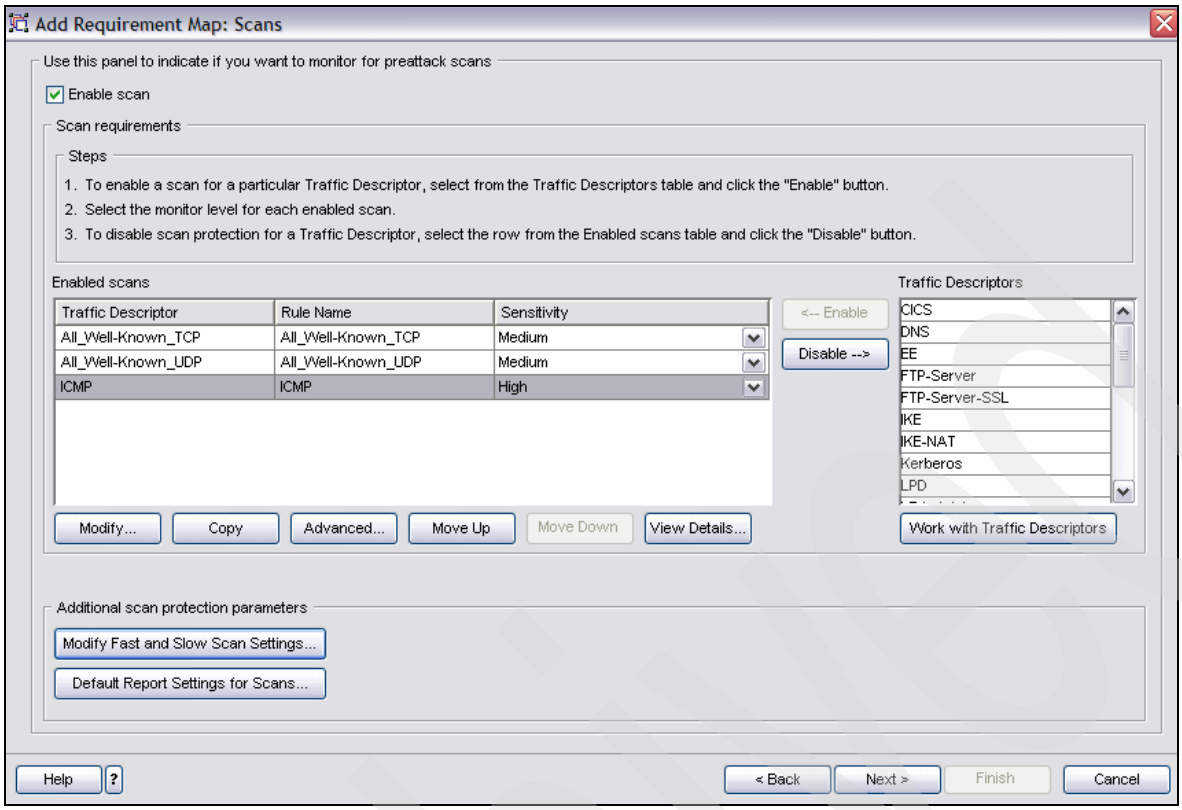


Figure 12-13 Add Requirement Map: Scans panel - choosing the default-enabled scans

2. In the Report Types panel, you indicate where to report IDS events. We selected **System console** and **SYSLOGD**, as shown in Figure 12-14. Click **Modify Details**.

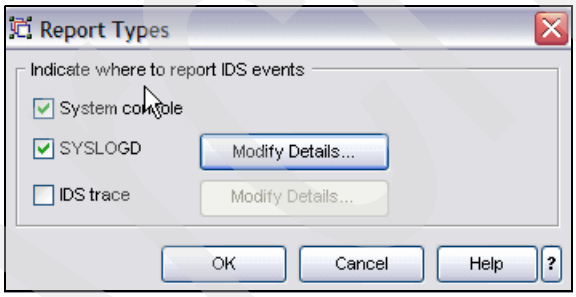


Figure 12-14 Report Types panel - indicating where to report IDS scan events

3. When you are configuring your IDS policy, select a high log level. We selected **6 - info**, as shown in Figure 12-15.

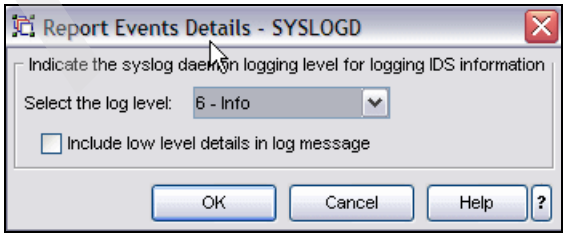


Figure 12-15 Report Events Details - SYSLOGD panel - selecting a log level

In the Report Events Details panel, click **OK**.

4. In the Report Types panel, click **OK**.
5. In the Add Requirement Map: Scans panel, click **Next**.

Traffic Regulation

Traffic Regulation (TR) policies are used to limit memory resource consumption and queue delay during peak loads. TR policies for TCP ports can limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as FTPD and otelnetd. A fair share algorithm is also provided based on the percentage of remaining available connections held by a source IP address.

IDS policies for UDP ports specify a queue length. Longer queues let applications with higher processing rate capacity absorb higher bursts of traffic. Shorter queues let applications with lower processing rate capacity reduce the queue delay time of packets that they accept.

1. In the Add Requirement Maps: Traffic Regulation panel, select the **Enable traffic regulation** check box.

   From the Traffic Descriptors, select **All_Well-Known_TCP** and click **Enable** as shown in Figure 12-16.

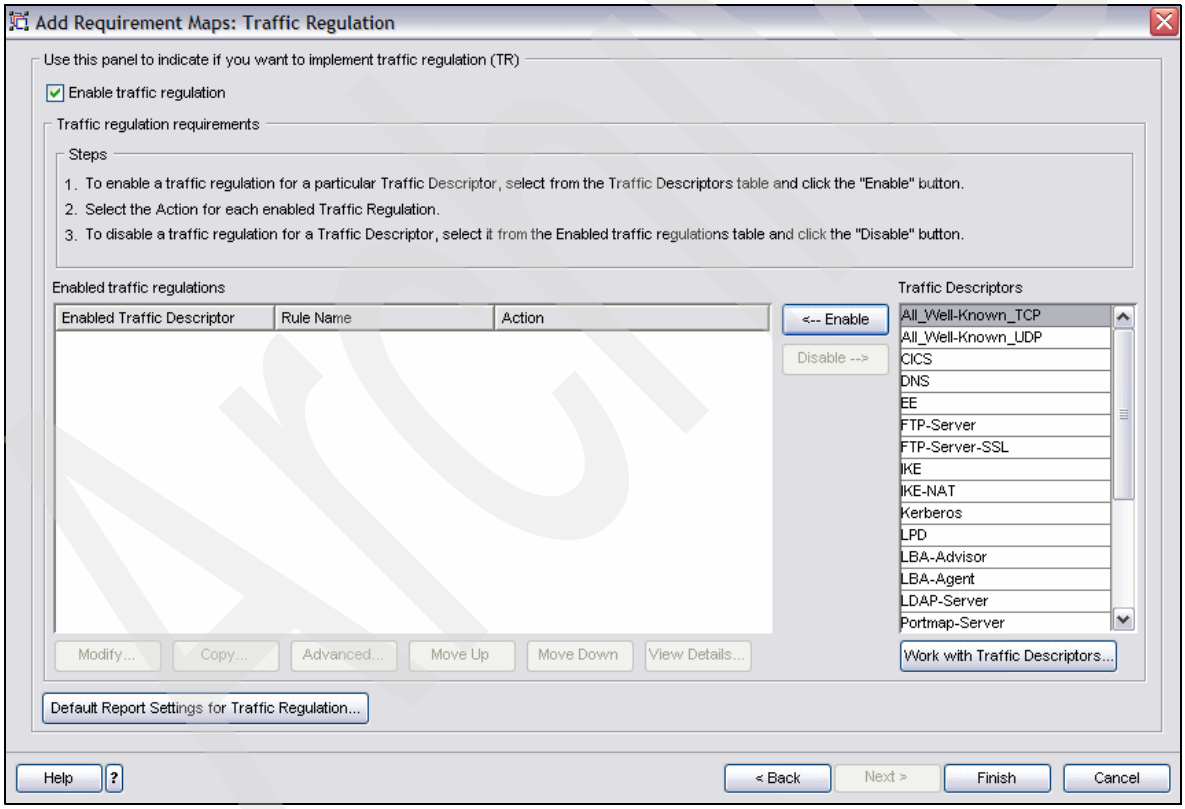


Figure 12-16 Add Requirement Maps: Traffic Regulation panel - adding All_Well-Known_TCP

2. In the Traffic Regulation Details panel, we accepted the default values, as shown in Figure 12-17. However, you should consider the following in your environment:
   - For TCP, cap the number of connections, or the number of connections any one user can have with an application; for example:
     - Limit by total connections: Limit the size of the total connection pool for IDS TCP traffic regulation functions.
     - Limit by percentage: Limit the percentage of the total connections that can be used by a single host.
     - Limit by socket or by all sockets: Determine whether the limits should be applied to each socket, or to the aggregate of all sockets for each port. Note that you cannot specify local host addresses for the traffic regulation rule if you choose to limit connections by the aggregate of all sockets for the port.
   - For UDP, cap the number of queued packets, for example:
     - Select one of a number of abstract queue sizes that map to internally-defined limits. These sizes are generally defined for each application as a function of response time, and are subject to change over time.

   Click **OK** and then click **Finish**.

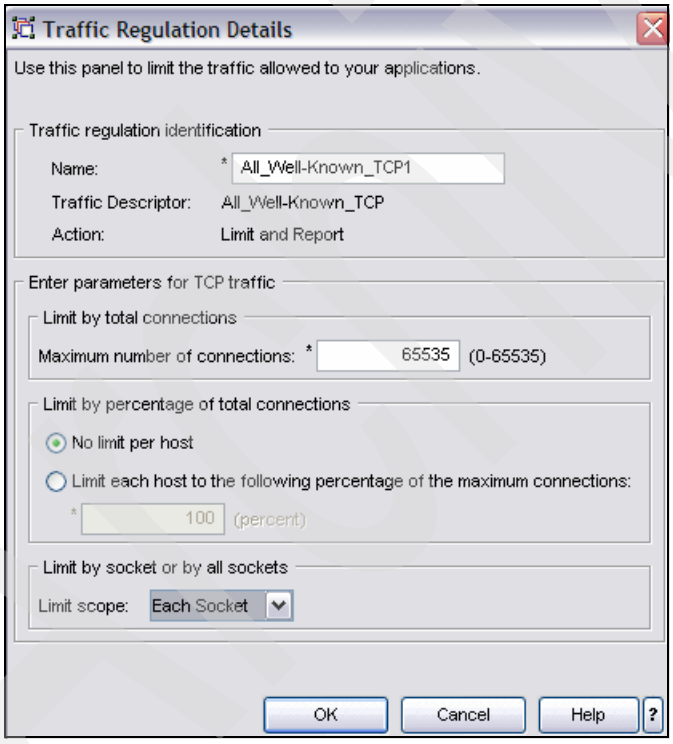


Figure 12-17 Traffic Regulation Details panel

Scan remote exclusions and local addresses

Setting scan remote exclusions and local addresses is optional. A scan exclusion consists of one or more scan exclusion range attributes that specify known legitimate scanners. To reduce false positives (that is, undesirable reports of scans by legitimate scanners), you can specify source IP addresses, a subnet mask length, and source port numbers of sources that you trust to be excluded from scan detection.

To do this, perform the following steps:

1. Select **IDS** → **Work with z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** from the Configuration Assistant Navigation Tree.
2. From the Configuration Assistant - Requirement Maps panel, select **IDS_Policy** from the requirement map list, as shown in Figure 12-18. Click **Set Addresses**.

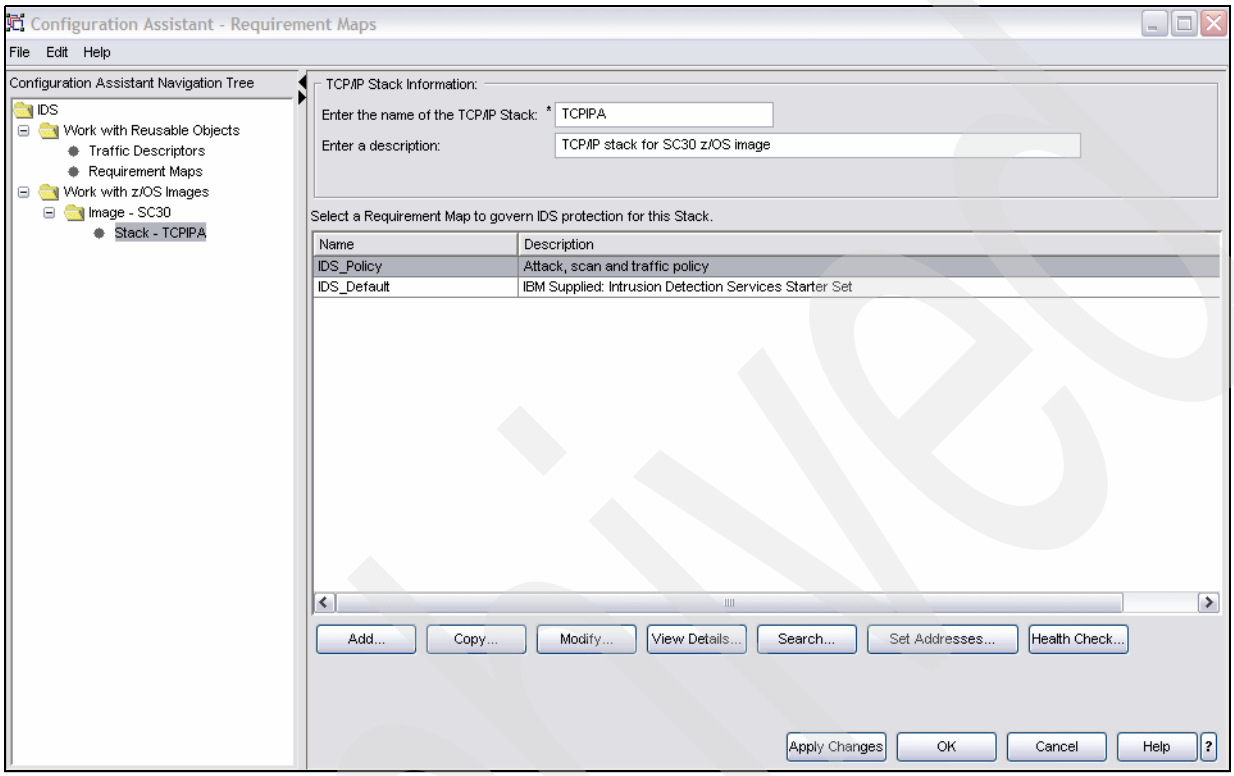


Figure 12-18 Configuration Assistant - Requirement Maps for stack TCPIPA

3. In the Advanced Stack Settings panel, select the traffic descriptor for which you want to set remote exclusions and local addresses and click **Set Remote Exclusions and Local Addresses**.
4. In the Stack Level Scan Settings panel, select **Exclude the following remote addresses and ports** and click **Add**.
5. In the Scan Exclusion Remote Details panel, enter the remote exclusion address and port. Click **OK**.

   We did not add any exclusion, therefore we clicked **Cancel**.
6. If you want to set local addresses for which the rule will apply, select the **Local Addresses** tab from Stack Level Scan Settings panel, and then select **Rule applies to only the following specified local addresses**. Click **Add** and enter the local IP address. Click **OK** to save the changes. We clicked **Cancel**, because we do not want to set local addresses.
7. In Stack Level Scan Settings panel, click **OK**.
8. In Advanced Stack Settings panel, click **OK**.

12.3.5 Installing the IDS policy

After you finish configuring the IDS policy, perform these steps:

1. Right-click **IDS** -> **Work with z/OS Images** -> **Image - *image_name*** -> **Stack - *stack_name*** from the Configuration Assistant Navigation Tree.

   From the drop-down menu, select **Install Configuration FIles**, as shown in Figure 12-19.

> **Tip:** Click the **Health Check...** button from the Connectivity Rules tab before you install the configuration file to the z/OS image. The Health Check reports on errors in your policy, which we found helpful.

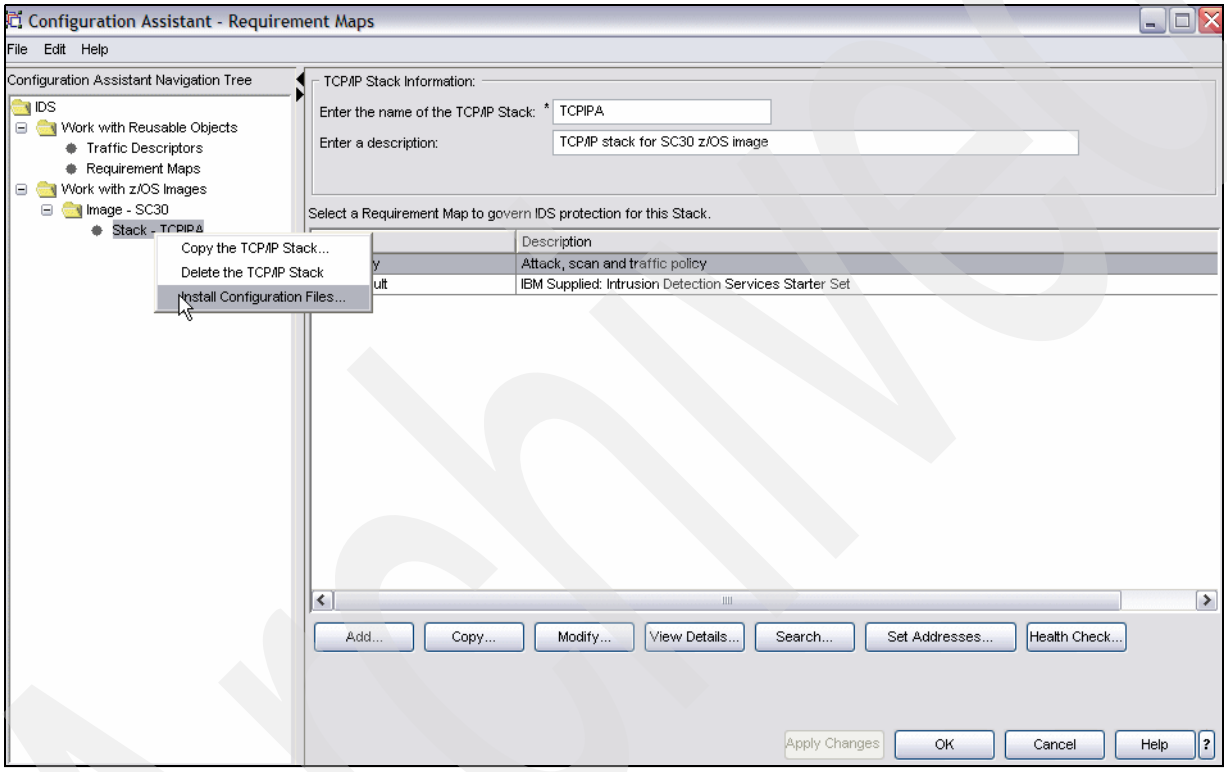


Figure 12-19 Configuration Assistant - Requirement Maps panel - installing configuration files

2. In the Installation - Stack = “stack_name” panel, select the ***stack_name* - IDS: Policy Agent Stack Configuration** from the list, as shown in Figure 12-20. Click **FTP**.

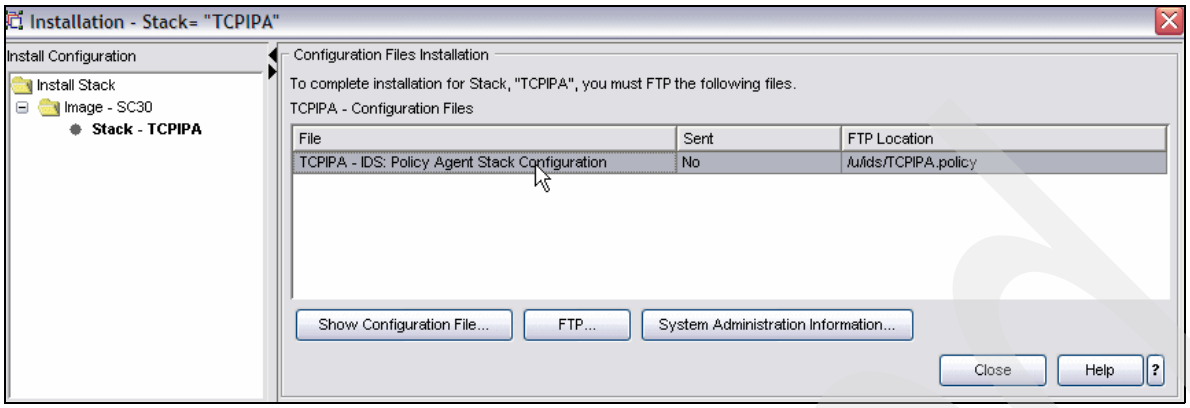


Figure 12-20 Installation - Stack panel - installing the IDS configuration file

3. In the FTP Configuration File panel, enter the host name, the FTP port (usually 21), a user ID and its password. In addition, enter the file name and location of the IDS policy. This file name and location should be the same file name and location that are mentioned in the TCP/IP stack configuration file, which is used by Policy Agent, as shown in Figure 12-21. Click **Send**.

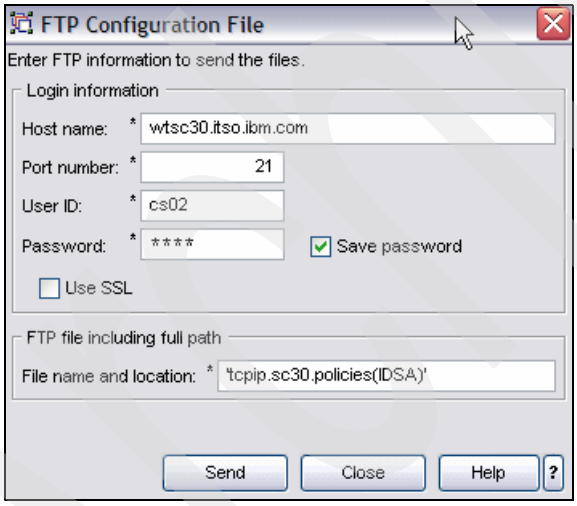


Figure 12-21 FTP Configuration File - sending the IDS policy to the SC30 z/OS image

4. After you receive the message `The FTP transfer was successful`, refresh the Policy Agent by running the console command:

   ```
   MODIFY PAGENT,REFRESH
   ```

   You should receive the following log messages:

   ```
   EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
   EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IDS
   ```

12.3.6 Checking that things are working

To verify that your policy is working, use tools which create the specific attacks that the policy is supposed to provide protection against.

For example, to check that the scan policy works, you can download a tool from the Web that scans ports. Then follow these steps:

1. Run a ports scan on the z/OS image IP address.
2. Run the NETSTAT IDS command to get an IDS summary of scan, attack, and traffic regulation detected.

Note: The user who invokes the NETSTAT IDS command must be permitted for READ access to the resource name: EZB.NETSTAT.mvsname.tcpprocname.IDS. For more information about the NETSTAT command, refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

In our test environment, we used a free utility that scans ports on a specific address. The console log messages shown in Example 12-3 appeared on the console.

Example 12-3 Console messages after running a fast ports scan from client on z/OS IP address

```
EZZ8762I EVENT TYPE: FAST SCAN DETECTED
EZZ8763I CORRELATOR 9 - PROBEID 0300FFF1
EZZ8764I SOURCE IP ADDRESS 192.168.1.254 - PORT 0
EZZ8766I IDS RULE ScanGlobal
EZZ8767I IDS ACTION ScanGlobalAction
```

We then issued the NETSTAT IDS command, which gave the output shown in Example 12-4. (We show only the part of the output that relates to scan detection.)

Example 12-4 Output of NETSTAT IDS command after running the ports scan

```
Intrusion Detection Services Summary:
Scan Detection:
  GlobRuleName: ScanGlobal
  IcmpRuleName: ICMP~1
  TotDetected:  1           DetCurrPlc: 1
  DetCurrInt:   0           Interval:   30
  SrcIPsTrkd:   1           StrgLev:    00000M
```

12.3.7 Additional information

In this section, we provide additional information, including a summary of common mistakes and logging.

NetView and z/OS IDS

NetView® z/OS V5R1, PTF UA11043, provides management support for z/OS Communications Server IDS. It provides the ability to:

- Trap IDS messages from the system console or syslogd and take predefined actions based on IDS event type.
- Route IDS messages to designated NetView consoles.
- Provide e-mail notifications to security administrators (including running trmdstat and attaching the output to the e-mail).
- Issue predefined commands.

IDSAUTO is a set of NetView REXX clists and automation table entries that automates IDS messsages and performs notifications and reporting via e-mails. These clists can be downloaded from the following Web site:

http://www.ibm.com/support/docview.wss?uid=swg24001743

Tivoli Risk Manager and z/OS IDS

You are able to send TEC events to IBM Tivoli® Risk Manager (V4R1 or later) for enterprise-wide correlation and analysis of intrusion events.

The format file provided by z/OS Communications Server to convert syslog messages to TEC events is available at:

http://www-1.ibm.com/support/docview.wss?uid=swg24006973

13

Policy-based routing

Policy-based routing provides a routing option to send traffic based on not only the destination IP address, but also on additional traffic criteria such as TCP port number, jobname, and source IP address. Using policy agent, the traffic criteria and policy-based routing tables are defined to the stack for optimized routing.

This chapter discusses the implementation of policy-based routing for z/OS Communications Server.

The following topics are discussed in this chapter.

Section	Topic
13.1, “Policy-based routing concept” on page 502	Basic concepts of policy-based routing
13.2, “Routing policy” on page 503	Configuration options for routing policies
13.3, “Implementing policy-based routing” on page 505	Implementation procedure for policy-based routing

13.1 Policy-based routing concept

When TCP/IP sends a packet, route selection is based on the destination IP address of the packet. When multiple routes to the same destination exist in the routing table, they are referred to as a “multipath group” and their use is controlled by the MULTIPATH/NOMULTIPATH value specified on the IPCONFIG statement in the TCP/IP profile.

When IPCONFIG NOMULTIPATH is coded, the first active route in the multipath group is used for all traffic to that destination. When IPCONFIG MULTIPATH is coded, traffic to that destination is distributed across all routes in the multipath group.

Because route selection is based on the destination IP address of the packet, all packets destined to the same IP address, including interactive traffic like TSO on Enterprise Extender and batch traffic like FTP, have to use the same route or set of routes. You may, however, have a requirement to separate different types of traffic sent to that destination. For example, you may need to ensure that the interactive traffic will not be affected by the congestion of the batch traffic. Prior to z/OS V1R9 Communications Server, the only way to accomplish this was to run the application in different LPARs or TCP/IP stacks and use separate route tables.

z/OS Communications Server provides a policy-based routing (PBR) function that allows the packet route to be selected based on criteria such as source or destination TCP/UDP port, jobname, and source IP address, Netacess security zone, and Multi-level security (MLS) label. With PBR, you can differentiate the routes for the outgoing traffic depending on its type, and optimize the network routing.

PBR utilizes the policy agent to define the traffic criteria and routing tables. In addition to the existing main routing table, up to 255 PBR tables can be added. The PBR table may contain static routes, dynamic routes, a combination of static, and dynamic routes. OMPROUTE manages the dynamic routes in a PBR table when dynamic routing is used.

Each set of traffic criteria defined to the policy agent specifies the set of routing tables (PBR tables and optionally, the main routing table) to be used for routing outbound traffic that matches the criteria. When an outbound packet matches the defined criteria, these routing tables are searched for a usable route as follows:

- The first PBR table is searched for a usable route to the destination IP address. This may be a host, subnet, network, supernet, or default route. If a usable route is found, it is used.
- If there is no usable route in the first PBR table, the next PBR table (if specified) is searched. Up to 8 PBR tables can be associated with a set of criteria and each is searched in the defined order, until a usable route is found.
- If there is no usable route found in any of the PBR tables, the main routing table may be searched for a usable route. This is only done when the definition for the matched traffic criteria specifies that the main routing table should be used as a backup to the PBR tables.

If an outbound packet does not match any criteria defined in the policy agent, then the main routing table is used.

Note: PBR only controls outgoing packets and applies only to IPv4 TCP and UDP traffic. Other traffic such as ICMP Ping, Traceroute, or IPv6 traffic must use the main routing table.

Also, PBR is supported for locally-created traffic only. All forwarded packets, including Sysplex Distributor traffic, continue to use the main routing table.

Avoid using policy-based routing in a CINET environment unless all applications establish affinity with a particular TCP/IP stack, or the routes in each TCP/IP stack route table are mutually exclusive with the routes on the other TCP/IP stacks, including the default route. Otherwise, CINET may not select the best stack for the connection, because it only has information from the main routing table of each stack.

13.2 Routing policy

The PBR policy is configured in a policy agent flat-file. It includes the following definitions.

- Routing rules
- Routing actions
- Routing tables

IBM Configuration Assistant for z/OS Communication Server can be used to generate the PBR configuration flat file, or you can code it directly to the flat file. We recommend using IBM Configuration Assistant for z/OS Communication Server to avoid syntax errors.

Note: LDAP is not supported for PBR policies.

Routing rules

A *routing rule* specifies a set of criteria for outbound traffic. Specifying one or more of the following options identifies the traffic to which the policy-based routing rule applies:

- Traffic descriptor
  - Source IP address (single IP address or range/group of IP addresses)
  - Destination IP address (single IP address or range/group of IP addresses)
  - Source port (single port or port range)
  - Destination port (single port or port range)
  - Protocol (TCP or UDP)
  - Jobname of the sending application (specific jobname or wild card)
  - Netaccess security zone
  - Multi-level security (MLS) label

Note: The source IP address for outbound packets can be influenced by a number configuration and application options. Do not use the source IP address as a traffic descriptor when the traffic relies on one of the following methods to select its source IP address:

- SOURCEVIPA: static VIPA address from the HOME list
- HOME IP address of the link over which the packet is sent

These methods select a route before the source IP address is determined. Therefore, the source IP address is zero (0) when route lookup is performed.

- Priority

  If a packet can match the traffic descriptor specified for more than one rule, the priority should be used to ensure that the intended rule is applied for the packet. Priority can be specified from 1 to 2,000,000,000. The default is 1 (the lowest priority).

  Note: Priority is not explicitly configured when the Configuration Assistant is used to generate the routing configuration file. The rule priority is determined by the order of the rules as shown on the rules panel.

- Time condition

  The time condition specifies when the routing rule is to be active. It is possible to specify a specific date, a date range, or a mask for months of the year and days of the week.

Routing actions

A *routing action* specifies which PBR routing tables are to be used for the traffic that matches a routing rule, and the searching order of the tables.

A routing action also defines whether the main routing table should be used as a backup when no usable route to a packet's destination IP address is found in any of the specified PBR routing tables.

Routing tables

PBR routing tables can include static routes, dynamic routes, or the combination of both. Up to 255 routing tables can be defined in a TCP/IP stack. However, up to eight routing tables can be specified for a specific routing action. Only active PBR routing tables are installed in the TCP/IP stack. A PBR routing table is considered active if it is referenced by an active routing rule.

When selecting a route from a PBR table, the precedence rules are the same as for the main routing table:

- If a route exists to the destination address (a host route), it is chosen first.
- If subnet, network, or supernetwork routes exist to the destination, the route with the most specific network mask (the mask with the most bits on) is chosen second.
- If the destination is a multicast destination and multicast default routes exist, the one with the most specific multicast address is chosen third.
- Default routes are chosen when no other route exists to a destination.

To define a static route, you specify:

- Destination IP address
- Next hop IP address
- Link name of the sending interface
- MTU size
- Whether the static route can be replaced by a dynamic route learned from OMPROUTE.
- Replaceable or non-replaceable by dynamic route (if both static and dynamic routes are defined in the policy-based routing table)

A dynamic routing entry provides parameters for OMPROUTE to use when generating dynamic routes for the route table. To define a dynamic routing entry, you specify:

- Link name of the sending interface
- First hop IP address (optional)

OMPROUTE uses the dynamic routing entries defined for the table to control the dynamic routes added to the table. The routes added are the best routes learned by OMPROUTE that use the interfaces and optional first hops specified in the defined dynamic routing entries.

The routing table name is case sensitive and must be 1 to 8 characters in length. Lower case letters can be used for the name. However, we recommend defining the name using all upper case letters to avoid confusion. This is because z/OS commands display the routing table name in all upper case letters, regardless of whether it is defined in lower case or upper case. The existing main route table is called EZBMAIN. The names EZBMAIN and ALL (in upper, lower, and mixed case) are reserved.

Note: To avoid performance issues, duplicate routing tables that contain the same routing entries should be avoided.

Advanced parameters for routing table definition are provided, as explained here:

- Multipath specifies the multipath algorithm that should be used with the policy-based routing table.
  - Use global (as defined in the IPCONFIG statement in TCP/IP profile)
  - Per connection
  - Per packet
  - Disable (use only the first active route to a destination)
- Ignore Path MTU Update indicates whether IPv4 ICMP Fragmentation Needed messages should be ignored for this route table.
- Dynamic XCF Routes indicates whether direct routes to dynamic XCF addresses on other TCP/IP stacks should be added to this route table.

For further information, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Note: There are considerations with FLUSH/NOFLUSH and PURGE/NOPURGE policy agent options for PBR, as explained here:

- FLUSH/NOFLUSH
  - The NOFLUSH option is not supported.
  - Routing policies are always deleted prior to installing new policies at the following times:
    - Policy agent startup.
    - TcpImage/PEPInstance statement added.
    - MODIFY,REFRESH command issued.
- PURGE/NOPURGE
  - The PURGE option is not supported.
  - Routing policies are never deleted during policy agent shutdown or when a TcpImage/PEPInstance statement is deleted.

To remove all routing policies from a TCP/IP stack, delete the RoutingConfig statement from the policy agent image configuration file for the stack.

13.3 Implementing policy-based routing

In this section we show two scenarios for implementing policy-based routing:

- Using jobname, protocol, and destination IP address as the traffic descriptor.
- Using port numbers as the traffic descriptor.

Both scenarios use dynamic routing entries for the policy-based routing table. In scenario 1 (based on jobname and destination IP address), we configured a mainframe-to-terminal connection. In scenario 2 (based on port numbers), we configured a mainframe-to-mainframe connection.

IBM Configuration Assistant for z/OS Communication Server was used to configure the policy rules for both scenarios. The generated configuration file is read by the policy agent at startup time.

To implement our policy-based routing scenarios, we performed these tasks.

1. Set up the policy agent and add authorization for the **`pasearch`** command in RACF:

   The policy agent was set up as described in 5.2, “Implementing PAGENT on z/OS” on page 229.

2. Configure the PBR policies:
   - Policy-based routing using jobname, protocol, and destination IP address
   - “Policy-based routing using protocol and port numbers” on page 522

13.3.1 Policy-based routing using jobname, protocol, and destination IP address

In this scenario we implement policy-based routing to force all TN3270 traffic to use a specific OSA link. TCPIPD in Figure 13-1 illustrates our policy-based routing scenario.

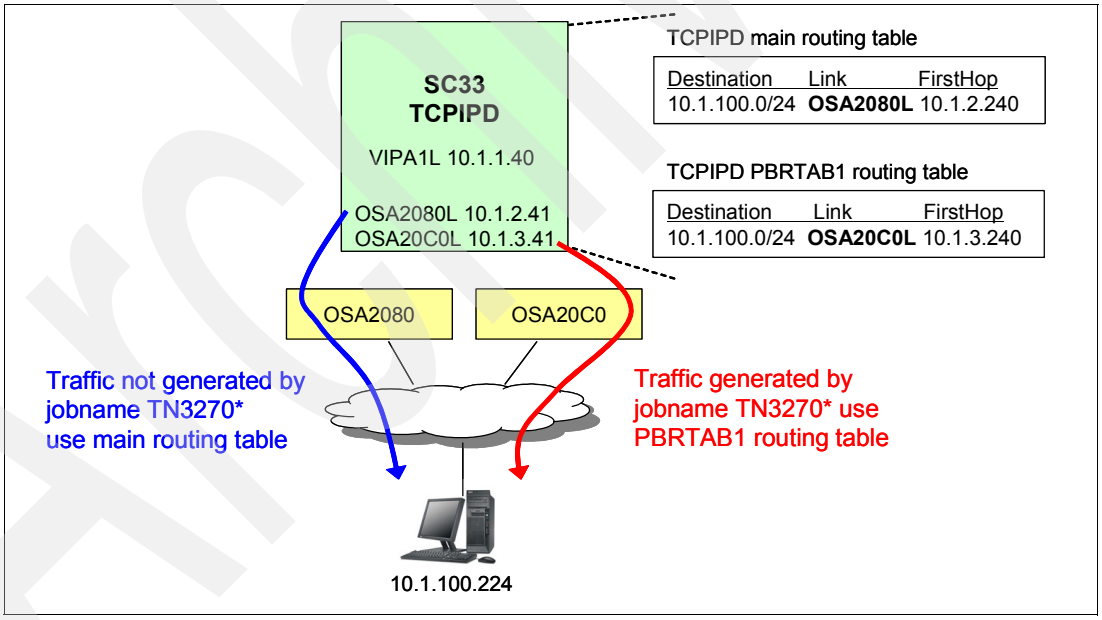


Figure 13-1 Policy-based routing using jobname, protocol, and destination IP address

Implementation tasks

We implemented policy-based routing using the following steps:

1. Configure the PBR policy with jobname, protocol, and destination IP address.
2. Upload the generated PBR policy file to z/OS.
3. Update the Policy Agent configuration file.
4. Start the Policy Agent.

No changes were required for the TCP/IP profile or the OMPROUTE profile.

Configure the PBR policy with jobname, protocol, and destination IP address

1. Start the IBM Configuration Assistant for z/OS Communication Server. In the Main Perspective panel, click the **Add a New z/OS image** option.

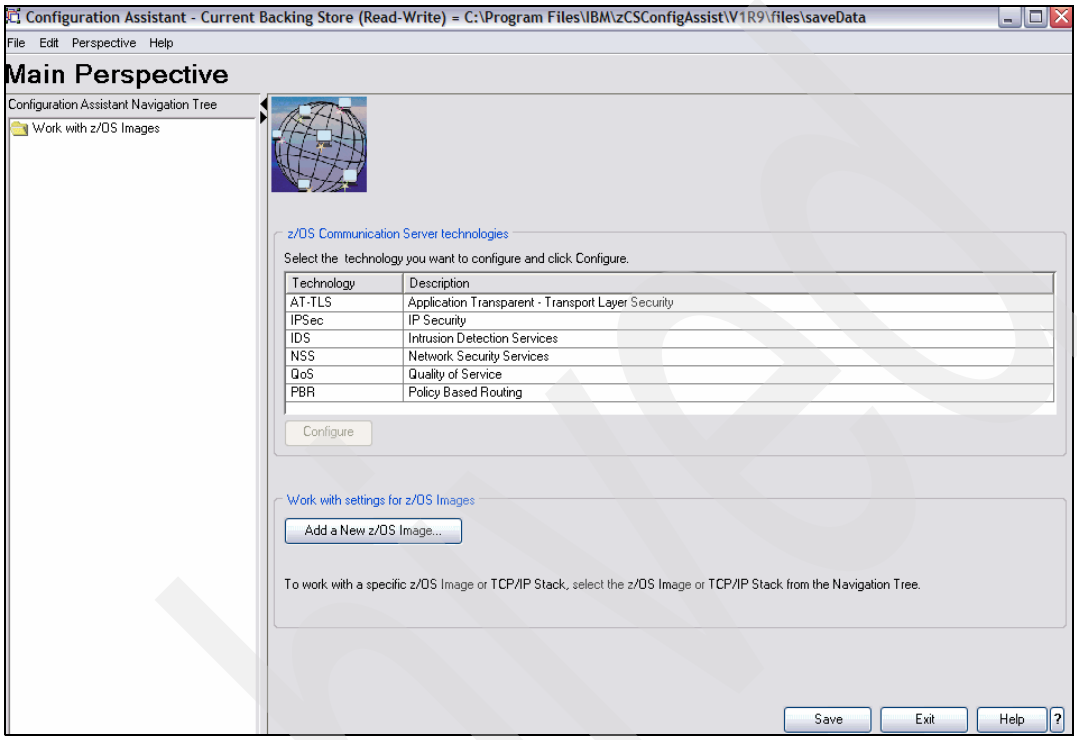


Figure 13-2 IBM Configuration Assistant for z/OS Communication Server Main Perspective panel

2. Enter the name of the z/OS image. (In our example, we entered `SC33`.) Click **OK**.

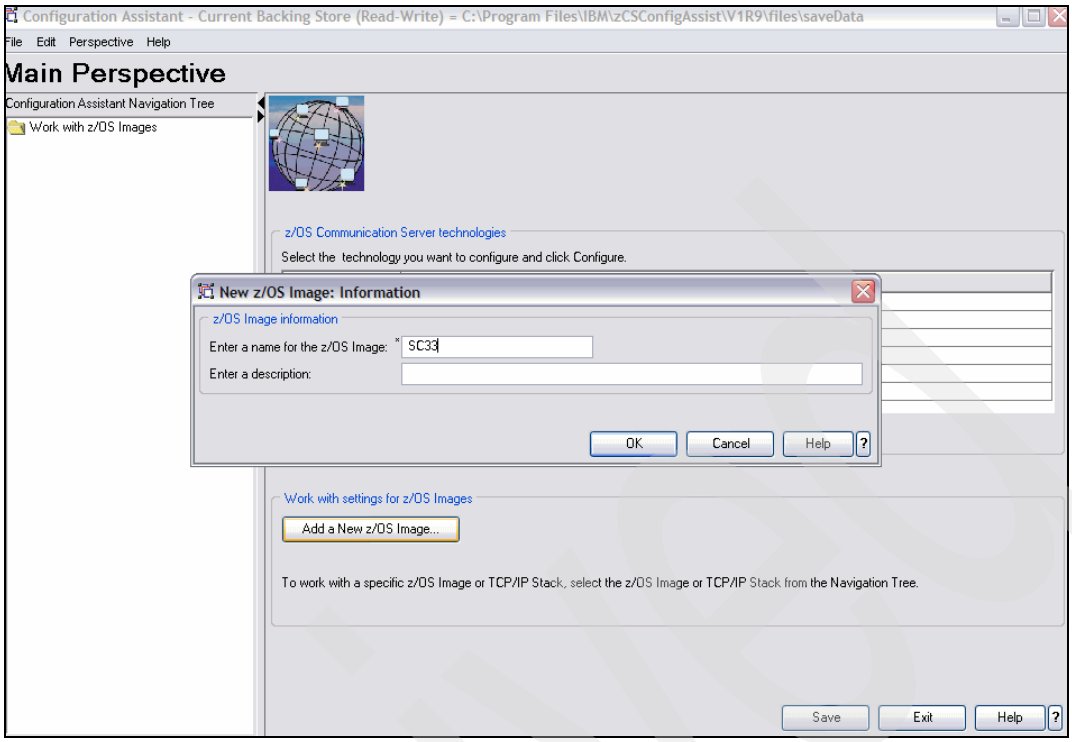


Figure 13-3 New z/OS Image panel: adding the SC33 z/OS image

3. In the Main Perspective panel, click the **Add New TCP/IP stack** option, and enter the name of the TCP/IP stack (in our case, `TCPIPD`).

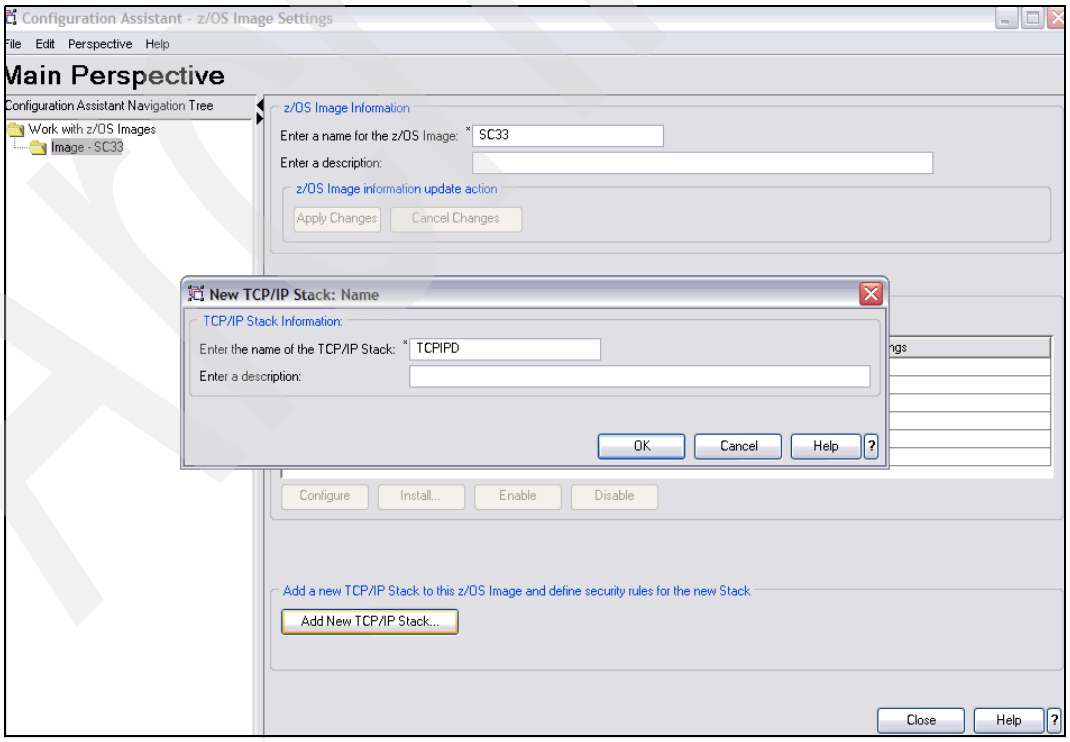


Figure 13-4 New TCP/IP Stack panel: adding the TCPIPD stack

4. In the Main Perspective panel, select PBR in the z/OS Communications Server technologies list. Click **Enable**, then click the **Configure** option.

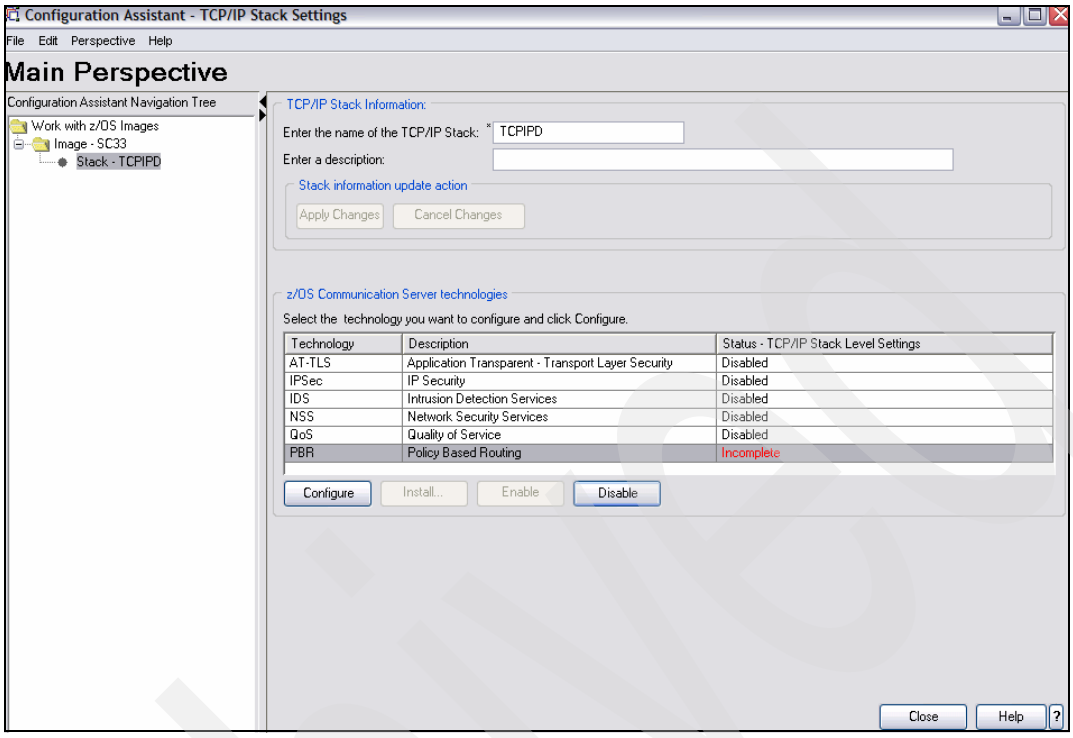


Figure 13-5 Main Perspective panel: adding the PBR policy

5. This leads you to the PBR Perspective panel. Notice the stack in the left pane shows `Incomplete Stack`, because the PBR policy is not yet configured. Click **Add**.

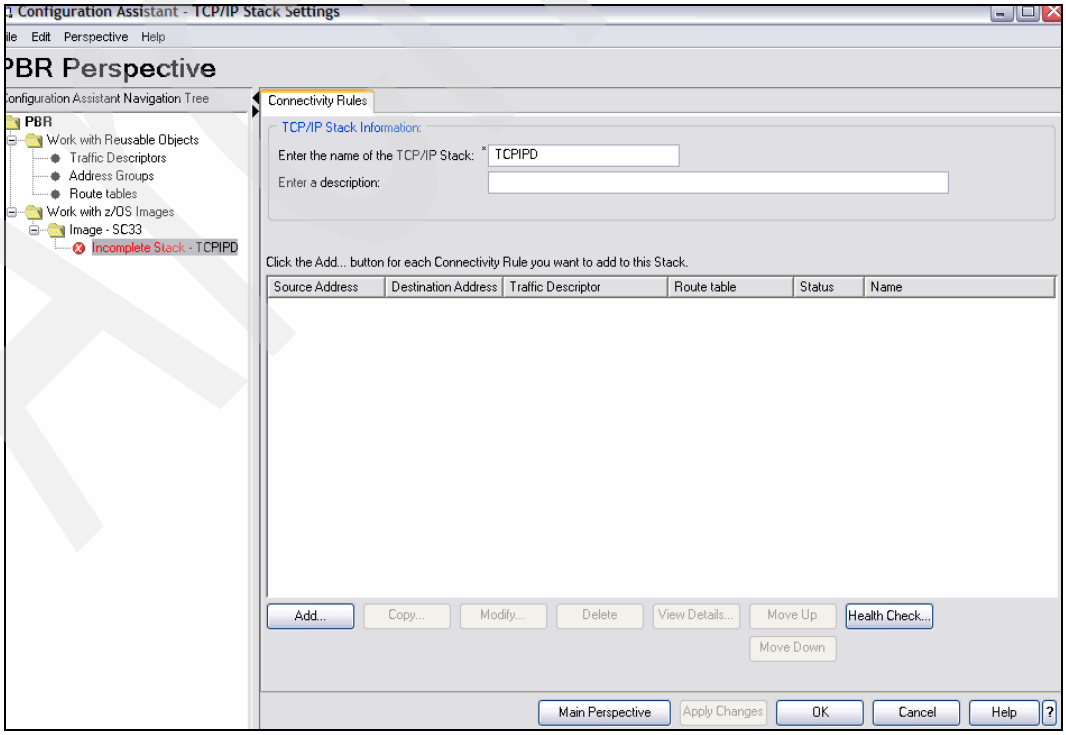


Figure 13-6 PBR Perspective panel: adding the PBR policy

6. Next, we defined the routing rule.

   Define the name for the routing rule. (In our example, we entered `PBRrule1` and a description.) Click **Next**.

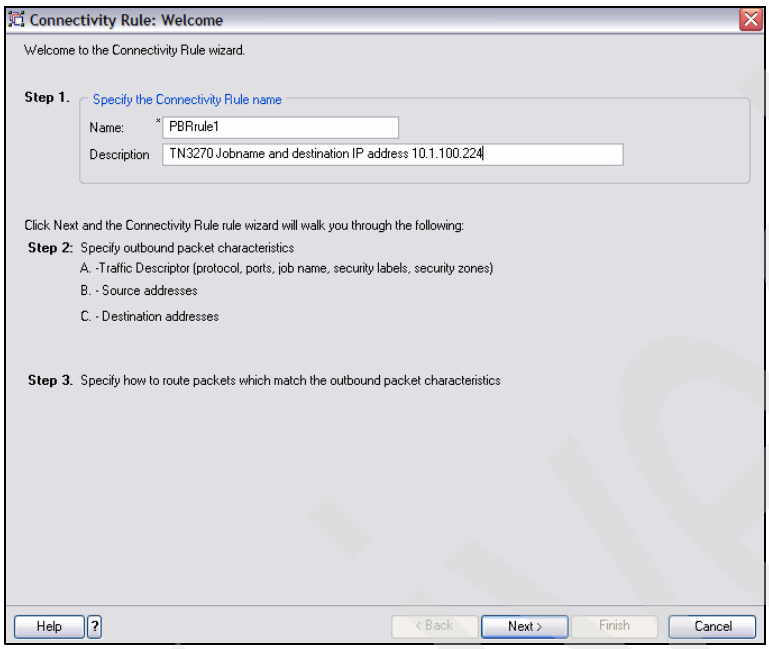


Figure 13-7 Connectivity Rule panel: Defining the routing rule

7. In the following three steps (2A through 2C), we configure the traffic descriptor. The first step (Rule Step 2A) is to specify the TCP or UDP protocol type and the port number.

   In our example, we selected TCP and clicked **Add**. If your policy does not require the TCP or UDP protocols or port numbers, select **Yes** (Specify the Traffic Descriptor parameters), then click **Next** to continue to step 9 on page 511 (Rule Step 2B).

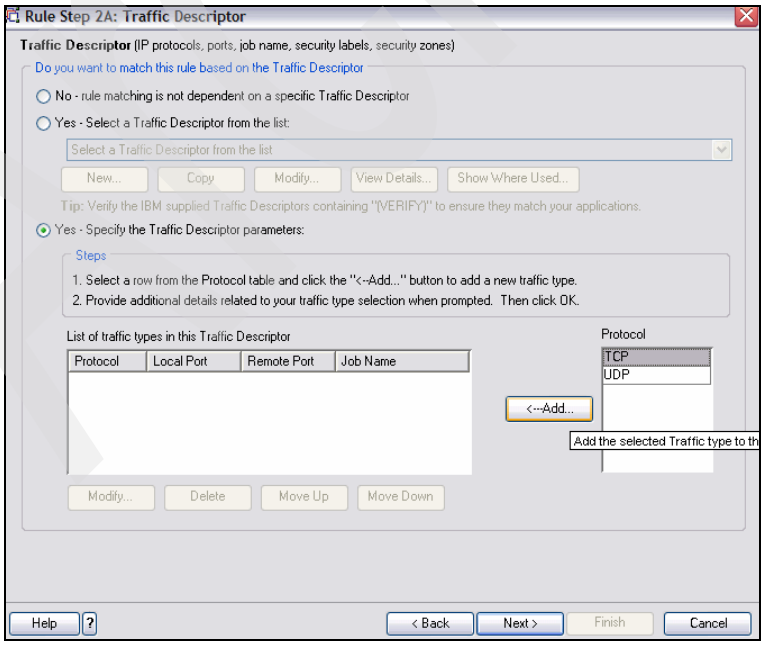


Figure 13-8 Connectivity Rule panel: defining the traffic description of ports and applications 1 of 2

8. Specify the source port, destination port, or jobname. (In our example, we selected All ports for both source port and destination port, and specified the jobname `TN3270*`.) See that TCP traffic with a jobname of TN3270* is added to the list of traffic types. Then click **OK**.

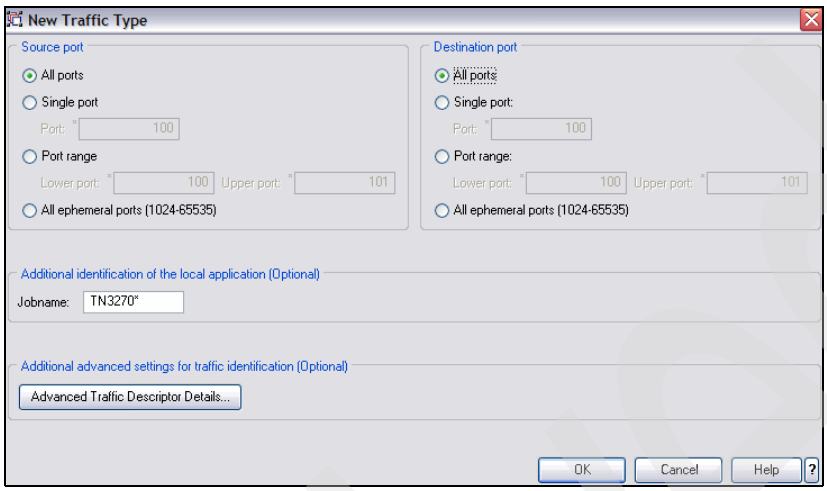


Figure 13-9 Connectivity Rule panel: defining the traffic description of ports and applications 2 of 2

9. In Rule Step 2B, specify the source IP address. (In our example, we clicked **No**, because we do not specify the source IP address as part of the traffic descriptor.) Click **Next**.

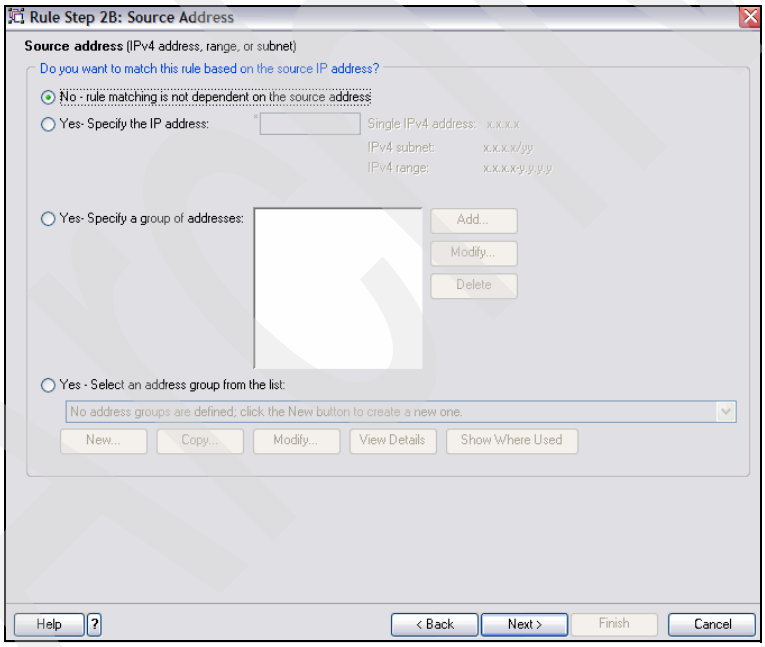


Figure 13-10 Connectivity Rule panel: defining the traffic description of source IP address

10. In Rule Step 2C, specify the destination IP address. (In our example, we specified the IP address of the terminal.) Click **Next**.

    You can also specify the time period when you want to activate this routing rule.

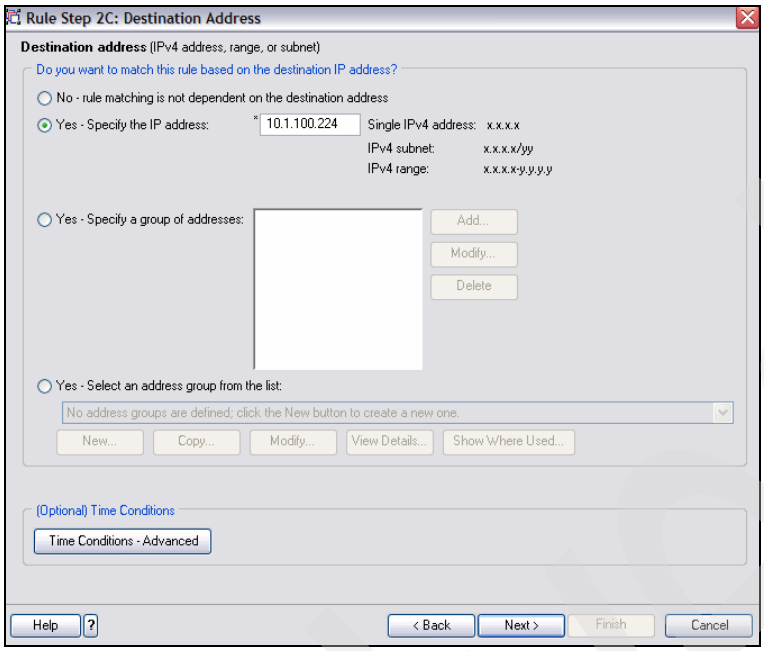


Figure 13-11 Connectivity Rule panel: defining the traffic description of destination IP address

11. In this step, you define routing table and routing actions. Select the **New** option.

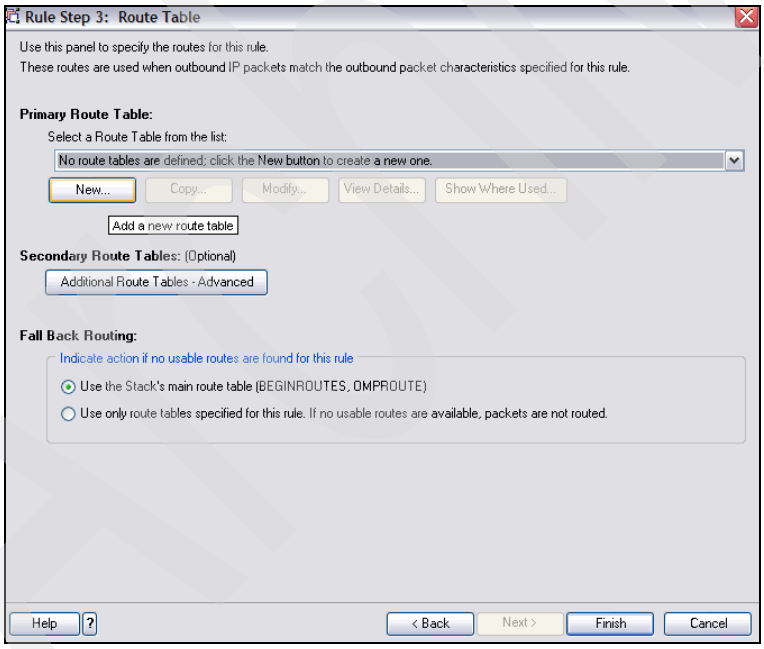


Figure 13-12 Route Table panel

12. In the New Table window, enter the name of the routing table. (In our example, we specified `PBRTAB1`.)
13. Now define routing entries. (In our example, we defined dynamic routing.) In the dynamic routing pane, click **Add**. Enter the name of the link that you want to send the traffic from. (We specified `OSA2OCOL`.) Click **OK**.

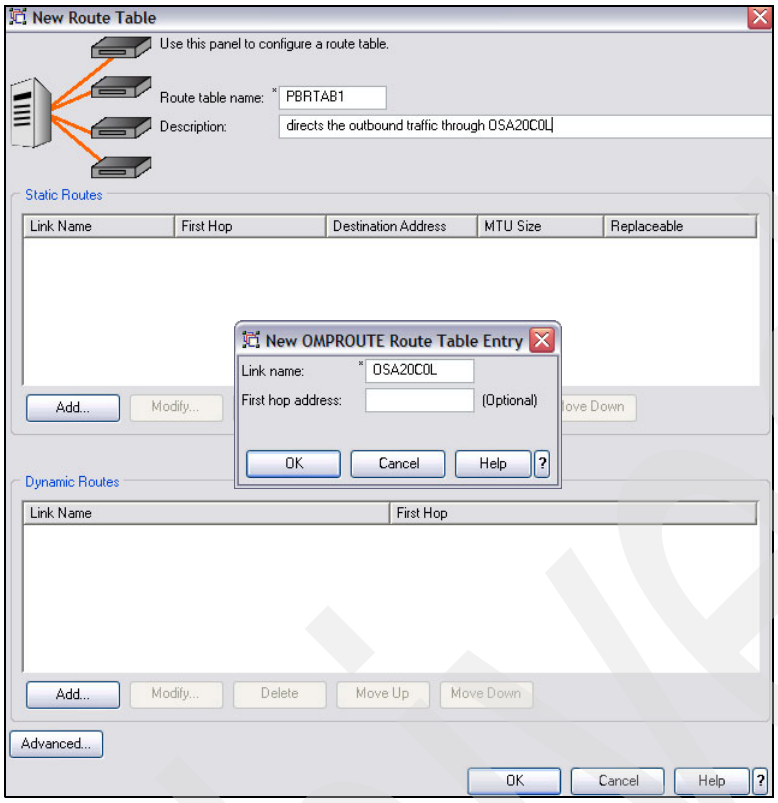


Figure 13-13 Route Table panel: defining the routing table and routing entries

14. Check that the link for dynamic routing is added to the Dynamic Routes list. Optionally, you may also define advanced options (MULTIPATH, Path MTU update, Dynamic XCF Routes) by selecting the **Advanced** option. Click **OK**.

15. Make sure the routing table you have just created is selected in the Primary Routing Table list. You can define additional routing tables by selecting the **Additional Route Tables - Advanced** option. We also selected to use the main routing table as a backup when no routing entries can be used for the traffic that matches the traffic descriptor criteria. Click **Finish**.

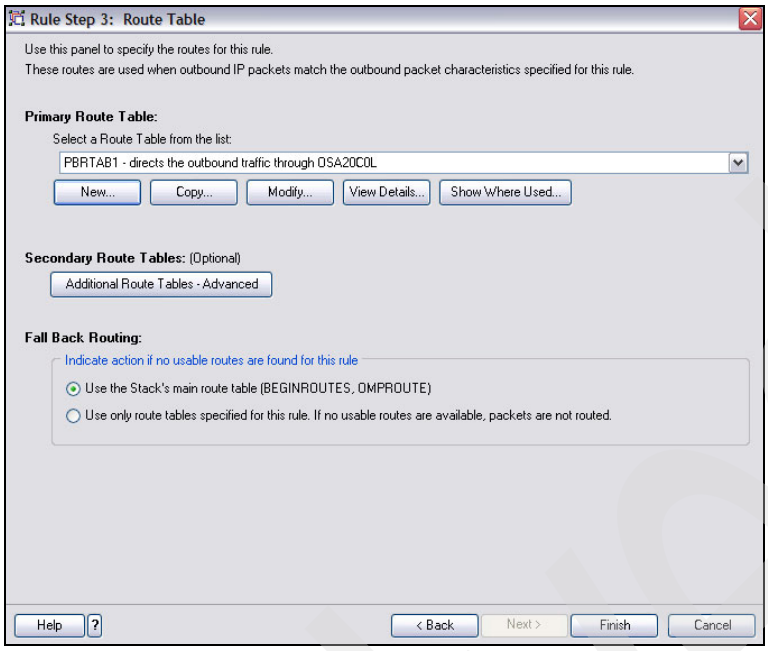


Figure 13-14 Route Table panel: defining routing actions

16. You are back to PBR Perspective. Click the **Apply Changes** option.

The definition is complete. Example 13-1 shows the flat file generated by the IBM Configuration Assistant for z/OS Communication Server.

Example 13-1 The policy configuration file generated by IBM Configuration Assistant

```
##
## PBR Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R9\files\saveData1
## FTP History:
## 2007-09-20 11:44:23  cs06 to 10.1.1.40
##
## TN3270 Jobname and destination IP address 10.1.100.224
RoutingRule PBRrule1 1
{
    IpDestAddr 10.1.100.224 2
    TrafficDescriptorGroupRef PBRrule1~
    Priority 500000
    RoutingActionRef PBRrule1 4
}

## Action for RoutingRule:PBRrule1 - TN3270 Jobname and destination IP address 10.1.100.224
RoutingAction PBRrule1 5
{
    UseMainRouteTable Yes
    RouteTableRef PBRTAB1 6
}

## PBRTAB1 - directs the outbound traffic through OSA20C0L
```

```
RouteTable PBRTAB1 7
{
    DynamicRoutingParms OSA20C0L 8
}

TrafficDescriptorGroup PBRrule1~
{
    TrafficDescriptor
    {
        Protocol TCP 9
        SourcePortRange 0
        DestinationPortRange 0
        Jobname TN3270* 3
    }
}
```

1 The routing rule PBRrule1 is defined.

2 The destination IP address 10.1.100.224 is specified in the routing rule PBRrule1.

3 The routing rule refers to the traffic with jobname TN3270*.

4 The rule refers to the routing action PBRrule1.

5 The routing action PBRrule1 is defined.

6 The routing action PBRrule1 refers to the routing table PBRTAB1.

7 The routing table PBRTAB1 is defined.

8 The routing table PBRTAB1 contains the dynamic routing entry with OSA20C0L specified as the sending interface name.

9 The routing rule refers to the TCP protocol.

Upload the generated PBR policy file to z/OS

1. In the Configuration Assistant Navigation Tree pane, right-click the TCP/IP stack name (TCPIPD, in our example), and select the **Install Configuration Files** option.

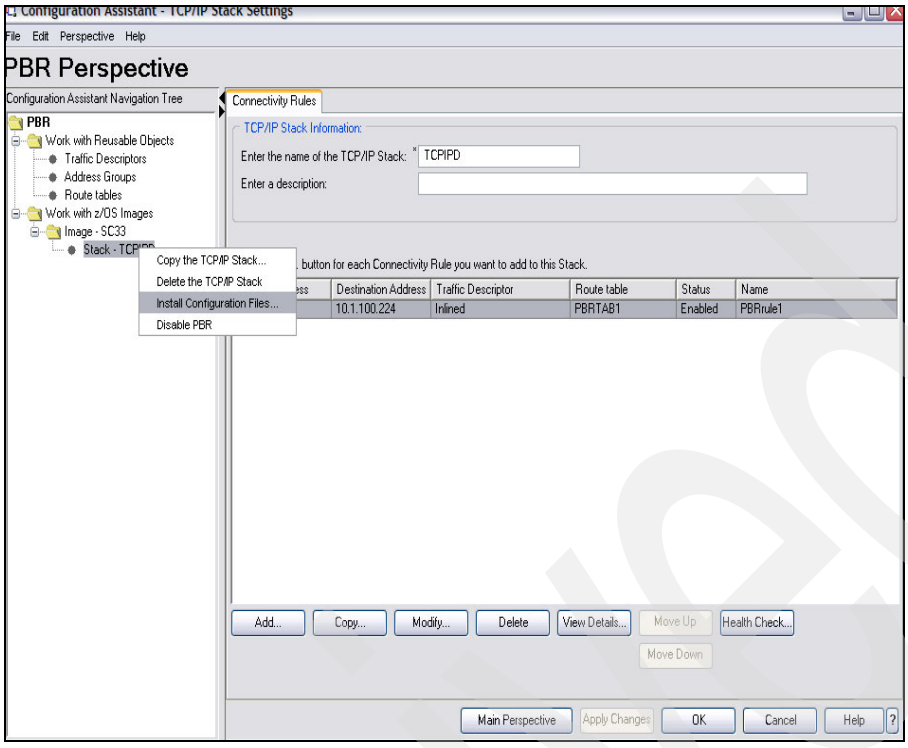


Figure 13-15 PBR Perspective panel with defined policy

2. Select the *stackname* - PBR: Policy Agent Stack Configuration in the list and click the **FTP** option. If you do not want to use FTP to upload the configuration file directly to z/OS, select the **Show Configuration File** option instead, then select **Save As** option to save it locally for later transfer.

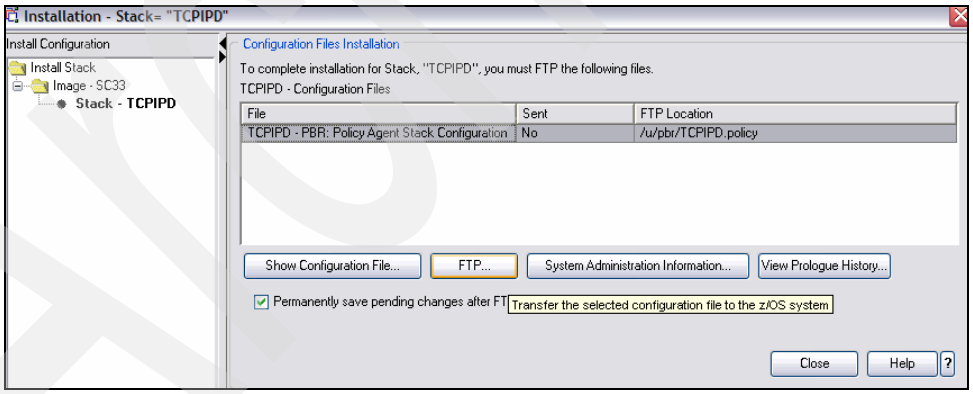


Figure 13-16 Installation panel: installing Configuration File

3. Enter the FTP server IP address, port number, FTP user ID, and password. Also specify the filename to be saved as. Click **Send**.

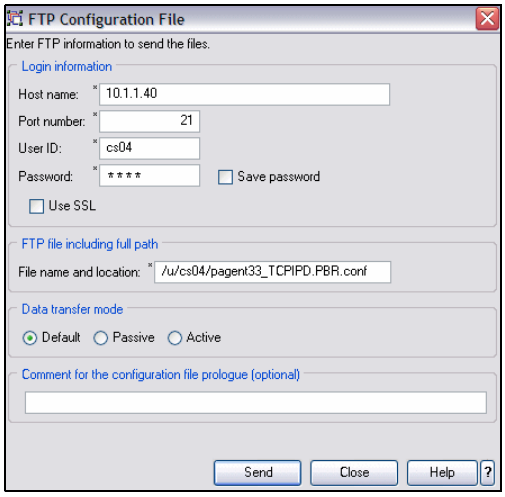


Figure 13-17 FTP Configuration File panel: transfer the configuration file to z/OS

4. When the FTP transfer is completed, the list in the Installation panel indicates the transferred date and time in the Sent column.

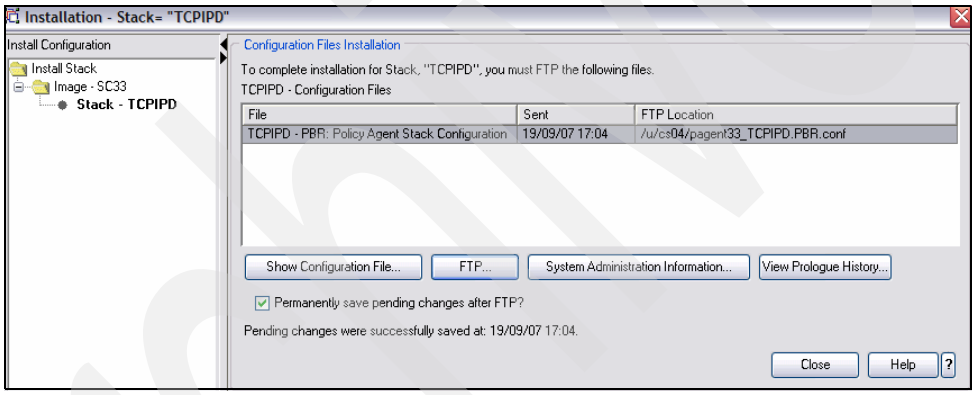


Figure 13-18 Completion of transfer through FTP

5. Copy the uploaded Policy Agent configuration file to the directory where you want to store it. In our example, we copied the file to /etc directory as shown in Example 13-2.

Example 13-2 Copy the uploaded configuration file to /etc directory

```
CS04 @ SC33:/u/cs04>cd /u/cs04
CS04 @ SC33:/u/cs04>ls
pagent33_TCPIPD.PBR.conf
CS04 @ SC33:/u/cs04>cp pagent33_PBR.conf /etc/
CS04 @ SC33:/u/cs04>
```

Update the Policy Agent configuration file

Example 13-3 shows the main configuration file of the Policy Agent. It defines the stack-specific configuration file name for TCPIPD stack.

Example 13-3 Main Policy Agent configuration file

```
#  ********************************************************************
#  /etc/pagent33.conf in SC33
#  ********************************************************************
```

```
  TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH NOPURGE 600
```

Example 13-4 shows the stack-specific configuration file for TCPIPD stack. We added the RoutingConfig statement that points to the policy configuration file.

Example 13-4 Policy Agent configuration file for TCPIPD

```
#  ********************************************************************
#  /etc/pagent33_TCPIPD.conf in SC33
# ********************************************************************
RoutingConfig /etc/pagent33_PBR.conf
```

Note: To enable the same policy configuration file to all TCP/IP stacks on the z/OS image, you may specify CommonRoutingConfig in the main policy agent configuration file, and specify RoutingConfig (with no file path) in the stack-specific configuration file.

Start the Policy Agent

Start the Policy Agent to enable the policy-based routing. The message 1 shows the policy-based routing policy is processed and now in effect.

Example 13-5 Starting the Policy Agent

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING 1
```

If you have the Policy Agent started already, use F *jobname*,REFRESH command as shown in Example 13-6. The message 2 informs the policy-based routing policy is loaded (or reloaded).

Example 13-6 Refreshing the Policy Agent

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING 2
```

Verification

Next, we show how to verify whether the policy was applied to the TCP/IP stack and that the policy-based routing performed as it was defined.

List the defined policies

The **pasearch -R** command lists the policy-based routing rules.

Example 13-7 pasearch -R display

```
CS06 @ SC33:/u/cs06>pasearch -R

TCP/IP pasearch CS V1R9                  Image Name: TCPIPD
  Date:                 09/20/2007       Time:  11:56:26
  Routing Instance Id:  1190303591

policyRule:             PBRrule1 1
```

```
  Rule Type:            Routing
  Version:              4                 Status:            Active 2
  Weight:               500000            ForLoadDist:       False
  Priority:             500000            Sequence Actions:  Don't Care
  No. Policy Action:    1
  policyAction:         PBRrule1 3
   ActionType:          Routing
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
   Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
   Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC:  04:00
   TimeZone:            Local
  Routing Condition Summary:              NegativeIndicator: Off
   IpSourceAddr Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   IpDestAddr Address:
    FromAddr:           10.1.100.224 4
    ToAddr:             10.1.100.224
   TrafficDescriptor:
    Protocol:           TCP  (6) 5
    SourcePortFrom      0                 SourcePortTo       0
    DestinationPortFrom 0                 DestinationPortTo  0
    JobName             TN3270* 6         SecurityZone
    SecurityLabel
  Routing Action:       PBRrule1
    Version:            4                 Status:            Active
    UseMainRouteTable   Yes
    RouteTable:         PBRTAB1 7
```

1 The policy rule in use is PBRrule1.

2 The rule is in active status.

3 PBRrule1 implements a routing action.

4 The traffic descriptor contains the destination IP address 10.1.100.224.

5 The routing rule refers to the TCP protocol

6 The traffic descriptor contains the jobname TN3270*.

7 The routing action for PBRrule1 is to use route table PBRTAB1, and it is active.

The **pasearch -T** command lists the policy-based routing tables.

Example 13-8 pasearch -T display

```
CS06 @ SC33:/u/cs06>pasearch -T

TCP/IP pasearch CS V1R9                 Image Name: TCPIPD
  Date:                 09/20/2007      Time:  12:09:45
```

```
  Routing Instance Id:  1190303591

  Route Table:          PBRTAB1 1
    Version:            1                 Status:           Active
    IgnorePathMtuUpdate No                Multipath         UseGlobal
    DynamicXCFRoutes    No
    DynamicRoutingParms
     link_name             OSA20C0L 2
```

1 The routing table PBRTAB1 is defined.

2 The routing table has a dynamic routing entry with link name OSA20C0L.

Verify the Routing Table

Example 13-9 and Example 13-10 show the main routing table managed by the TCP/IP stack and OMPROUTE. Notice that OSA2080L is to be used for sending the packets destined to 10.1.100.0/24. This route table will be used for all traffic that does not match the criteria that we defined.

Example 13-9 Main routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE
EZD0101I NETSTAT CS V1R9 TCPIPD 937
IPV4 DESTINATIONS
DESTINATION         GATEWAY         FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.2.220      UGO       000000  OSA2080L
DEFAULT             10.1.2.240      UGO       000000  OSA2080L
DEFAULT             10.1.3.220      UGO       000000  OSA20C0L
DEFAULT             10.1.3.240      UGO       000000  OSA20C0L
10.1.1.40/32        0.0.0.0         UH        000000  VIPA1L
10.1.2.0/24         0.0.0.0         UO        000000  OSA2080L
10.1.2.41/32        0.0.0.0         UH        000000  OSA2080L
10.1.3.0/24         0.0.0.0         UO        000000  OSA20C0L
10.1.3.41/32        0.0.0.0         UH        000000  OSA20C0L
10.1.100.0/24       10.1.2.240      UGO       000000  OSA2080L 1
```

Example 13-10 Main routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE
EZZ7847I ROUTING TABLE 951
TYPE   DEST NET        MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                0  101     21      10.1.2.220       (8)
 DIR*  10.1.1.0         FFFFFF00  1       63214   10.1.1.40
 DIR*  10.1.1.40        FFFFFFFF  1       63214   VIPA1L
 SPF*  10.1.2.0         FFFFFF00  100     63208   OSA2080L         (2) 1
 SPF*  10.1.3.0         FFFFFF00  100     63208   OSA20C0L         (2)
 SPF   10.1.100.0       FFFFFF00  101     63208   10.1.2.240       (4) 1
```

1 OSA2080L is to be used for sending the packet destined to 10.1.100.0/24.

Example 13-11 on page 521 and Example 13-12 on page 521 show the policy-based routing table managed by the TCP/IP stack and OMPROUTE.

Notice that the sending interface for all routing entries is OSA20C0L. This route table will be used for all traffic generated by TN3270* and destined for 10.1.100.224.

Example 13-11 Policy-based routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R9 TCPIPD 932
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
  DYNAMICXCFROUTES:    NO
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.3.220       UGO       000000  OSA20C0L
DEFAULT             10.1.3.240       UGO       000000  OSA20C0L
10.1.2.0/24         10.1.3.240       UGO       000000  OSA20C0L
10.1.3.0/24         0.0.0.0          UO        000000  OSA20C0L
10.1.100.0/24       10.1.3.240       UGO       000000  OSA20C0L 2
```

Example 13-12 Policy-based routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE,PRTABLE=PBRTAB1
EZZ7847I ROUTING TABLE 941
TABLE NAME: PBRTAB1
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0               0  101     209     10.1.3.220       (2)
 SPF   10.1.2.0         FFFFFF00  101     209     10.1.3.240
 SPF*  10.1.3.0         FFFFFF00  100     209     OSA20C0L 2
 SPF   10.1.100.0       FFFFFF00  101     209     10.1.3.240 2

DEFAULT GATEWAY IN USE.

TYPE COST     AGE       NEXT HOP
SPIA 101      209       10.1.3.220       (2)
                       0 NETS DELETED
DYNAMIC ROUTING PARAMETERS
  INTERFACE:  OSA20C0L 2 NEXT HOP: ANY
```

2 OSA20C0L is to be used for sending the packets destined for 10.1.100.0/24.

Now we start a new TN3270 session from terminal with IP address 10.1.100.224. The traffic uses our defined policy and therefore uses the policy-based routing table.

As shown in Example 13-13, we used the Netstat COnn/-c command and the Netstat ALL/-a command to see if the established TCP connection matches the routing rule PBRrule1, and if it uses the routing table PBRTAB1. According to the PBRTAB1 routing table, OSA20C0L is used for sending traffic on this connection.

Example 13-13 Netstat All Display

```
CS06 @ SC33:/u/cs06>netstat -p TCPIPD -c
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD          12:25:40
User Id  Conn     State
-------  ----     -----
TN3270D  000016B5 Establsh
  Local Socket:   ::ffff:10.1.1.40..23
  Foreign Socket: ::ffff:10.1.100.224..4872

CS06 @ SC33:/u/cs06>netstat -p TCPIPD -A -P 4872
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD          12:26:31
Client Name: TN3270D                  Client Id: 000016B5
  Local Socket: ::ffff:10.1.1.40..23
```

```
  Foreign Socket: ::ffff:10.1.100.224..4872
.....
RoutingPolicy:      Yes
      RoutingTableName: PBRTAB1 1
      RoutingRuleName:  PBRrule1 2
```

1 The routing table PBRTAB1 is in use for this connection.

2 The routing rule PBRrule1 is in use for this connection.

If the TN3270 session from terminal with IP address 10.10.100.222 is initiated, RoutingPolicy would appear, but with a value of No. However, RoutingTableName and RoutingRuleName would not appear.

To verify that the traffic is actually going through the defined OSA, you can use Netstat Devlinks command as shown in Example 13-14 to see the BytesOut and Outbound Packets are incremented on the specific OSA.

Example 13-14 Netstat Devlinks display

```
CS06 @ SC33:/u/cs06>netstat -p TCPIPD -d -K OSA20C0L
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD         12:29:25
DevName: OSA20C0           DevType: MPCIPA
  DevStatus: Ready
  LnkName: OSA20C0L          LnkType: IPAQENET   LnkStatus: Ready
.....
Link Statistics:
    BytesIn                          = 176
    Inbound Packets                  = 2
    Inbound Packets In Error         = 0
    Inbound Packets Discarded        = 0
    Inbound Packets With No Protocol = 0
    BytesOut                         = 55228
    Outbound Packets                 = 296
    Outbound Packets In Error        = 0
    Outbound Packets Discarded       = 0
```

13.3.2 Policy-based routing using protocol and port numbers

In this scenario we implement policy-based routing to force all FTP traffic (ports 20 and 21) to use the OSA connection between TCPIPC and TCPIPD, while all other traffic will use the HiperSockets connection. Figure 13-19 on page 523 illustrates the policy-based routing scenario configuration.

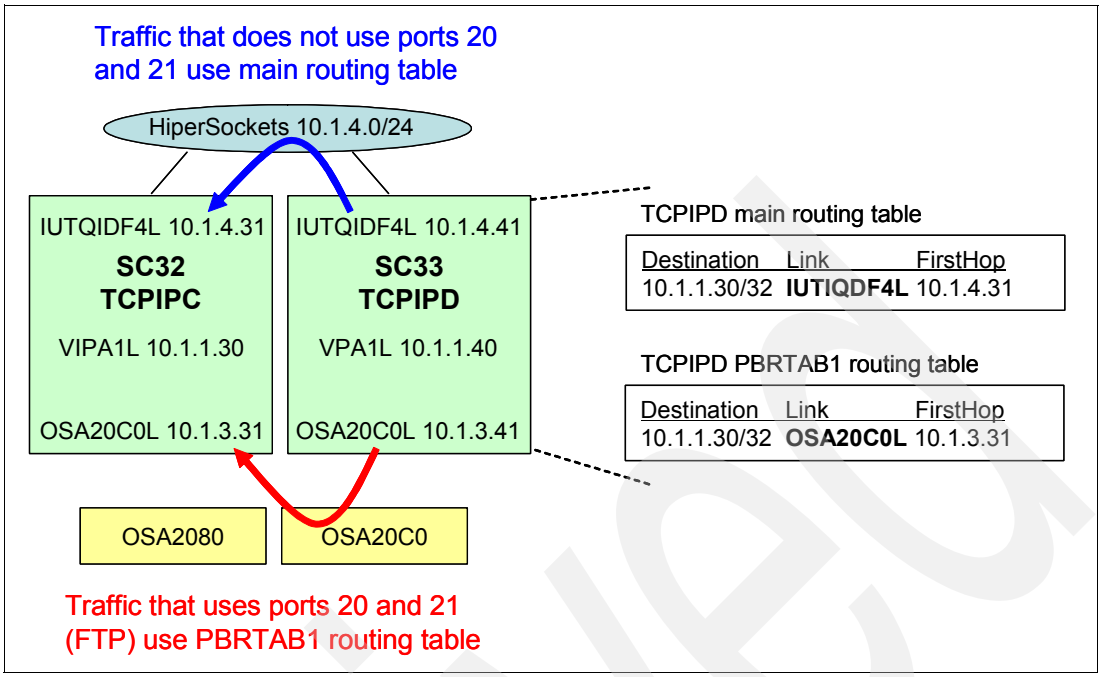


Figure 13-19 Policy-based routing using protocol and port numbers

Implementation tasks

We implemented policy-based routing using the following steps:

1. Configure the PBR policy with protocol and port numbers.
2. Upload the generated PBR policy file to z/OS.
3. Update the Policy Agent configuration file.
4. Start the Policy Agent.

No changes are required for the TCP/IP profile or the OMPROUTE profile.

Configure the PBR policy with protocol and port numbers

To configure the PBR policy for this scenario we used the same configuration file created for scenario 1 described in 13.3.1, “Policy-based routing using jobname, protocol, and destination IP address” on page 506, and we added the additional rule. If you are starting the PBR configuration from the beginning, first follow steps 1 through 4 described in 13.3.1, “Policy-based routing using jobname, protocol, and destination IP address” on page 506, and then continue with the following steps.

To add this rule we used the IBM Configuration Assistant for z/OS Communication Server, and followed these steps:

1. Start IBM Configuration Assistant for z/OS Communication Server. In Configuration Assistant Navigation Tree, click **Image-SC33** and then select stack **TCPIPD**.

   In the field z/OS Communications Server Technologies, the PBR technology is shown as `Enabled`, meaning we already have policies saved for PBR. Select **PBR** and click the **Configure** button, as shown in Figure 13-20 on page 524.

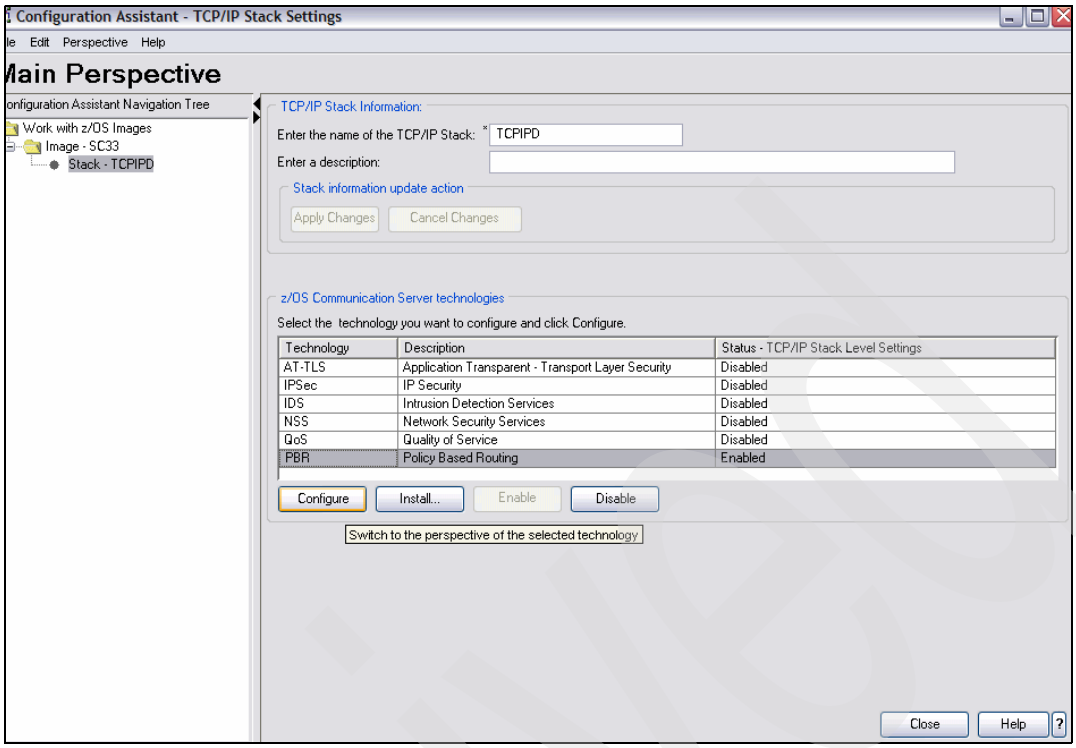


Figure 13-20 Configuration Assistance main perspective

2. The Configuration Assistant opens the PBR perspective, connected to stack TCPIPD as shown in Figure 13-21. In the Conectivity Rules field, we clicked **Add** to add our new PBR rule.

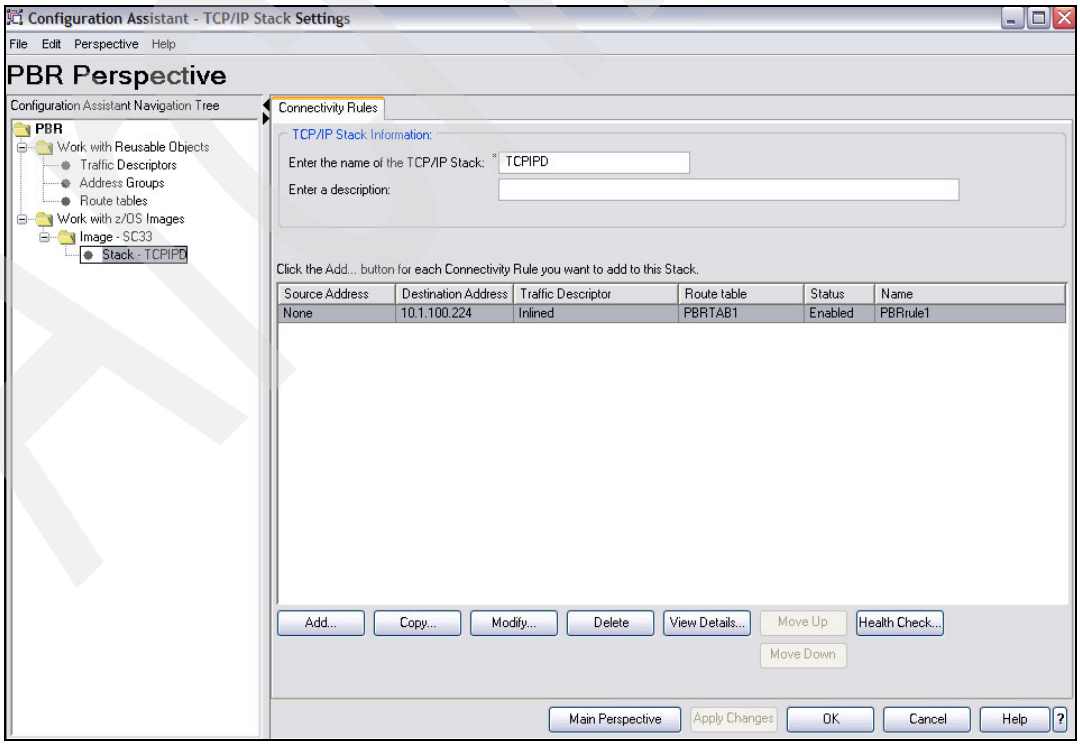


Figure 13-21 z/OS Configuration Assistant - PBR perspective

3. The Connectivity rule: Welcome panel is shown in Figure 13-22. We defined a name for our new PBR rule (PBRrule2), and described the rule in the Description field. Then we clicked **Next** to proceed to the next step.

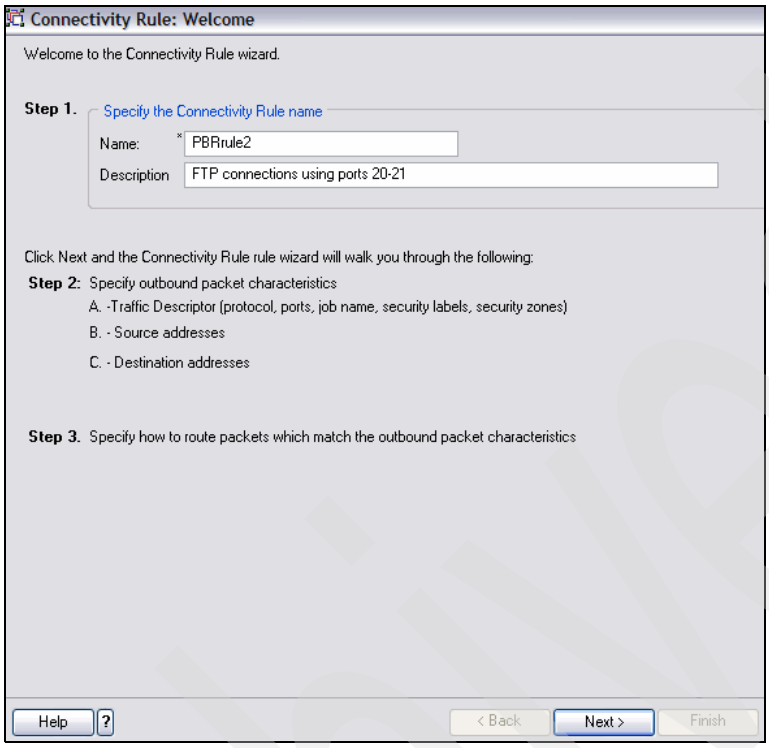


Figure 13-22 Connectivity rule: Welcome

4. In Rule Step2A: Traffic Descriptor, we added the policy for our PBR rule. To generate the traffic descriptor we selected **Yes - Specify the Traffic Descriptor Parameters**, then selected the **TCP** protocol in the Protocol field, and clicked **Add**, as shown in Figure 13-23 on page 526.

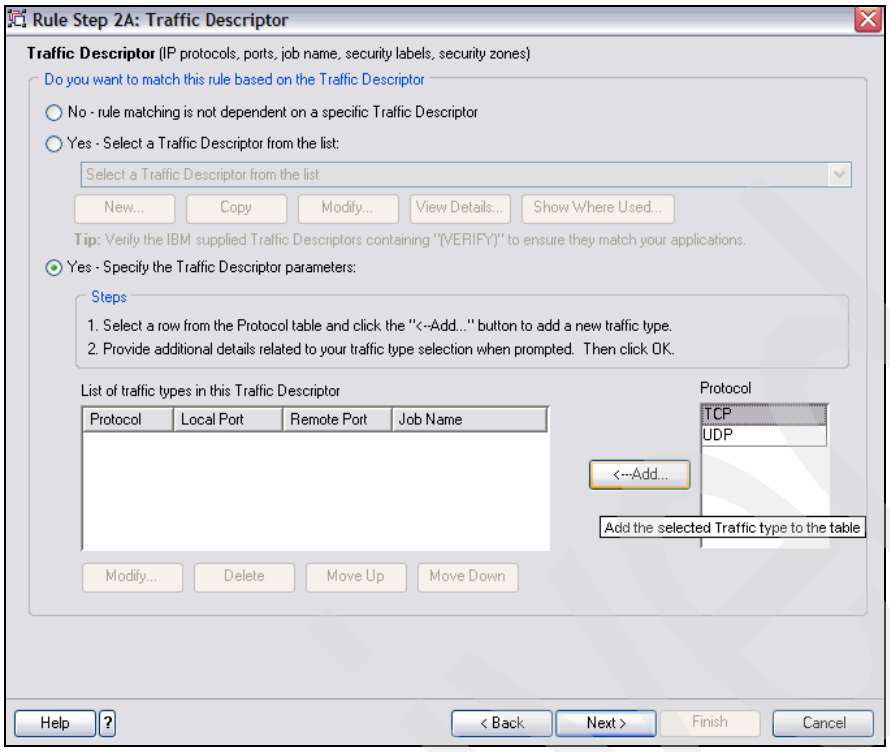


Figure 13-23 Rule Step2A: Traffic Descriptor

5. In the New Traffic Type panel, we defined the policy. In our scenario we wanted to use this PBR rule only for outbound traffic with source ports 20 and 21 (FTP data and control ports).

   To create this policy we went to the Source port field, clicked **Port range** and entered the port numbers (in the Lower Port field we entered 20; in the Upper Port field, we entered 21, as shown in Figure 13-24. Then we clicked **OK** to save our input data and save our policy.

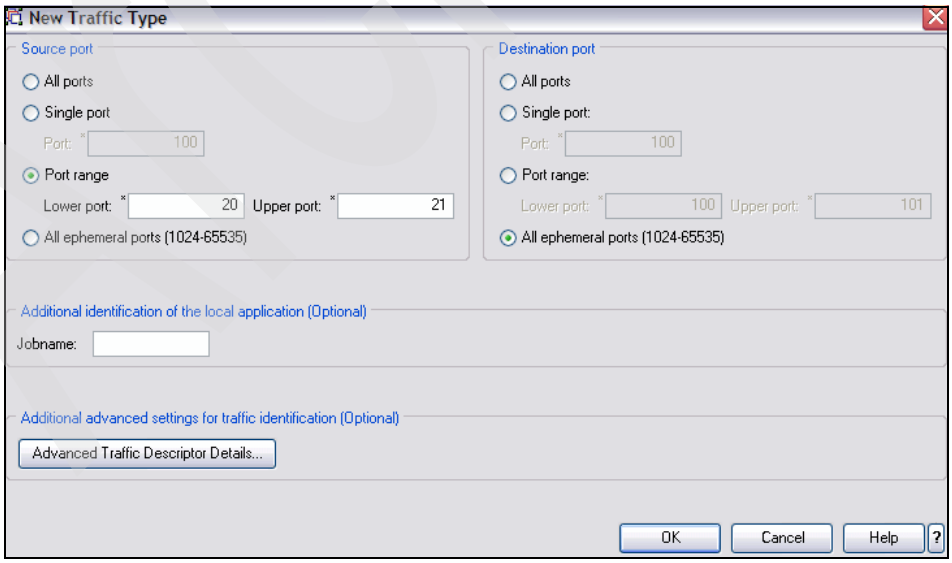


Figure 13-24 New Traffic Type

6. Back on the Rule Step2A: Traffic Descriptor panel, we confirmed our policy had been added in field List of traffic types in this Traffic Descriptor. We continued our implementation by clicking **Next**, as shown in Figure 13-25.

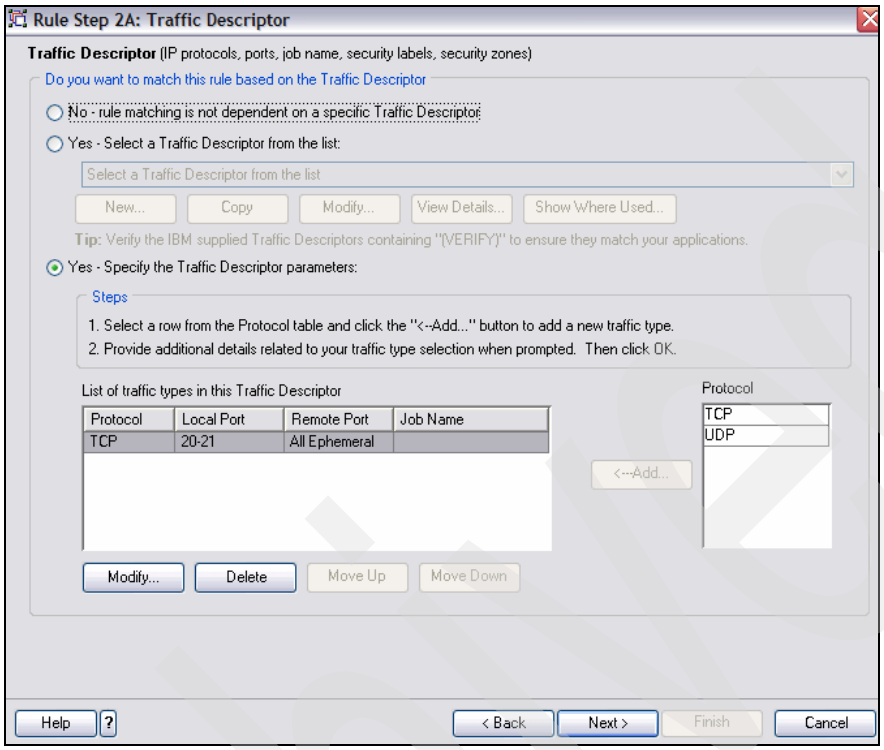


Figure 13-25 Rule Step 2A: Traffic Descriptor

7. The next panels, Rule Step 2B and Rule Step 2C, are used to define the source and the destination address for this PBR rule. In our scenario we are defining a rule that does not use the origin or destination address (we selected **NO - rule matching is not dependent on the destination address**), and clicked **Next** for both panels, as shown in Figure 13-26 on page 528.

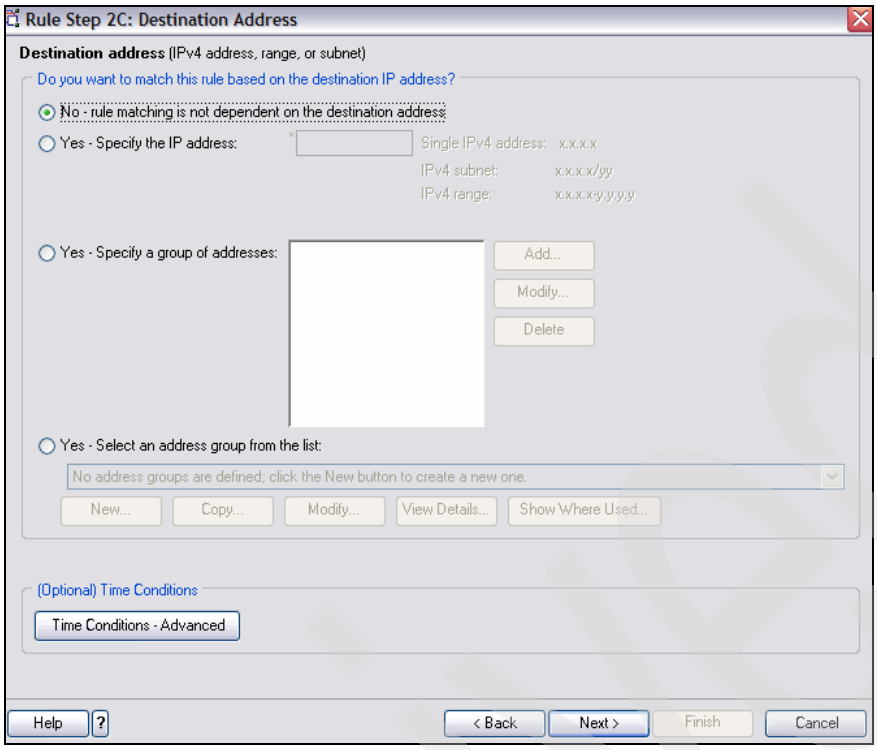


Figure 13-26 Rule Step 2C: Destination Address

8. Next, we provide a route table to our PBR rule, which can be a new or existing table. In our scenario we used the same route table created in scenario 1 (PBRTAB1), which routes all outbound traffic through link OSA20C0L. To see how the table is created refer to “Policy-based routing using jobname, protocol, and destination IP address” on page 506, starting with step 6 on page 510.

   To choose the table, we used the drop-down list in the field Primary Route Table and selected the desired route table, **PBRTAB1**, as shown in Figure 13-27 on page 529. We also selected **Use the Stack's main route table** for fallback routing. We clicked **Finish** to end the configuration and return to the PBR Perspective.

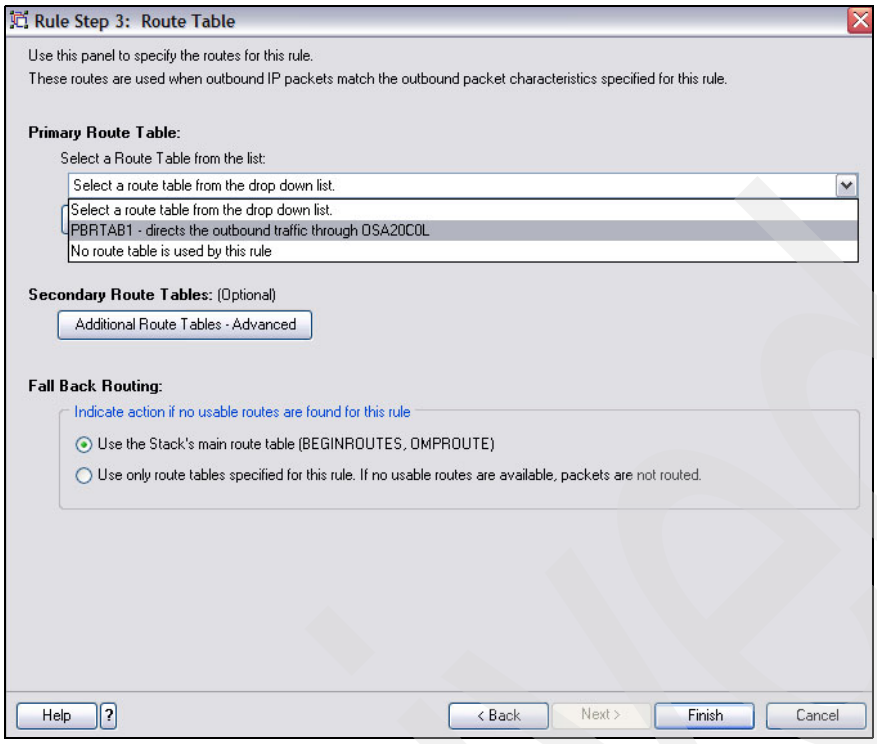


Figure 13-27 Rule Step 3: Route Table

9. Back at the PBR Perspective, our connectivity rule is created and we clicked **Apply Changes** to save it in the configuration file.

Upload the generated PBR policy file to z/OS

1. Now we install the PBR policy configuration file, which entails sending the file to z/OS to be included in the Policy Agent configuration. To install the file, right-click the desired TCP/IP stack, **TCPIPD,** and select **Install Configuration Files**, as shown in Figure 13-28 on page 530.

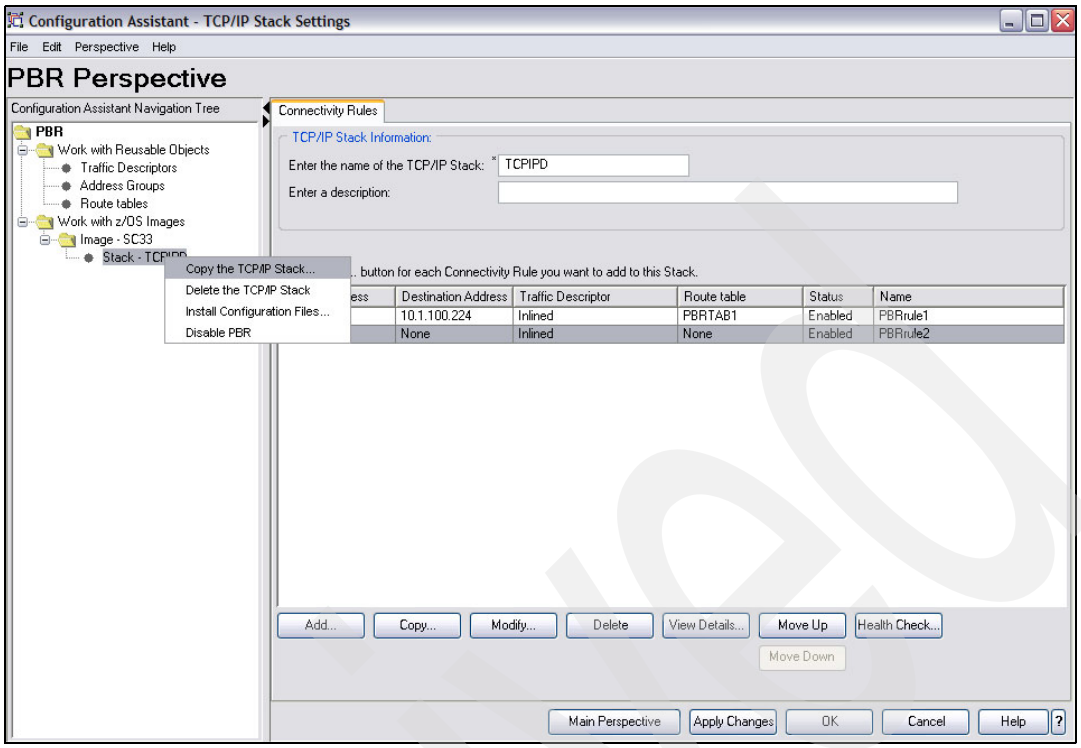


Figure 13-28 PBR perspective

2. On the right side of the panel, we selected the file we wanted to install in the **Configuration Files Installation** field and clicked **FTP...**, as shown in Figure 13-29.

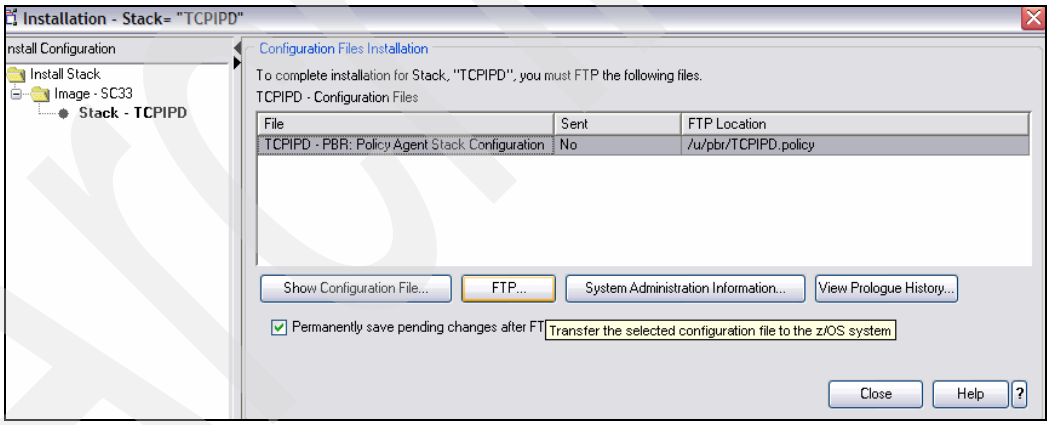


Figure 13-29 Configuration files Installation

3. Choosing the **FTP** option on the Configuration files Installation panel opens an FTP Configuration File. Here we define the FTP information needed to send the files to z/OS, as shown in Figure 13-30 on page 531.

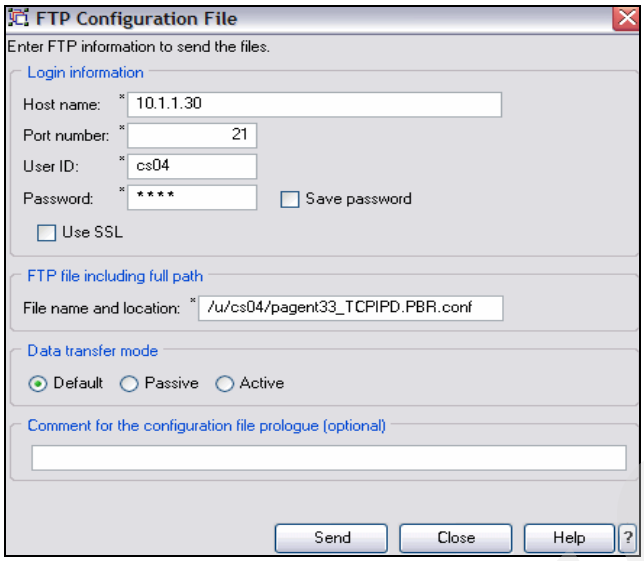


Figure 13-30 FTP Configuration File

4. After we sent the files to z/OS, we confirmed that the files were sent by viewing the Configuration Files Installation field, as shown in Figure 13-31.

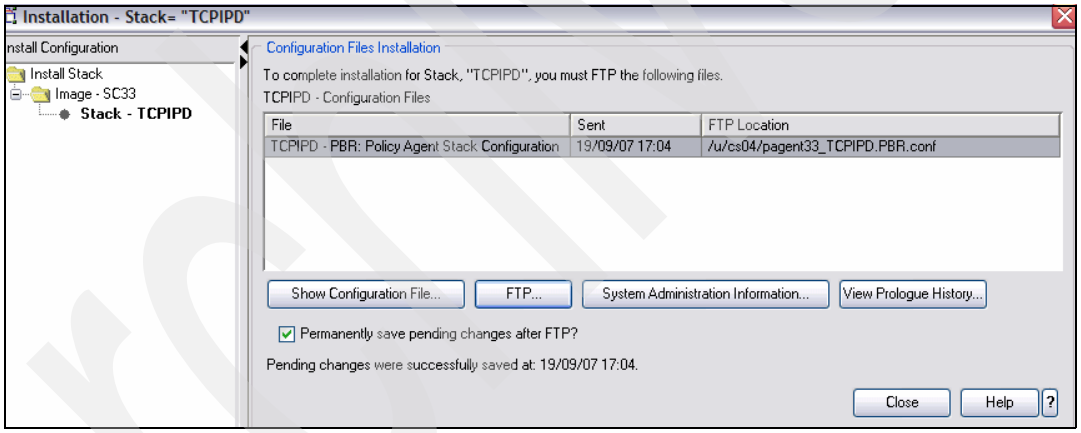


Figure 13-31 Installation Files

5. Now copy the uploaded Policy Agent configuration file to directory where you want to store it. (In our example, we copied the file to /etc directory.)

Example 13-15 Copying the uploaded configuration file to /etc directory

```
CS04 @ SC33:/u/cs04>cd /u/cs04
CS04 @ SC33:/u/cs04>ls
pagent33_TCPIPD.PBR.conf
CS04 @ SC33:/u/cs04>cp pagent33_PBR.conf /etc/
CS04 @ SC33:/u/cs04>
```

Update the Policy Agent configuration file

Example 13-16 on page 532 shows the Policy Agent configuration file defines the stack-specific configuration file for TCPIPD.

Example 13-16 Global Policy Agent configuration file

```
#  ******************************************************************
#  /etc/pagent33.conf in SC33
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH NOPURGE 600
```

We added the RoutingConfig statement that points to the policy configuration file.

Example 13-17 Policy Agent configuration file for TCPIPD

```
#  ******************************************************************
#  /etc/pagent33_TCPIPD.conf in SC33
# ******************************************************************
RoutingConfig /etc/pagent33_PBR1.conf
```

Note: To enable the same policy configuration file to all TCP/IP stacks on the z/OS image, you may specify CommonRoutingConfig in the main policy agent configuration file, and specify RoutingConfig (with no file path) in the stack-specific configuration file.

Start the Policy Agent

Start the Policy Agent to have the policy-based routing in effect.

Example 13-18 Starting the Policy Agent

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
ZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING
```

If you have the Policy Agent started already, use the F *jobname*,REFRESH command.

Verification

We used the **pasearch -R** command to list the policy-based routing rules and to verify that the rule we created for our scenario 2 is applied, as shown in Example 13-19.

Example 13-19 pasearch -R

```
CS04 @ SC33:/u/cs04>pasearch -R

TCP/IP pasearch CS V1R9                  Image Name: TCPIPD
  Date:                 09/20/2007       Time:  17:08:50
  Routing Instance Id:  1190321501
policyRule:             PBRrule2                                        1
  Rule Type:            Routing
  Version:              4                Status:            Active      2
  Weight:               499990           ForLoadDist:       False
  Priority:             499990           Sequence Actions:  Don't Care
  No. Policy Action:    1
  policyAction:         PBRrule2                                        3
   ActionType:          Routing
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
```

```
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
   Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC:  04:00
   TimeZone:            Local
  Routing Condition Summary:              NegativeIndicator: Off
   IpSourceAddr Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   IpDestAddr Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   TrafficDescriptor:
    Protocol:           TCP  (6)
    SourcePortFrom      20                SourcePortTo       21              4
    DestinationPortFrom 1024              DestinationPortTo  65535
    JobName                               SecurityZone
    SecurityLabel
  Routing Action:       PBRrule2                                            5
    Version:            4                 Status:            Active
    UseMainRouteTable   Yes
    RouteTable:         PBRTAB1
```

The resulting display shows the status of our policy:

1 The policy rule in use is PBRrule2.

2 The rule is in active status.

3 PBRrule2 implements a routing action.

4 The routing condition is to use this rule if ports 20 or 21 are being used as the source port.

5 The routing action for PBRrule2 is to use route table PBRTAB1, and it is active.

The **pasearch -T** command lists the policy-based routing tables active in our environment.

Example 13-20 The pasearch -T command

```
CS04 @ SC33:/u/cs04>pasearch -T

TCP/IP pasearch CS V1R9                 Image Name: TCPIPD
  Date:                 09/20/2007      Time:  17:37:33
  Routing Instance Id:  1190323638

  Route Table:          PBRTAB1
    Version:            1               Status:            Active
    IgnorePathMtuUpdate No              Multipath          UseGlobal
    DynamicXCFRoutes    No
    DynamicRoutingParms
     link_name            OSA2OC0L
```

Example 13-21 on page 534 and Example 13-22 on page 534 show the main routing table managed by TCP/IP stack and OMPROUTE. Notice IUTIQDF4L is to be used for sending the traffic destined to 10.1.1.30. We had IPCONFIG NOMULTIPATH coded in our configuration,

which means that IUTIQDF4L is the first route in the multipath group and will be used all the time. Only the first link in the multipath group is used.

Example 13-21 Main routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE
EZD0101I NETSTAT CS V1R9 TCPIPD 628
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.2.220       UGO       000000  OSA2080L
DEFAULT             10.1.2.220       UGO       000000  OSA20A0L
DEFAULT             10.1.2.240       UGO       000000  OSA2080L
DEFAULT             10.1.2.240       UGO       000000  OSA20A0L
DEFAULT             10.1.3.220       UGO       000000  OSA20C0L
DEFAULT             10.1.3.220       UGO       000000  OSA20E0L
DEFAULT             10.1.3.240       UGO       000000  OSA20C0L
DEFAULT             10.1.3.240       UGO       000000  OSA20E0L
10.1.0.40/32        0.0.0.0          UH        000000  VIPA3L
10.1.1.30/32        10.1.4.31        UGHO      000000  IUTIQDF4L
10.1.1.30/32        10.1.5.31        UGHO      000000  IUTIQDF5L
10.1.1.30/32        10.1.6.31        UGHO      000000  IUTIQDF6L
10.1.1.30/32        10.1.7.31        UGHO      000000  IQDIOLNK0A01072
90 OF 90 RECORDS DISPLAYED
```

Example 13-22 Main routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE
EZZ7847I ROUTING TABLE 686
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                0  101     417     10.1.2.220       (8)
 DIR*  10.1.1.0          FFFFFF00  1       84836   10.1.1.40
 SPF   10.1.1.10         FFFFFFFF  100     31528   10.1.4.11        (4)
 SPF   10.1.1.20         FFFFFFFF  100     31466   10.1.4.21        (4)
 SPF   10.1.1.30         FFFFFFFF  100     31371   10.1.4.31        (4)
 DIR*  10.1.1.40         FFFFFFFF  1       84836   VIPA1L
 SPF*  10.1.2.0          FFFFFF00  100     84830   OSA2080L         (2)
 SPF*  10.1.3.0          FFFFFF00  100     84830   OSA20C0L         (2)
 SPF*  10.1.4.0          FFFFFF00  90      84830   IUTIQDF4L
 SPF*  10.1.5.0          FFFFFF00  90      31545   IUTIQDF5L
 SPF*  10.1.6.0          FFFFFF00  90      31545   IUTIQDF6L

 DEFAULT GATEWAY IN USE.

 TYPE COST     AGE       NEXT HOP
 SPIA 101      417       10.1.2.220       (8)
                        0 NETS DELETED, 1 NETS INACTIVE
```

Example 13-23 and Example 13-24 on page 535 show the policy-based routing table managed by TCP/IP stack and OMPROUTE. Notice OSA20C0L is to be used for sending traffic to any destination if the conditions established in our routing policy are met, as we defined.

Example 13-23 Policy-based routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R9 TCPIPD 710
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
```

```
  DYNAMICXCFROUTES:    NO
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.3.220       UGO       000000  OSA2OC0L
DEFAULT             10.1.3.240       UGO       000000  OSA2OC0L
10.1.1.10/32        10.1.3.12        UGHO      000000  OSA2OC0L
10.1.1.20/32        10.1.3.240       UGHO      000000  OSA2OC0L
10.1.1.30/32        10.1.3.32        UGHO      000000  OSA2OC0L
```

Example 13-24 Policy-based routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE,PRTABLE=PBRTAB1
EZZ7847I ROUTING TABLE 316
TABLE NAME: PBRTAB1
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                0  101     309     10.1.3.220       (2)
 SPF   10.1.1.10        FFFFFFFF  110     3756    10.1.3.12        (2)
 SPF   10.1.1.20        FFFFFFFF  111     314     10.1.3.240
 SPF   10.1.1.30        FFFFFFFF  110     3905    10.1.3.32        (2)
 SPF   10.1.2.0         FFFFFF00  101     12619   10.1.3.240
 SPF   10.1.2.10        FFFFFFFF  200     3756    10.1.3.12        (2)
 SPF   10.1.2.20        FFFFFFFF  111     314     10.1.3.240
 SPF   10.1.2.30        FFFFFFFF  200     3905    10.1.3.32        (2)
 SPF*  10.1.3.0         FFFFFF00  100     12619   OSA2OC0L
```

Next, we tested whether our routing policy was working as expected. We started an FTP session from a TSO user on LPAR SC32, using the CS04 user. This session uses ports 20 and 21, which should match the routing policy. Outgoing traffic should use the policy-based routing table.

To verify this, we first used the command Netstat COnn/-c to find our established session and get the port number being used in this connection, as shown in Example 13-25.

Example 13-25 TSO command Netstat conn tcp tcpipd

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD          00:09:38
User Id  Conn     State
-------  ----     -----
FTPDD1   00000031 Listen
  Local Socket:   ::..21
  Foreign Socket: ::..0
FTPDD1   00002362 Establsh
  Local Socket:   ::ffff:10.1.1.40..21
  Foreign Socket: ::ffff:10.1.5.31..1028
```

Then we use the command Netstat ALL/-a with the port filter statement to verify that the established TCP connection matches the routing rule PBRrule2 and uses the routing table PBRTAB1; see Example 13-26.

Example 13-26 Netstat All tcp tcpipd (port 1028 command

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD          00:10:18
Client Name: FTPDD1                   Client Id: 00002362
  Local Socket: ::ffff:10.1.1.40..21
  Foreign Socket: ::ffff:10.1.5.31..1028
 ...
RoutingPolicy: Yes
       RoutingTableName: PBRTAB1
RoutingRuleName:  PBRrule2
```

```
     ReceiveBufferSize:  0000065536        SendBufferSize:     0000065536
     ReceiveDataQueued:  0000000000
     SendDataQueued:     0000000000
 ----
```

To verify the traffic is actually going through the expected OSA link, OSA20C0L, we used the command Netstat Devlinks and looked at fields; BytesOut and Outbound Packets, before and after the data traffic operation to see they had incremented on the expected OSA link, as shown in Example 13-14 on page 522.

Example 13-27 Netstat Devlinks tcp tcpipd (intfname OSA20C0L display command)

```
***** Before data transfer *******

 MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPD          00:27:59
 DevName: OSA20C0           DevType: MPCIPA
 DevStatus: Ready
 LnkName: OSA20C0L          LnkType: IPAQENET   LnkStatus: Ready
 ...
 Link Statistics:
    BytesIn                          = 176
    Inbound Packets                  = 2
    Inbound Packets In Error         = 0
    Inbound Packets Discarded        = 0
    Inbound Packets With No Protocol  = 0
    BytesOut                         = 69744
    Outbound Packets                 = 416
    Outbound Packets In Error        = 0
    Outbound Packets Discarded       = 0

**** After data tranfer ******

MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIPD         00:32:11
DevName: OSA20C0          DevType: MPCIPA
DevStatus: Ready
LnkName: OSA20C0L         LnkType: IPAQENET   LnkStatus: Ready
....
Link Statistics:
   BytesIn                         = 176
   Inbound Packets                 = 2
   Inbound Packets In Error         = 0
   Inbound Packets Discarded        = 0
   Inbound Packets With No Protocol  = 0
   BytesOut                         = 173076
   Outbound Packets                 = 532
   Outbound Packets In Error        = 0
   Outbound Packets Discarded       = 0
```

Part 4

Application-based security

In this part, we explain how you can use some built-in security functions for additional security. We identify a few of the common security exposures and tools that can be used to address the issues. The major emphasis is on Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Application Transparent-TLS (AT-TLS) features supported by the z/OS standard applications (TN3270 and FTP).

These applications provides client and server authentication and data encryption methods by using digital certificates with a protocol called Transport Layer Security (TLS). The TN3270 and FTP servers supports two TLS methods: TLS and AT-TLS. TLS is provided natively by internal code, and is the basic or less functional method. Application Transparent TLS (AT-TLS) is provided externally through the policy agent, and is the preferred method because of the additional functionality it provides.

14

Telnet security

In this chapter, we explain how you can implement and use the security features of the z/OS TN3270 server. The server supports native connection security through data overrun protection, network access control, and basic Transport Layer Security (TLS) functions. It also supports Application Transparent Transport Layer Security (AT-TLS) through configuration of the policy agent. It is defined as a *controlling* application when set up in AT-TLS support mode.

The following topics are discussed in this chapter.

Section	Topic
14.1, “Conceptual overview of TN3270 security” on page 540	An overview of TN3270 security features
14.2, “TN3270 native TLS connection security” on page 542	TN3270 native security functions
14.3, “Basic native TLS configuration example” on page 547	Implement a basic TN3270 connection security with native TLS
14.4, “TN3270 with AT-TLS security support” on page 553	TN3270 TLS connection security using AT-TLS
14.5, “Basic AT-TLS configuration example” on page 558	Implement a basic TN3270 connection security with AT-TLS
14.6, “Problem determination for Telnet server security” on page 577	Problem determination methods for TN3270 server issues
14.7, “Additional information sources for TN3270 AT-TLS support” on page 577	References to additional information about TN3270 security topics

14.1 Conceptual overview of TN3270 security

TN3270 security addresses issues related to client access to the z/OS system and to applications on the system and the protection of data that is exchanged between the client and the server. Security policies can be defined based on user IDs or on user IP addresses.

This section discusses the following topics:

- What is TN3270 security
- How TN3270 security works
- How TN3270 security can be applied

14.1.1 What is TN3270 security

TN3270 security can control client access to the z/OS system by verifying the user ID of clients requesting access to applications. The user ID can be obtained directly from the user by way of a prompting message that requests the user to supply the user ID and password. Client access can also be controlled by using digital certificate exchanges between the client and server. Network access control can be achieved by imposing security policies based on a combination of the client user ID, the client IP address, and system resources.

The TN3270 server provides client and server authentication and data encryption methods by using digital certificates with a protocol called Transport Layer Security (TLS). The TN3270 server supports two TLS methods: TLS and AT-TLS. TLS is provided natively by TN3270 internal code, and is the basic, less functional method. Application Transparent TLS (AT-TLS) is provided externally through the policy agent, and is the preferred method because of the additional functionality it provides.

14.1.2 How TN3270 security works

Data overrun protection can be used to protect against such things as the number of bytes received from a client without an end-of-record being received, the number of data segments queued to be sent to VTAM, the number of session requests received by a client in a period of time, and the number of chained request units received over a given application session without a corresponding end chain request unit.

Network Access Control (NAC) limits user access to certain IP security zones defined by the NETACCESS statement. A security product, such as RACF, is used to check the permission of user IDs to send data to or receive data from these security zones.

Transport Layer Security (TLS) makes use of digital certificates to authenticate the server to the client, to optionally authenticate the client to the server, and to provide encryption algorithms and keys to encrypt exchanged data. In order to activate TLS security for a specific connection, a special negotiation is required at the beginning of the client connection process. This negotiation involves the exchange and validation of one or more digital certificates, the sharing of a private and public key, and agreement on an encryption algorithm to be used to encrypt data exchanged between the client and server.

TN3270 security can control client access to the z/OS system by verifying the user ID of clients requesting access to applications. The user ID can be obtained directly from the user by way of a prompting message that requests the user to supply the user ID and password. Client access can also be controlled by using digital certificate exchanges between the client and server. Network access control can be achieved by imposing security policies based on a combination of the client user ID, the client IP address, and system resources.

In addition to native TLS support, the TN3270 server can use Application Transparent Transport Layer Security (AT-TLS) to manage secure connections. TLS managed by AT-TLS (TTLSPORT) supports more security functions than TLS managed by the TN3270 (SECUREPORT).

Note: If you have security parameters in a PARMSGROUP statement mapped to host names, you will not be able to emulate that mapping with AT-TLS. If you are not using PARMSGROUP to map to host names, we urge you to migrate to the use of AT-TLS as a consistent solution for all of your TCP applications.

Be aware of these AT-TLS capabilities and requirements when planning AT-TLS support for TN3270:

- Dynamically refresh a key ring.
- Support new or multiple key rings.
- Specify the label of the certificate to be used for authentication instead of using the default.
- Support SSL session key refresh.
- Support SSL session reuse.
- Support SSL sysplex session ID caching.
- Trace decrypted SSL data for Telnet in a data trace.
- Receive more granular error messages in syslog for easier debugging.
- Policy Agent must be active.
- TLS security defined within the TN3270 profile will continue to be available and can be implemented concurrently with AT-TLS.

Note: The TN3270 client, activated through the TSO TELNET command, does not support any SSL or TLS security protocols. Basic mode is the only method supported by the TN3270 client.

Use AT-TLS as a consistent security solution for all of your TCP-based applications. Certain applications (such as FTP and Telnet), however, have already been programmed to use the SSL/TLS toolkit directly and provide additional security functions such as application-negotiated SSL/TLS and certificate-based user ID mapping.

If, however, you do not need those additional functions, consider implementing FTP and TN3270 with the full AT-TLS support. AT-TLS is the preferred method of implementing TLS support in order to achieve a consistent solution for *all* of your TCP applications.

14.1.3 How TN3270 security can be applied

To implement data overrun protection, statements can be placed into the TN3270 server configuration profile dataset, into the TCP/IP stack configuration profile dataset, and into VTAM start options.

To implement network access control, statements are placed into the TN3270 server configuration profile dataset to define security zones and the users and IP address that belong to those zones. Then security permissions are defined to the SAF security package (RACF) to control access to the defined zones.

To implement native TLS support, statements are placed into the TN3270 server configuration profile dataset which instruct the server how to use the TLS key ring and digital certificate encryptions techniques. Then RACF definitions create the key ring, the digital certificates, and the permissions that the server and client users have during the TLS

verification process. The TN3270 server's native support of TLS is limited in its functionality when compared to that of the AT-TLS implementation.

If you already have TLS implemented for an existing instance of the TN3270 server, it is easy to migrate from TLS to AT-TLS support. AT-TLS support is implemented by moving most of the TLS definitions from the TN3270 server profile into the policy agent's AT-TLS configuration profile section. The TN3270 server is defined as an AT-TLS controlling application to the policy agent, and can retain its control of the secure relationship with the client. The AT-TLS implementation provides more flexibility and functionality than basic TLS, and is therefore the preferred method of implementing digital certificate support for the TN3270 server. The z/OS Configuration Assistant GUI is used to create the appropriate policy agent statements for AT-TLS.

14.2 TN3270 native TLS connection security

This section provides an overview of executing the Telnet server with native connection security features.

14.2.1 Description of TN3270 native connection security

Telnet server connection security addresses the following security requirements:

- Encrypting traffic between the TN3270 client and server
- Authenticating the server (to ensure that the client is indeed connecting to the server that the client expects to be connecting into)
- Authenticating the client (to only provide TN3270 services to specific clients and to customize the service provided to those clients)

To enable Telnet server connection security, the port over which the client connects needs to be configured with the appropriate security parameters.

This section discusses the following topics:

- Dependencies for Telnet server native connection security
- Benefits of Telnet server native connection security
- Considerations for Telnet server native connection security
- Secure TN3270 connections

Dependencies for Telnet server native connection security

To implement secure connections, the Telnet server task must have APF authorized access to the System SSL DLLs. Refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for details.

A key ring file is used to store one or more key pairs and certificates. To implement Telnet server connection security, a key ring file is required and must be populated with certificate information before the Telnet server accesses it on behalf of client connection processes.

The Telnet server profile configuration must be updated with appropriate statements supporting the security features selected. There are two essential profile statements required for establishing secure ports:

- SECUREPORT

  All TLS/SSL-enabled TN3270 ports must be defined by specifying a TELNETPARMS block for each port. The SECUREPORT port designation statement in the TELNETPARMS block indicates the port is capable of handling SSL connections.

- KEYRING

  The server certificate is required for the server authentication process defined by the SSL protocol. This certificate is stored in a key ring. The key ring type and location is specified in the KEYRING statement. Only one key ring can be used by the Telnet server at any time. The KEYRING statement can be coded in the TELNETGLOBALS or TELNETPARMS statement blocks.

  - KEYRING SAF *serverkeyring* statement can specify the keyring managed by RACF. The server certificate is connected to a key ring called SERVERKEYRING and is designated as the default certificate.
  - All uses of the KEYRING statement must specify the same data type and name. If coded in the TELNETGLOBALS statement block, any TELNETPARMS KEYRING values must match. If they do not, the port update is rejected.
  - If all secure ports (specified on the SECUREPORT statement) are in a *stopped* status when a profile update occurs, the KEYRING file is refreshed when a new secure port is activated.

    Note: If a secure port is active during a profile update, the KEYRING name cannot change. If a change is attempted, an error message is issued for this parameter and the profile update for the related port is rejected. To change the KEYRING name, *all* secure ports must first be stopped.

Three other optional, but important, profile statements can be used to assist in defining the types and level of security imposed on the port connections:

- CONNTYPE

  The type of connection (secure, non-secure, or any) to be used on the port is specified with this statement. If it is specified in the TELNETPARMS block, it sets the default type for the port. If it is specified in a PARMSGROUP block, it can override the port's default setting.

- CLIENTAUTH

  The CLIENTAUTH parameter indicates that the client must send a client certificate to the server. If this parameter is not specified, a client certificate is not requested during the SSL handshake and no certificate-based client authentication is done. The level of validation done depends on the option specified.

- ENCRYPTION

  The Telnet server has a default ordered list of encryption algorithms it will negotiate with the client, beginning with the lowest level of encryption to the highest, until it synchronizes (negotiates) to one that is compatible with the client.

  Use this statement to alter the order and to limit the list of supported algorithms. Perhaps you need to limit the choices to the two highest levels of encryption, not allowing the lower levels to even be attempted. If this parameter is not specified, all encryption algorithms that can be specified will be negotiated by TN3270, from low-level to high-level encryption.

  Each z/OS system level supports a specific set of encryption algorithms. The algorithms that can be specified are shown in Table 14-1.

Table 14-1 Encryption algorithms: default ordered list

ENCRYPTION algorithm_name	Abbreviation in Telnet displays
SSL_RC4_SHA	4S
SSL_RC4_MD5	4M

ENCRYPTION algorithm_name	Abbreviation in Telnet displays
SSL_AES_256_SHA	A2
SSL_AES_128_SHA	A1
SSL_3DES_SHA	3S
SSL_DES_SHA	DS
SSL_RC4_MD5_EX	4E
SSL_RC2_MD5_EX	2E
SSL_NULL_SHA	NS
SSL_NULL_MD5	NM
SSL_NULL_Null	NN

- There are more statements available for customizing secure ports. For a complete list of parameter statements and syntax, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For usage discussions, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Note: Optional security parameters for SECUREPORTs can be specified in the TELNETGLOBALS, TELNETPARMS, or PARMSGROUP blocks. The parameters specified in the PARMSGROUP block apply only to the clients mapped to the PARMSGROUP block by the PARMSMAP statement, and they override the parameters specified in the TELNETPARMS or TELNETGLOBALS block.

The parameters specified in the TELNETPARMS block apply to any connection for that port if not overridden by a PARMSGROUP parameter. The parameters specified in the TELNETGLOBALS block apply to any connection for any port if not overridden by a TELNETPARMS or PARMSGROUP parameter.

Benefits of Telnet server native connection security

Any time there are security concerns for accessing data, connecting to systems, or abusing system resources, security measures must be reviewed and implemented. The Telnet server supports security features that address three main areas of concern: Data Overrun conditions, Transport Layer Security, and Network Access Control. The most commonly used of these is Transport Layer Security (TLS). These three areas are discussed in the following sections:

- Benefits of implementing Data Overrun security
- Benefits of implementing Transport Layer Security (TLS)
- Benefits of implementing Network Access Control

Benefits of implementing Data Overrun security

A number of resource utilization limitations can be established for user connections. These limits can guard, for example:

- The number of bytes received from a client without an End Of Record (EOR) being received
- The number of data segments (RPLs) queued to be sent to VTAM
- The number of session requests received by a TN3270 client in a 10-second period

- The number of chained RUs that can be received over a given session from a host application without a corresponding end chain RU

Benefits of implementing Transport Layer Security (TLS)

The Telnet server provides the ability to secure TN3270 connections with the transport layer security (TLS) or secure sockets layer (SSL) protocol. A port that is configured to use the TLS/SSL protocol is referred to as a *secure port* or SECUREPORT. A port that does not use the TLS/SSL protocol is referred to as a *basic port*.

Connections are also either secure or basic. The flows between the Telnet server and VTAM (and SNA applications) are unaffected by TN3270 security configuration. The Internet Engineering Task Force (IETF) TLS-based Telnet Security Draft, supported by the z/OS Communications Server, allows a TN3270 negotiation to determine whether the client wants or supports TLS/SSL prior to beginning the handshake.

The default Telnet server action for a secure port is to first attempt a TLS/SSL handshake. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt will be made to negotiate TLS/SSL as defined by the TLS-based Telnet Security Draft. If the client responds that a secure connection is desired, the handshake is started. If the client rejects TLS/SSL, the connection will be closed. This enables installations to support SSL secure clients without knowing ahead of time which protocol the client will use.

The z/OS Communications Server Telnet server supports client authentication. Client authentication is done with the CLIENTAUTH parameter. Client authentication uses RACF services to translate the client certificate to an associated user ID to be used as a client identifier.

Benefits of implementing Network Access Control

Network Access Control (NAC) limits user access to certain IP security zones defined by the NETACCESS statement. A security product, such as RACF, is used to check the permission of user IDs to send data to or receive data from these security zones. The NAC user ID is based on the application's address space information. TCP/IP is an exempt user and has access to all zones.

If the Telnet server is running within the stack's address space, then TN3270 processes are also exempt from Network Access Control based on address space user ID information. The NACUSERID parameter enables Network Access Control checking for the Telnet server. This parameter is used to associate Telnet server ports with a user ID defined to the security server. The user ID specified on the NACUSERID parameter must be a valid user ID defined to the security server. If not, the TN3270 port will fail initialization.

Considerations for Telnet server native connection security

Note these potential issues with the use of TN3270 server connection security:

- The complexity of implementing security policies can delay a normal server implementation.
- Changes to security polices may adversely affect existing methods of connectivity and cause interruption of service.
- Costs associated with additional security measures may not be fully understood or provisioned.
- Implementation of security policies usually crosses departmental boundaries and requires close cooperation and planning. This effort is sometimes underestimated or even overlooked.

Depending on the business requirements of your organization, some type of access security and connection security will be necessary. Some organizations may implement security at the networking level, not requiring any authentication or encryption at the application level. Others may dictate that full level 3 digital certificate authentication and SSLV3/TLS encryption be the standard. Therefore, we do not recommend a specific implementation. However, we suggest that you familiarize yourself with the strong security features that TN3270 supports. Refer to the following sites for a better understanding of security methodology:

```
http://www.verisign.com/repository/crptintr.html
http://www.verisign.com/client/about/introCryp.html
```

Secure TN3270 connections

Depending on the business requirements of your organization, some type of access security and connection security is probably necessary. Your organization may implement security at the networking level, not requiring any authentication or encryption at the stack or application level. Other organizations may dictate that digital certificate authentication and SSLV3/TLS encryption be the standard. Here we demonstrate how to implement basic TN3270 SSL support both with and without client authentication. We suggest that you familiarize yourself with the strong security features that TN3270 supports.

Note: Most TN3270 security requirements could also be addressed on a stack-wide basis through the use of z/OS Communications Server IP Security (IPSec) and Application Transparent Transport Layer Security (AT-TLS) capabilities. For more information about this topic, refer to Chapter 9, “IP Security” on page 373, to Chapter 11, “Application Transparent Transport Layer Security” on page 443, and to Chapter 3, “Certificate management in z/OS” on page 31, according to your requirements.

14.2.2 Configuration of TN3270 native connection security

The Telnet server supports various methods and levels of connection security.

The following topics discuss these methods:

- Data overrun security
- Network Access Control
- Native Transport Layer Security

Data overrun security

The MAXRECEIVE, MAXVTAMSENDQ, MAXREQSESS, and MAXRUCHAIN parameters can be coded at all three parameter block levels (TELNETGLOBALS, TELNETPARMS, and PARMSGROUP) for different levels of granularity. Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for syntax details about each statement.

Network Access Control

When TN3270 is modified with a VARY TCPIP,,OBEYFILE command, the NACUSERIDs are reverified for the TN3270 ports defined in the data set referenced by the command. If a TN3270 port has NACUSERID NAC_name_1, you cannot use the VARY TCPIP,,OBEYFILE command to change that port’s NACUSERID to NAC_name_2. The port must first be stopped, and then started with the new NAC_name_2 using the VARY TCPIP,,OBEYFILE command.

On a restricted TCP/IP stack (not SYSMULTI), the NACUSERID will run under the SECLABEL of the TCP/IP stack. On an unrestricted TCP/IP stack (SYSMULTI), the NACUSERID will run under the default SECLABEL defined in the user profile. If no default

SECLABEL is configured in the user profile, SYSLOW is used by default. The NACUSERID user profile must be permitted to the SECLABEL it is to run under, or port activation will fail.

The NETACCESS statement in the TCP/IP profile is used to configure portions of your IP network into named security zones. Each defined security zone must have a SERVAUTH profile for the resource named EZB.NETACCESS.*sysname*.*tcpname*.*zonename*. The user ID associated with the TN3270 port must have READ access to the security zone that maps its bind address (0.0.0.0/32 for INADDR_ANY unless overridden by the PORT statement in the TCP/IP profile) and to every security zone that maps client IP addresses that TN3270 is to accept connections from on this port.

Native Transport Layer Security

To configure native Transport Layer Security (TLS), define a port number on a SECUREPORT statement. Multiple ports can be defined in TN3270 server. If you have a port configured for basic (non-secure) connections, you may add one or more ports for secure connections. If you want granular control of the connection type the client should use, you can implement it with a single secure port using client ID groups, or you can implement it with multiple secure ports, each with its own set of requirements. The security policies of your organization will dictate the method that is used. You can satisfy those security requirements by carefully choosing the statements you need and where to locate them within the TN3270 profile configuration.

For implementation example of the secure connection using the native TLS support, we show a minimum configuration scenario using native TLS in 14.3, “Basic native TLS configuration example” on page 547.

For further information about how the TN3270 standalone started task is configured, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

14.3 Basic native TLS configuration example

In this scenario we define a minimum configuration to implement the native TLS connection security. For an example of advanced configuration using client ID groups, refer to Appendix B, “Telnet security: advanced settings” on page 675.

We configure TN3270 with native TLS using a standalone started task TN3270D and TCPIPD stack on System SC33. TN3270 server uses TCP port 992 for native TLS secure connections. Stack affinity has been established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block.

Figure 14-1 on page 548 depicts the environment used for this scenario.

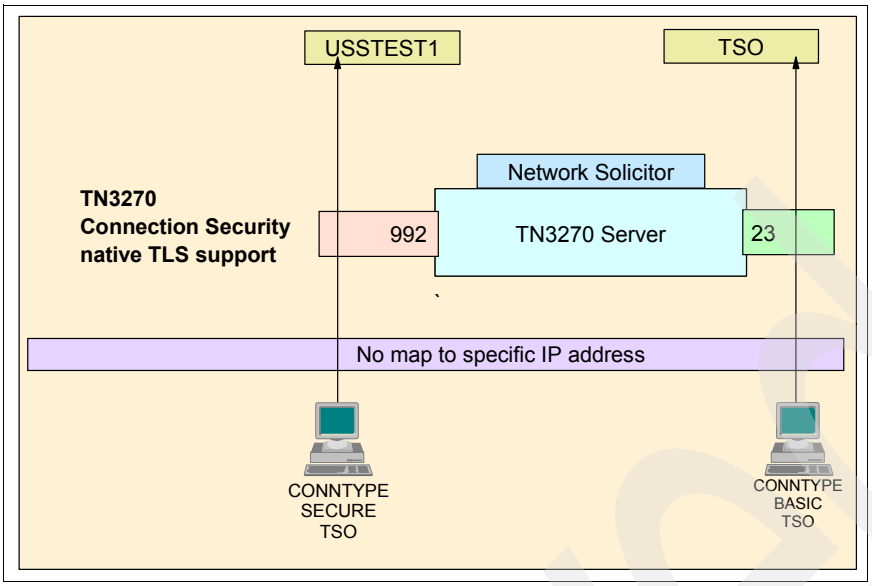


Figure 14-1 Minimum TN3270 native TLS configuration

14.3.1 Implementation tasks

The following tasks are required to enable native TLS/SSL support for TN3270, with server authentication:

- Generate the key ring and certificates.
- Add a TLS-enabled port and security parameters to TN3270 profile.

Generate the key ring and certificates

Note: Chapter 3, "Certificate management in z/OS" on page 31, contains detailed examples and explanations for the steps involved in preparing the key ring and certificates used in this scenario. Refer to that chapter for a complete discussion about using a shared key ring and SITE certificate.

Example 14-1 shows the RACF statements necessary to create the shared key ring, the CA certificate, and the SITE certificate used by this scenario.

Example 14-1 Sample RACF for key ring and certificates

```
racdcert certauth gencert                          - 1
  subjectsdn( o('IBM Corporation')                  -
              ou('ITSO Certificate Authority')      -
               C('US'))                             -
   NOTBEFORE(DATE(2007-09-11))                      -
   NOTAFTER(DATE(2008-09-11))                       -
   keyusage(certsign)                               -
   withlabel('CS19 ITSO CA1')
 setropts raclist(facility) refresh
 racdcert certauth list

 racdcert site gencert subjectsdn(cn('ITSO.IBM.COM')   - 2
    o('IBM Corporation')                            -
    ou('ITSO CS19 Shared SITE')                     -
    C('US'))                                        -
    withlabel('CS19 ITSO SharedSite1')              -
```

```
   signwith(certauth label('CS19 ITSO CA1')
racdcert site list

racdcert ID(TCPIP) ADDRING(SharedRing1) 3

racdcert ID(TCPIP) CONNECT(CERTAUTH                  - 4
                   LABEL('CS19 ITSO CA1')            -
                   RING(SharedRing1)                 -
                   USAGE(CERTAUTH)

racdcert ID(TCPIP) CONNECT(SITE                      - 5
                   LABEL('CS19 ITSO SharedSite1')    -
                   RING(SharedRing1)                 -
                   DEFAULT                           -
                   USAGE(PERSONAL)

setropts raclist(DIGTRING) refresh
setropts raclist(DIGTCERT) refresh
racdcert listring(*) id(TCPIP)
```

1 Creates a self-signed Certificate Authority certificate.

2 Creates a SITE certificate.

3 Creates a key ring.

4 Connects a self-signed Certificate Authority certificate to a key ring.

5 Connects a SITE certificate to a key ring.

Add a TLS-enabled port and security parameters to TN3270 profile

ConfigureTN3270 to include one or more TLS/SSL-enabled ports and specify the name of the key ring created in the previous step in the TELNETGLOBALS block or the TELNETPARMS block. The Telnet server profile configuration must be updated with appropriate statements supporting the security features selected.

The TN3270 profile for native TLS secure connection using port 992 is shown in Example 14-2.

Example 14-2 Minimum TN3270 profile for native TLS secure connection

```
TELNETGLOBALS
   TCPIPJOBNAME TCPIPD
   TNSACONFIG ENABLED COMMUNITY j0s9m2ap AGENT 161
   TELNETDEVICE IBM-3277     SNX32702,SNX32702
   TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
   TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
   TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
   TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
   TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
   TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
   TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
   TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
   TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
   TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
   TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
   TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
   TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
```

```
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
ENDTELNETGLOBALS
;
TELNETPARMS
    SECUREPORT 992         1 ; Port 992 supports native TLS
    CONNTYPE SECURE        2 ; Support SSL
    KEYRING SAF TCPIP/SharedRing1       ; Keyring shared by servers
    CLIENTAUTH NONE        3 ; Client Certificate none
    ENCRYPT SSL_DES_SHA    4 ; Ciphers used
            SSL_3DES_SHA
    ENDENCRYPT
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
      SC33DS01..SC33DS99
  ENDDEFAULTLUS

  USSTCP   USSTEST1        ; Use USSTABLE USSTEST1
  ALLOWAPPL SC3*           ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    SECUREPORT 992 ; Port 23 supports basic(non-secure) connections
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC33DB01..SC33DB99
  ENDDEFAULTLUS

  DEFAULTAPPL SC33TS        ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
```

```
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

1 Port 992 is used for native TLS secure connections.

2 Always use secure connection.

3 The name of the key ring created in the previous step.

4 The list of supported ciphers on this port.

Start the TN3270 server

To apply the new definition, start or restart the TN3270 server. The OBEYFILE command can be used also, but it is only effective for new connections. Example 14-3 shows the messages given at the initialization of the TN3270 server.

Example 14-3 Starting the TN3270 server

```
S TN3270D
.....
EZZ6001I TELNET SERVER STARTED
.....
EZZ6003I TELNET LISTENING ON PORT   992 1
EZZ6003I TELNET LISTENING ON PORT    23 2
```

1 The native TLS port 992 is now active.

2 The basic connection port 23 is also active (definition is not shown in this chapter).

14.3.2 Activation and verification

To verify the configuration, the following commands are useful.

- Use TELNET CONN displays to show TN3270 connections.
- Display Telnet CONN, CONN detail to show connection TLS information.
- Display PROF to show profile SSL information.

We used IBM Personal Communications (PComm) V5.7 to establish the connection. Because we used a self-signed certificate, we downloaded and installed the certificate of Certificate Authority into the PComm.

Use TELNET CONN displays to show TN3270 connections

Example 14-4 shows the display of existing TN3270 connections using the TELNET CONN command.

Example 14-4 Display Telnet CONN to see TN3270 connections

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 517
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- ---------------------- -------- --------  --- --------
00008663 DS ::FFFF:10.1.100.224..1811                          1
                                  SC33DS01 SC33TS04  TAE SNX32702   2
----- PORT:   992 3 ACTIVE               PROF: CURR CONNS:      1
------------------------------------------------------------
```

```
4 OF 4 RECORDS DISPLAYED
```

1 The client IP address and port number.

2 The LUNAME, APPLID, LOGMODE used for the connection.

3 The connection uses port 992.

Display Telnet CONN, CONN detail to show connection TLS information

Example 14-5 shows the display of a detailed information, including TLS, for a specific connection. Specify the connection ID that was displayed in Example 14-4 on page 551 for CONN= option.

Example 14-5 Display Telnet CONN, CONN detail for a specific connection

```
D TCPIP,TN3270D,T,CONN,CONN=8663,DETAIL
EZZ6065I TELNET CONNECTION DISPLAY 520
  CONNECTED: 17:23:03  09/24/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00008663   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..1811 1
    DESTIP..PORT: ::FFFF:10.1.1.40..992
    LINKNAME: VIPA1L
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  SECURE        ACCESS: SECURE  DS TLSV1 2
  PROTOCOL: TN3270E          DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                       NEWENV FUNCTIONS: --
  LUNAME: SC33DS01
  APPL: SC33TS04
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                    OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                   >*DEFLUS*                   --------
    DEFLT APPL: **N/A**
    USS TABLE:  NL (NULL)
                   >USSTEST1                   P-------
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  SSS-E*---  *---ST-  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  SSS*E****  *P**STS  SDD* TP-CURR
  *M*****  **TSBTQ***RT  ECF  SSS*E****  *P**STS  SDD* <-FINAL 3
35 OF 35 RECORDS DISPLAYED
```

1 The client IP address and port number.

2 The connection is secure, and uses DS (SSL_DES_SHA) cipher. Refer to Table 14-1 on page 543 for the complete list of supported ciphers.

3 The connection is secure and encrypted. Each letter of PCKLECXN2 stands for one of the SECURITY parameters listed, in the same order, in Example 14-6.

Display PROF to show profile SSL information

The Display PROF command shows the information about the profile being used. Example 14-6 shows the profile defined for port 992.

Example 14-6 Display Telnet PROFILE for TLS information

```
D TCPIP,TN3270D,T,PROF,PORT=992,DET,MAX=*
EZZ6080I TELNET PROFILE DISPLAY 376
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  SSS-E*---  *---ST-  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  SSS*E****  *P**STS  SDD* CURR
.....
 SECURITY
   SECUREPORT            992 1
   CONNTYPE          SECURE 2
   KEYRING           SAF TCPIP/SharedRing1 3
   CRLLDAPSERVER     NONE
   ENCRYPTION        DS,3S 4
   CLIENTAUTH        NONE 5
   NOEXPRESSLOGON
   NONACUSERID
   NOSSLV2
 TIMERS
.....
   SSLTIMEOUT              5
.....
   KEYRING           SAF TCPIP/SharedRing1 6
```

1 The port 992 is used.

2 The port is for secure connection.

3 The name of the key ring in use.

4 The list of ciphers begin used (DS for SSL_DES_SHA and 3S for SSL_3DES_SHA). Refer to Table 14-1 on page 543 for the complete list of supported ciphers.

5 The client authentication is not used.

6 The key ring used is SharedRing1, which is managed by an SAF product (RACF, in our case).

14.4 TN3270 with AT-TLS security support

z/OS Communications Server provides the Application Transparent - Transport Layer Security (AT-TLS) support for TN3270E to implement TLS connection security.

This section discusses the following topics:

- Description of TN3270 AT-TLS support
- Configuration of TN3270 AT-TLS support

- Activation and verification

14.4.1 Description of TN3270 AT-TLS support

TN3270 with AT-TLS provides security just like native TLS connection security:

- Encrypting traffic between the TN3270 client and server
- Authenticating the server (to ensure that the client is indeed connecting to the server that the client expects to be connecting to)
- Authenticating the client (to only provide TN3270 services to specific clients and to customize the service provided to those clients)

Native TLS connection security uses a direct interface to the System Secure Sockets Layer (System SSL). TN3270 with AT-TLS uses AT-TLS API to implement secure connections. The security parameters are defined in AT-TLS policies instead of in the Telnet profile, and are loaded into the stack by Policy Agent. This section discusses the following TN3270 AT-TLS topics:

- Dependencies
- Advantages
- Considerations

Dependencies

TN3270 with AT-TLS needs the Policy Agent to define the security parameters. The Policy Agent uses a flat file for AT-TLS configuration. The IBM Configuration Assistant for z/OS Communication Server can be used to generated the AT-TLS configuration flat file. You may also code it directly to the file. We recommend using IBM Configuration Assistant for z/OS Communication Server to avoid syntax errors.

Native TLS connection security is referred to as the SECUREPORT statement, and the AT-TLS security configuration is referred to as the TTLSPORT statement. SECUREPORT and TTLSPORT may exist in the same TN3270 profile and run concurrently.

The following security parameters, which are used for the SECUREPORT statement, should be omitted for the TTLSPORT configuration because the equivalent statements are defined in the AT-TLS policy configuration.

KEYRING Key ring specification
ENCRYPTION Cipher specification
CLIENTAUTH Client authentication level
CRLLDAPSERVER CRL LDAP server specification
SSLV2 or NOSSLV2 SSLV2 protocol usage
SSLTIMEOUT Handshake timeout

Table 14-2 describes the equivalent statement in AT-TLS policy configuration for these security parameters.

Table 14-2 The equivalent statement in AT-TLS configuration for security parameters

Telnet profile statement	Equivalent statement in AT-TLS configuration
KEYRING	Key ring in: TTLSKeyRingParms within TTLSEnvironmentAction

Telnet profile statement	Equivalent statement in AT-TLS configuration
SSLV2	SSLv2 in: TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction or TTLSConnectionAdvancedParms within TTLSConnectionAction
SSLTIMEOUT	HandshakeTimeout in: TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction or TTLSConnectionAdvancedParms within TTLSConnectionAction
ENCRYPTION	TTLSCipherParms in: TTLSConnectionAction or TTLSEnvironmentAction
CRLLDAPSERVER	GSK_LDAP_Server and GSK_LDAP_Server_Port in: TTLSGskLdapParms within TTLSEnvironmentAction
CLIENTAUTH NONE	HandshakeRole Server in: TTLSConnectionAction or TTLSEnvironmentAction
CLIENTAUTH SSLCERT	HandshakeRole ServerWithClientAuth and ClientAuthType required in: TTLSConnectionAction or TTLSEnvironmentAction/ TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction
CLIENTAUTH SAFCERT	HandshakeRole ServerWithClientAuth and ClientAuthType SAFCHECK in: TTLSConnectionAction or TTLSEnvironmentAction/ TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction

The following combinations of encryption and authentication ciphers are available in an AT-TLS configuration.

- The entire SSL V3/TLS V1 cipher choices in preferred order are listed in Table 14-3.

Table 14-3 Entire SSL V3/TLS V1 cipher choices in preferred order

Cipher name	Abbreviation in Telnet displays
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	0x39
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	0x38

Cipher name	Abbreviation in Telnet displays
TLS_DH_RSA_WITH_AES_256_CBC_SHA	0x37
TLS_DH_DSS_WITH_AES_256_CBC_SHA	0x36
TLS_RSA_WITH_AES_256_CBC_SHA	0x35
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	0x33
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	0x32
TLS_DH_RSA_WITH_AES_128_CBC_SHA	0x31
TLS_DH_DSS_WITH_AES_128_CBC_SHA	0x30
TLS_RSA_WITH_AES_128_CBC_SHA	0x2F
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	0x16
TLS_DHE_RSA_WITH_DES_CBC_SHA	0x15
TLS_DHE_DSS_WITH_3DES_EDE_CDC_SHA	0x13
TLS_DHE_DSS_WITH_DES_CBC_SHA	0x12
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	0x10
TLS_DH_RSA_WITH_DES_CBC_SHA	0x0F
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	0x0D
TLS_DH_DSS_WITH_DES_CBC_SHA	0x0C
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0x0A
TLS_RSA_WITH_DES_CBC_SHA	0x09
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	0x06
TLS_RSA_WITH_RC4_128_SHA	0x05
TLS_RSA_WITH_RC4_128_MD5	0x04
TLS_RSA_EXPORT_WITH_RC4_40_MD5	0x03
TLS_RSA_WITH_NULL_SHA	0x02
TLS_RSA_WITH_NULL_MD5	0x01
TLS_NULL_WITH_NULL_NULL	0x00

- The entire SSL Version 2 cipher choices in preferred order are listed in Table 14-4.

Table 14-4 SSL Version 2 cipher choices in preferred order

Cipher name	Abbreviation in Telnet displays
TLS_DES_192_EDE3_CBC_WITH_MD5	0x7
TLS_DES_64_CBC_WITH_MD5	0x6
TLS_RC2_CBC_128_CBC_EXPORT40_WITH_MD5	0x4
TLS_RC2_CBC_128_CBC_WITH_MD5	0x3

Cipher name	Abbreviation in Telnet displays
TLS_RC4_128_EXPORT40_WITH_MD5	0x2
TLS_RC4_128_WITH_MD5	0x1

The CONNTYPE statement can be defined for TTLSPORT the same way as in the SECUREPORT for granular control of the connection type the client should use. The AT-TLS policy can define the connectivity rule based on destination or source IP address or port numbers, and in some cases it can be a replacement for the PARMSGROUP and PARMSMAP statement in the TN3270 profile.

Advantages

AT-TLS supports all the functions System SSL provides, including the following functions which native TLS connection security do not support.

- Dynamically refresh a key ring
- Support new or multiple key rings
- Specify the label of the certificate to be used for authentication instead of using the default certificate
- Support SSL session key refresh
- Support SSL session reuse
- Support ID cashing for SSL sessions in a sysplex
- Trace decrypted SSL data for Telnet in a data trace
- Receive more granular error messages in syslog for easier debugging

We recommend using AT-TLS connection security over native TLS connection security, because AT-TLS continues to be updated as new System SSL functions become available.

Considerations

The AT-TLS policy can map security parameters to specific clients or client groups, based on source and destination IP address or port number. In some cases the AT-TLS policy can be a replacement for client mapping statements such as PARMSGROUP and PARMSMAP. However, the AT-TLS policy cannot map the security parameters to the host name of the client. If you want to identify the client with host name to map to specific security parameters, use native TLS security instead of AT-TLS.

14.4.2 Configuration of TN3270 AT-TLS support

To implement AT-TLS secure connections, define a port number on a TTLSPORT statement. Multiple ports can be defined in the TN3270 server. If you have a port configured for basic (non-secure) connections, you may add one or more ports for secure connections. If you prefer granular control of the connection type the client should use, implement it with a single secure port using client ID groups, or implement it with multiple secure ports, each with its own set of requirements. The security policies of your organization will dictate the method that is used. You can satisfy those security requirements by carefully choosing the statements you need and where to locate them within the TN3270 profile configuration.

The TN3270 server with AT-TLS uses the Policy Agent to define security parameters.

As an implementation example of a secure connection using AT-TLS, we show a minimum configuration scenario using AT-TLS in 14.5, “Basic AT-TLS configuration example” on page 558.

For further information about how the TN3270 standalone started task is configured, refer to *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

14.5 Basic AT-TLS configuration example

In this scenario we define a minimum configuration to implement AT-TLS connection security. For an example of an advanced configuration using client ID groups, refer to Appendix B, “Telnet security: advanced settings” on page 675.

We configure TN3270 with AT-TLS using a standalone started task TN3270D and TCPIPD stack on System SC33. The TN3270 server uses TCP port 4992 for native TLS secure connections. Stack affinity has been established for the Telnet server by using the TCPIPJOBNAME statement within the TELNETGLOBALS block.

The configuration is almost identical to the scenario in shown in 14.3, “Basic native TLS configuration example” on page 547. Notice, however, that some of the security parameters are omitted from the TN3270 profile because the equivalent statements are defined in the AT-TLS policy.

Figure 14-2 depicts the environment we used for this scenario.

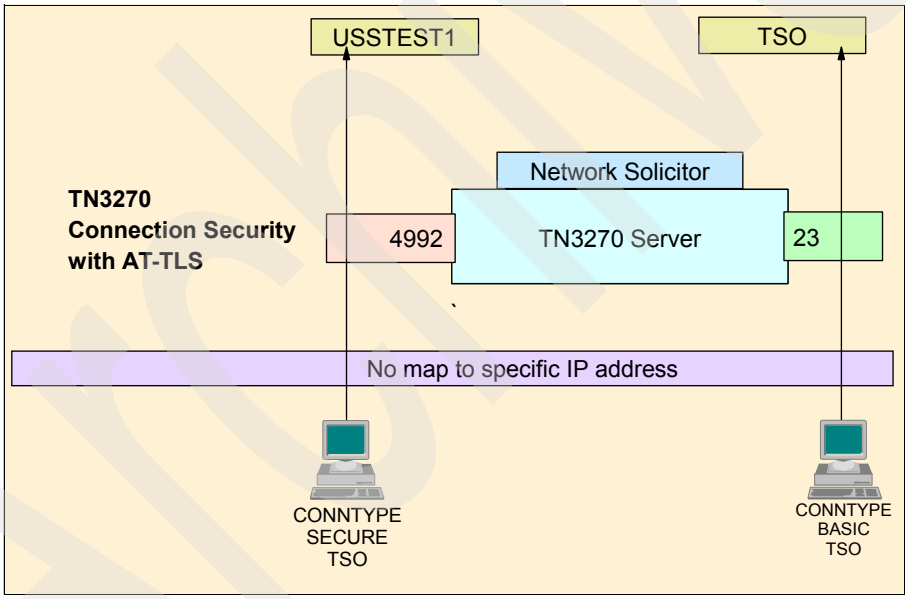


Figure 14-2 TN3270 with minimum AT-TLS configuration diagram

14.5.1 Implementation tasks

Perform the following tasks to configure TN3270 AT-TLS support:

- Set up the Policy Agent.
- Create a certificate.
- Add authorization for the **pasearch** command in RACF.
- Modify the TCP/IP profile.
- Define AT-TLS policies.
- Upload the policy to z/OS.
- Modify the Policy Agent configuration file.
- Define the AT-TLS port in the TN3270 configuration file.

Set up the Policy Agent

We set up the Policy Agent as shown in 5.2, “Implementing PAGENT on z/OS” on page 229.

Create a certificate

We can continue to use the certificate created for native TLS connection security as shown in 14.2, “TN3270 native TLS connection security” on page 542.

Add authorization for the pasearch command in RACF

If users other than superusers need to issue a **pasearch** command, add authorization for the **pasearch** command in RACF. The **pasearch** command is a sensitive command so make sure you restrict the access for only administrators or operators. We set up the RACF authorization as shown in 5.2, “Implementing PAGENT on z/OS” on page 229.

Modify the TCP/IP profile

To enable AT-TLS to the TCPIP stack, add a TCPCONFIG TTLS statement in the TCP/IP profile; see Example 14-7. To apply the change, either restart the TCP/IP stack or use the OBEYFILE command.

Example 14-7 Modify TCP/IP profile

```
TCPCONFIG TTLS
```

Define AT-TLS policies

We used the IBM Configuration Assistant for z/OS Communication Server to define AT-TLS policies.

1. Start the IBM Configuration Assistant for z/OS Communication Server. In the Main Perspective panel, shown in Figure 14-3, click the option **Add a New z/OS image**.

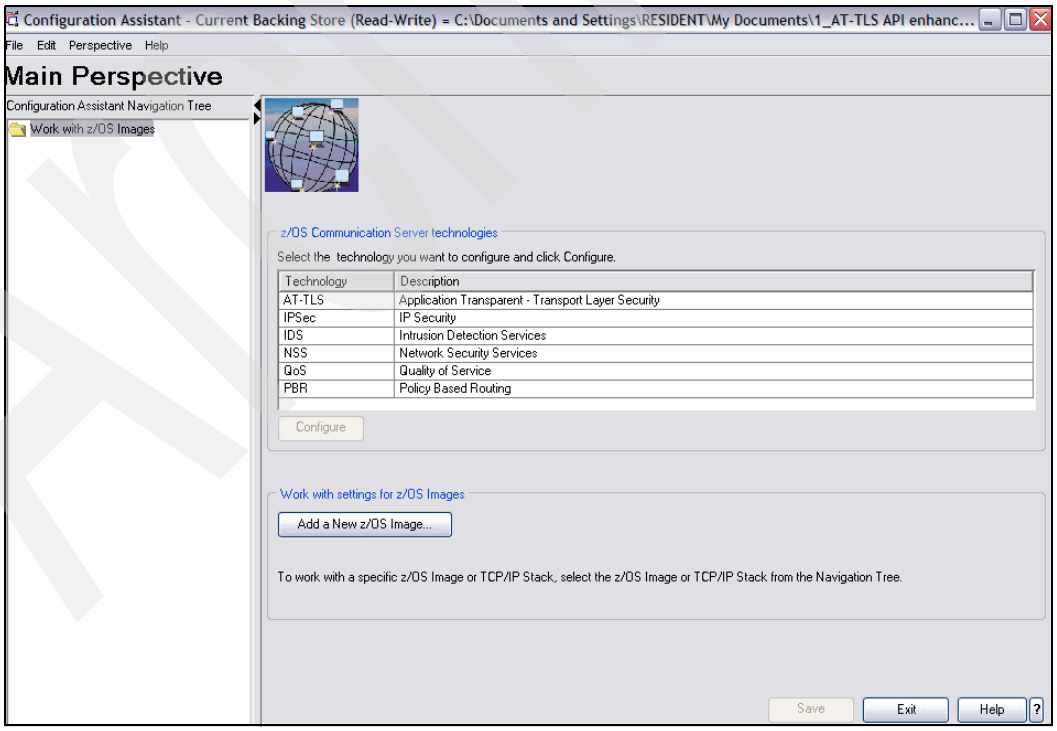


Figure 14-3 Main Perspective panel

2. Enter the name of the z/OS image (in our example, we typed in SC33). Click **OK**.

3. In the Main Perspective panel, click the **Add New TCP/IP stack** option, and enter the name of the TCP/IP stack (in our case, `TCPIPD`).
4. In the Main Perspective panel, select AT-TLS in the z/OS Communications Server technologies list. Click **Enable**, then click the **Configure** option.

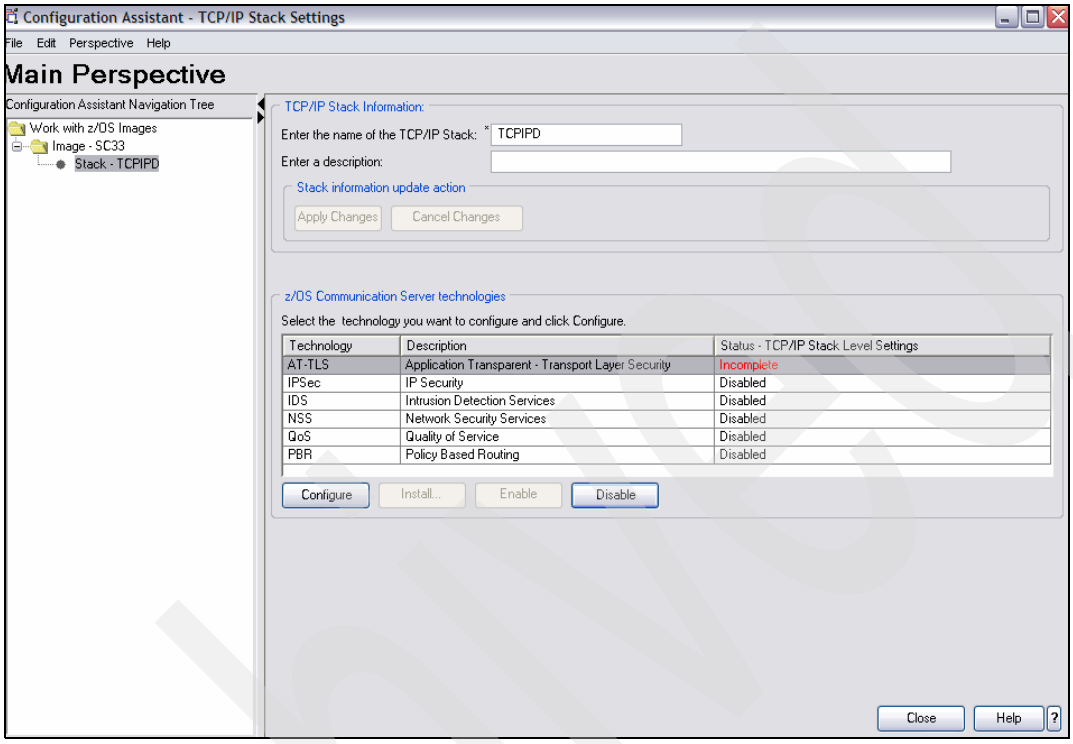


Figure 14-4 Main Perspective panel: selecting the policy type

5. This leads you to the AT-TLS Perspective panel, shown in Figure 14-5 on page 561. Notice the stack in the left pane shows `Incomplete Stack`, because the AT-TLS policy is not yet configured. Click the **Add** option.

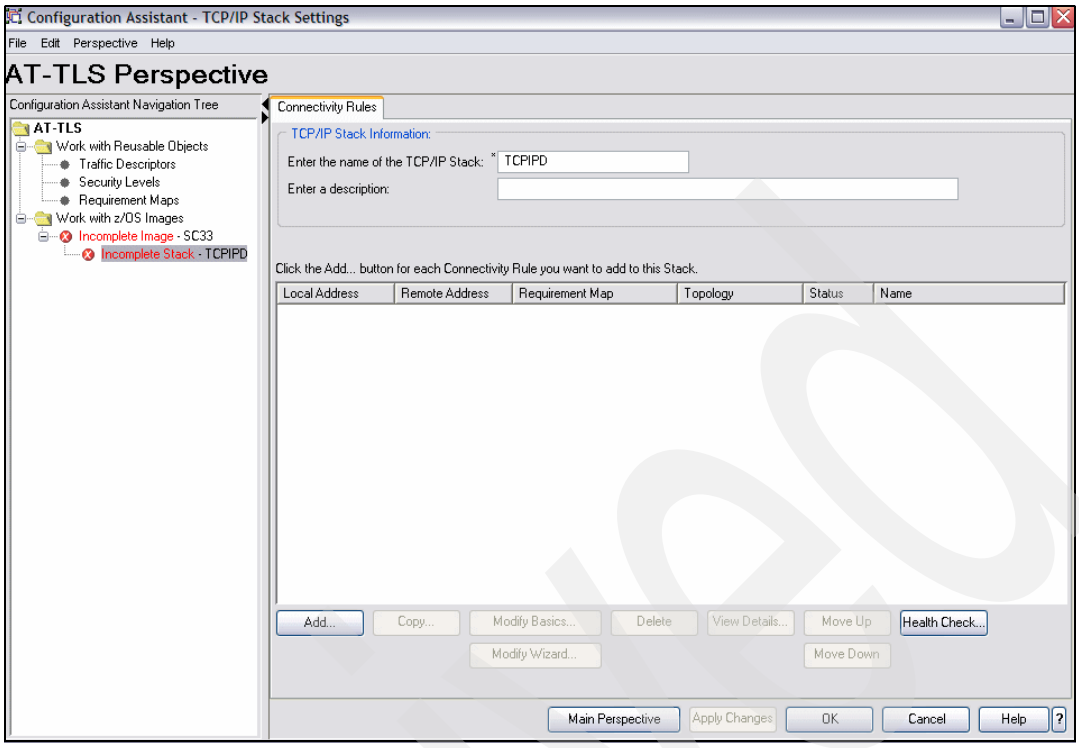


Figure 14-5 AT-TLS Perspective panel

6. In the New Connectivity Rule panel, click **Next**.
7. In the New Connectivity Rule: Data Endpoints panel, define the source IP address and destination IP address of the traffic that you want to apply this policy to. (In our example, we selected **All IP V4 Addresses** for source and destination IP address.) Also enter the connectivity rule name. (We named our policy `ATTLS_TN3270D_Rule1`). Click **Next**.

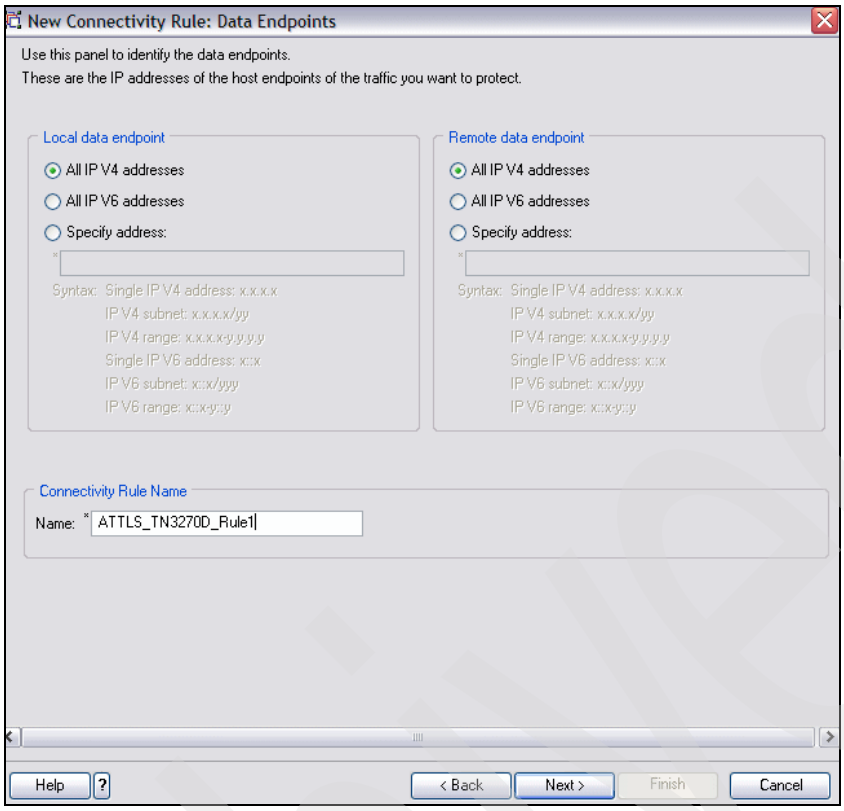


Figure 14-6 New Connectivity Rule: Data Endpoints panel

8. In the Select Requirement Panel, you define the description of the traffic you want to apply the policy. You may choose the predefined requirement map from the list if it fits your requirement. Select the predefined AT-TLS Sample entry and click **View Details**. You will see it contains the descriptions for TN3270E with TCP port 23 and CICS with TCP port 3000. (We used TCP port 4992 for TN3270E in our example, so we created a new requirement map instead of using the predefined one.) Click **Add**.
9. In the New Requirement Map panel, enter the name of the new requirement map. (We named it `ATTLS_TN3270D_ReqMap`). Select the **Work with Traffic Descriptors** option.

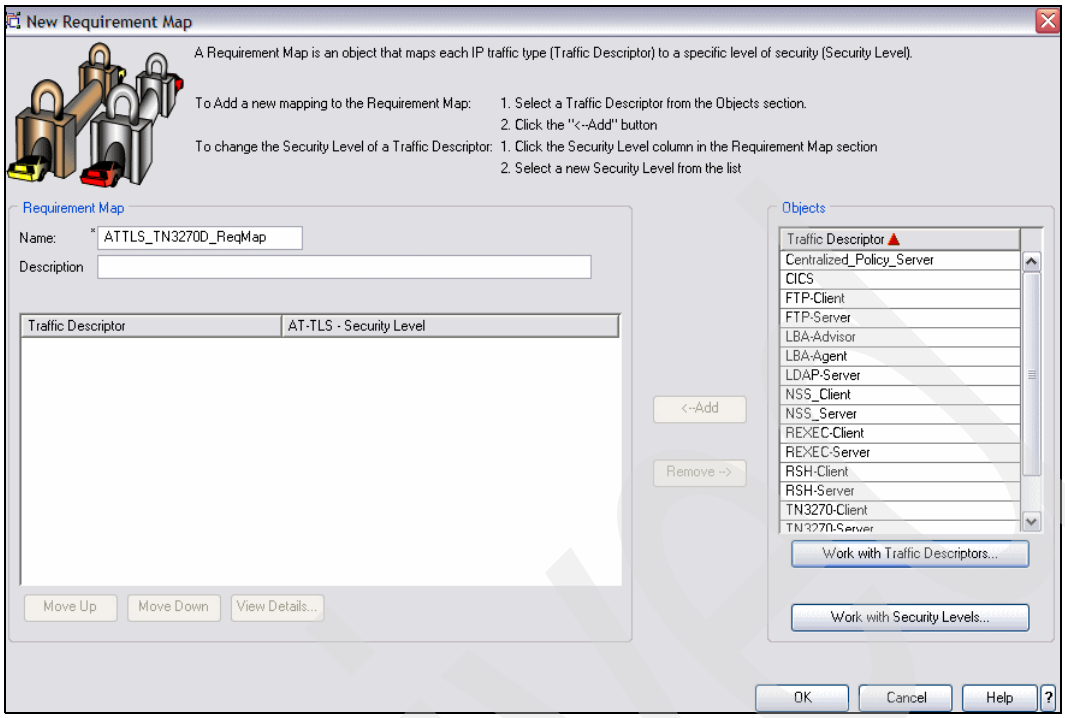


Figure 14-7 New Requirement Map panel: defining requirement map

10. Click **Add** to define a new traffic descriptor.

11. Enter the name of the new traffic descriptor. (We named it `ATTLS_TN3270D_4992`.) Click **Add** to define a new traffic descriptor.

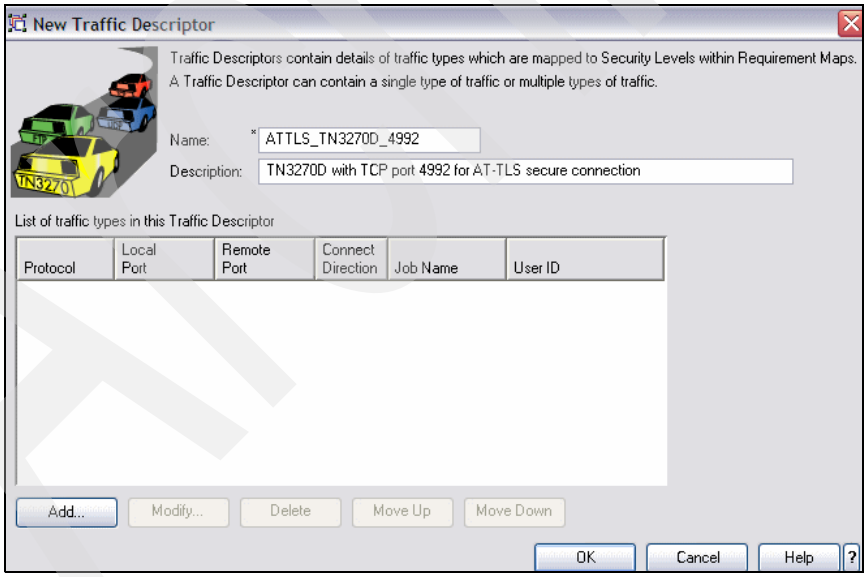


Figure 14-8 New Traffic Descriptor panel

12. Define the local port number and remote port number. (We applied the policy to the TN3270E server with TCP port number 4992 in our example, so we specified a local port number `4992` and the jobname `TN3270D`.) Click the **AT-TLS Advanced** option.

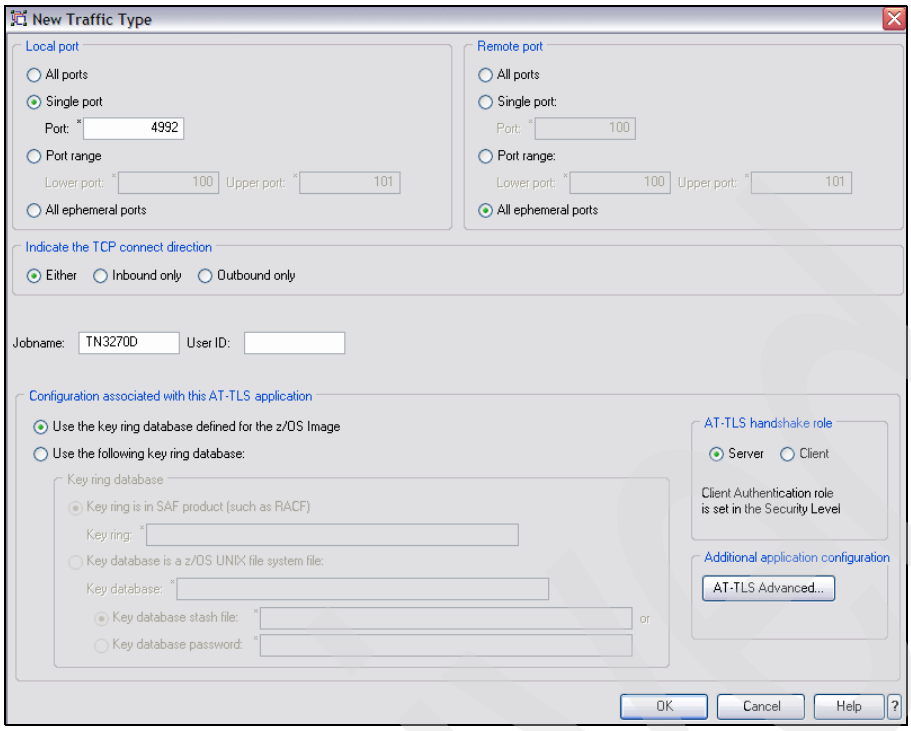


Figure 14-9 New Traffic Type panel: Defining the traffic description

13. In the Key Ring and Advanced AT-TLS settings, select the AT-TLS Tuning tab, and toggle the **Application Controlled** option to **On**. Click **OK**.

Note: The TN3270 profile SSL handshake timeout (SSLTIMEOUT) is 5 seconds by default with native TLS support. The AT-TLS handshake timeout is 10 seconds by default. If you want to make them match, specify the AT-TLS handshake timeout parameter in the AT-TLS Tuning tab.

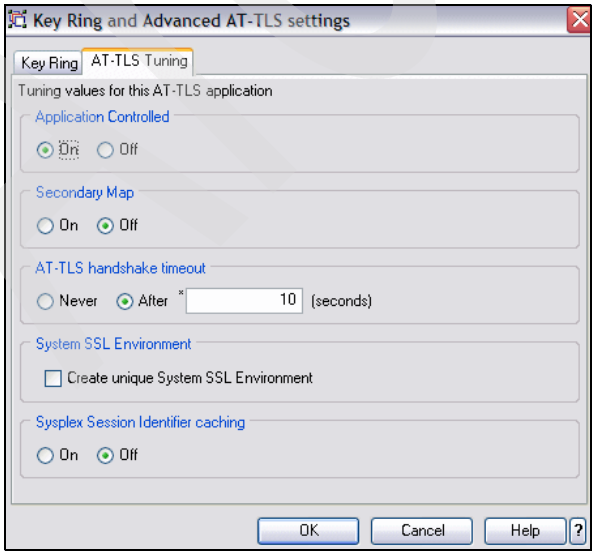


Figure 14-10 Key Ring and Advanced AT-TLS settings

14. In the New Traffic Descriptor panel, you will see the entry you have just created. Click **OK**.
15. In the Traffic Descriptor Objects panel, you will see the traffic descriptor entry you created. Click **OK**.
16. In the New Requirement Map panel, select the traffic descriptor you have created. (In our example, we selected `ATTLS_TN3270_4992`.) Click the **<--Add** option.
17. Select the AT-TLS Security Level. (In our example, we choose the **AT-TLS_Gold** option.) Click **OK**. Table 14-5 shows the list of IBM-Supplied security levels.

Table 14-5 IBM-supplied security levels

Security level	Type	Entire TLS Version 1/SSL Version 3 Cipher Suite in Preferred Order
Permit	No security	N/A
AT-TLS__Bronze (Low level of protection)	AT-TLS	0x02 - TLS_RSA_WITH_NULL_SHA
AT-TLS__Silver (Medium level of protection)	AT-TLS	0x09 - TLS_RSA_WITH_DES_CBC_SHA 0x0A - TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F - TLS_RSA_WITH_AES_128_CBC_SHA
AT-TLS__Gold (High level of protection)	AT-TLS	0x0A - TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F - TLS_RSA_WITH_AES_128_CBC_SHA
AT-TLS__Platinum (Extremely high level of protection)	AT-TLS	0x35 - TLS_RSA_WITH_AES_256_CBC_SHA

Note: Choose the security level depending on the supported ciphers of the host you establish the TLS connection with. You may also create a new security level to support the ciphers that fit your requirement by selecting the **Work with Security Levels** option.

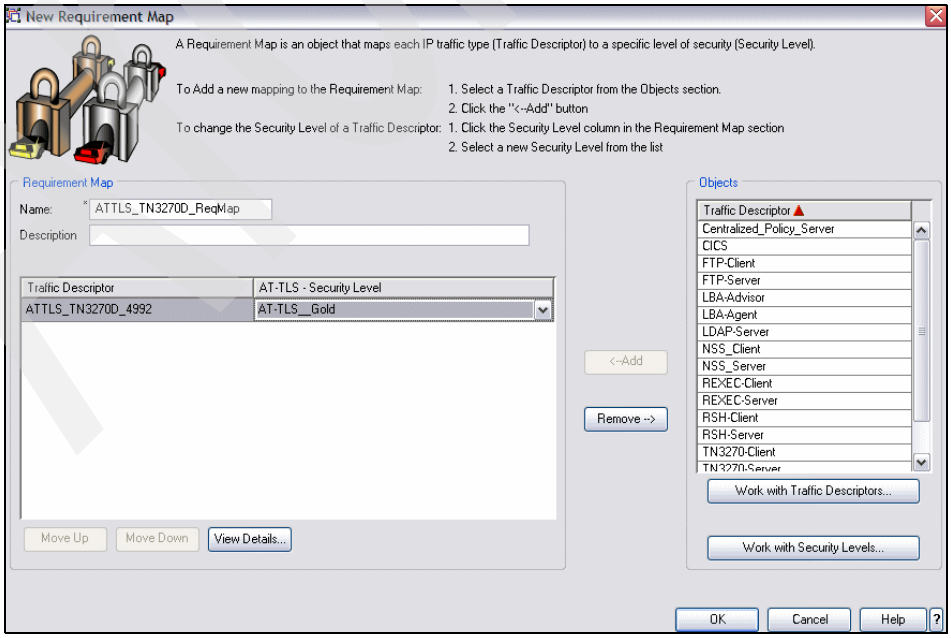


Figure 14-11 New Requirement Map panel: mapping traffic descriptor and security level

18. When the Requirement Map Tip panel appears, click **Proceed**.
19. Back at the New Connectivity Rules panel, select the requirement map you have created and click **Next**.
20. In the New Connectivity Rule: Finish panel, click **Finish**.
21. In the AT-TLS Perspective panel, as shown in Figure 14-12, you will see the defined connectivity rule but still see the Incomplete Image in the left pane because the KeyRing is not defined yet. Click the **Apply Changes** option to save the configuration.

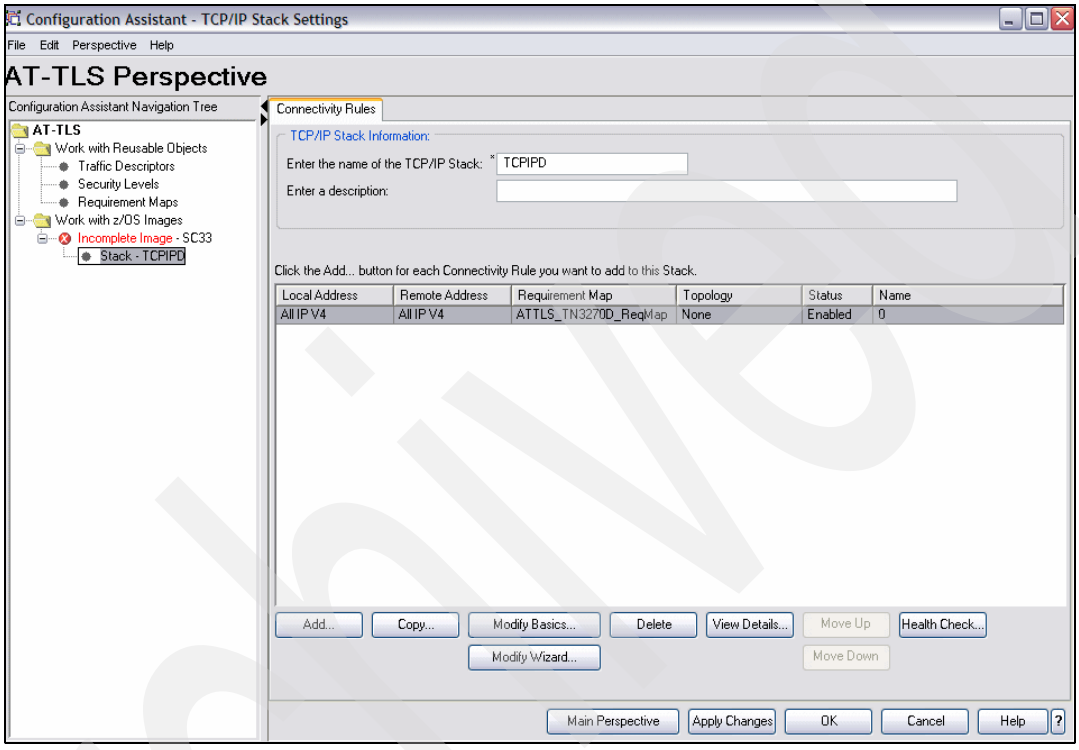


Figure 14-12 AT-TLS Perspective panel with defined connectivity rule

22. In the left pane, select the image name (`SC33`, in our case). Required AT-TLS Image Level Settings should be displayed. If they are not displayed, you may select the Image Level Settings tab in the right pane. Specify the key ring name or the key database name that you have created in the step “Create a certificate” on page 559. (In our example, we specified `SharedRing1`.) Click **OK**.

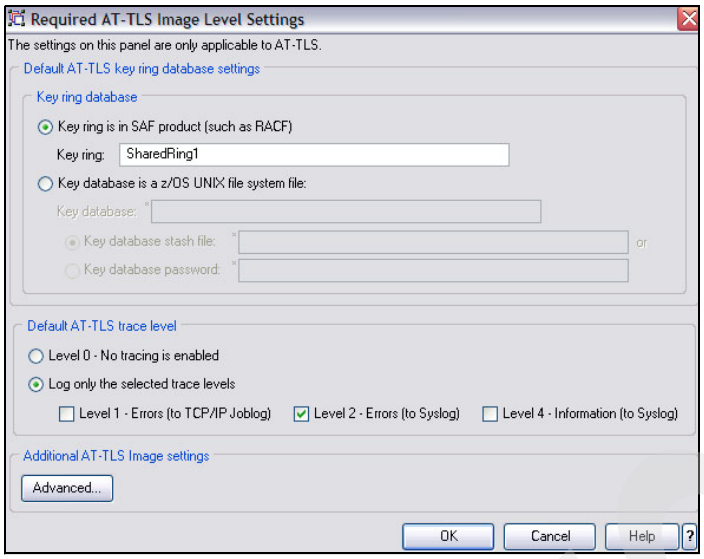


Figure 14-13 AT-TLS Image Level Settings panel: specifying the key ring

23. At this point, the policy definition is complete. Example 14-8 shows the configuration file generated by the IBM Configuration Assistant for z/OS Communication Server.

Example 14-8 Policy configuration flat file

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\AT-TLS\pagent33_TTLS.conf.bkstore
## FTP History:
## 2007-09-20 06:29:20  cs06 to 10.1.1.40
##
TTLSRule                          ATTLS_TN3270D_Rule1~1
{
  LocalAddrSetRef                 addr1
  RemoteAddrSetRef                addr1
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Jobname                         TN3270D
  Direction                       Both
  Priority                        255
  TTLSGroupActionRef              gAct1~ATTLS_TN3270D_4992
  TTLSEnvironmentActionRef        eAct1~ATTLS_TN3270D_4992
  TTLSConnectionActionRef         cAct1~ATTLS_TN3270D_4992
}
TTLSGroupAction                   gAct1~ATTLS_TN3270D_4992
{
  TTLSEnabled                     On
}
TTLSEnvironmentAction             eAct1~ATTLS_TN3270D_4992
{
  HandshakeRole                   Server
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~SC33
}
```

```
TTLSConnectionAction              cAct1~ATTLS_TN3270D_4992
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~AT-TLS__Gold
  TTLSConnectionAdvancedParmsRef  cAdv1~ATTLS_TN3270D_4992
}
TTLSConnectionAdvancedParms       cAdv1~ATTLS_TN3270D_4992
{
  ApplicationControlled           On
}
TTLSKeyringParms                  keyR~SC33
{
  Keyring                         SharedRing1
}
TTLSCipherParms                   cipher1~AT-TLS__Gold
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddrSet                         addr1
{
  Prefix                          0.0.0.0/0
}
PortRange                         portR1
{
  Port                            4992
}
PortRange                         portR2
{
  Port                            1024-65535
}
```

Upload the policy to z/OS

1. In the Configuration Assistant Navigation Tree pane, right-click the TCP/IP stack name (TCPIPD, in our example), and select the **Install Configuration Files** option.

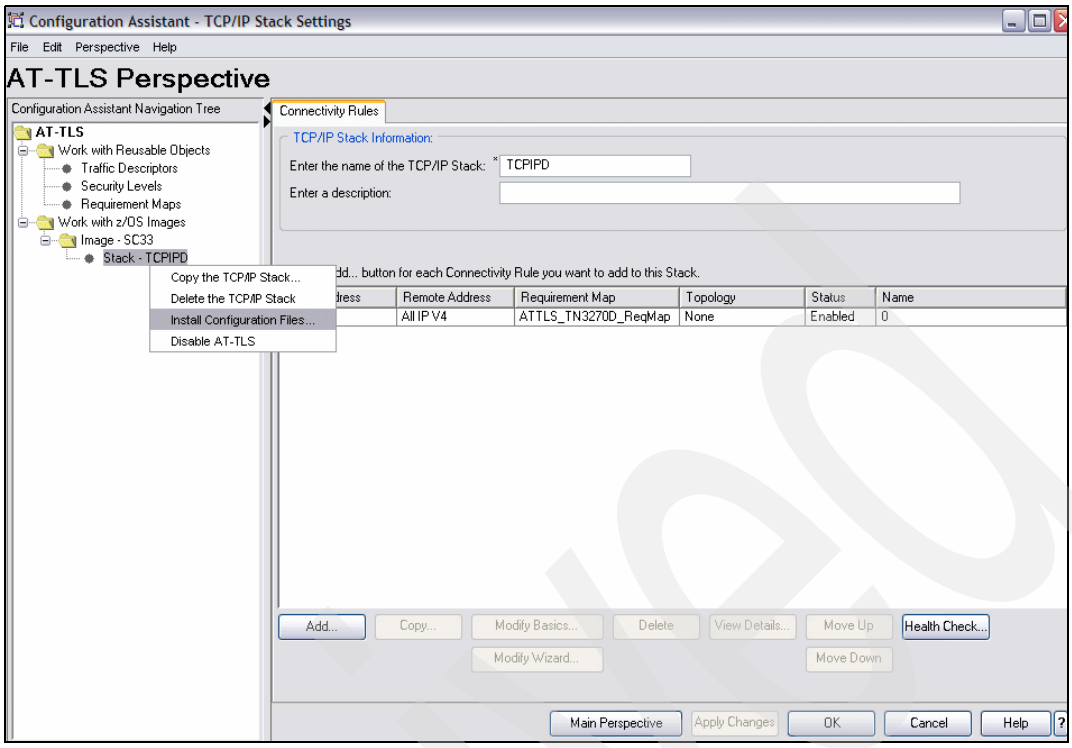


Figure 14-14 AT-TLS Perspective panel with defined policy

2. Select the *stackname* - AT-TLS: Policy Agent Stack Configuration in the list and click the **FTP...** option. If you do not want to use FTP to upload the configuration file directly to z/OS, select the **Show Configuration File** option instead and then select **Save As** option to save it locally for later transfer.

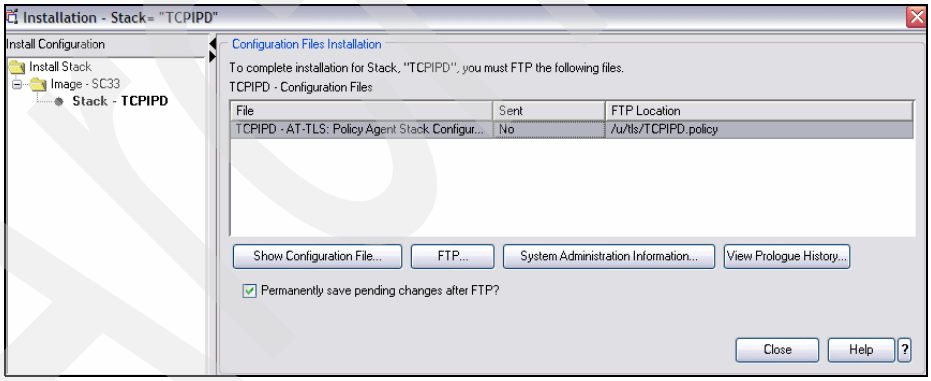


Figure 14-15 Installation panel: Installing Configuration File

3. Enter the FTP server IP address, port number, FTP user ID, and password. Also specify the filename to be saved as. Click **Send**.

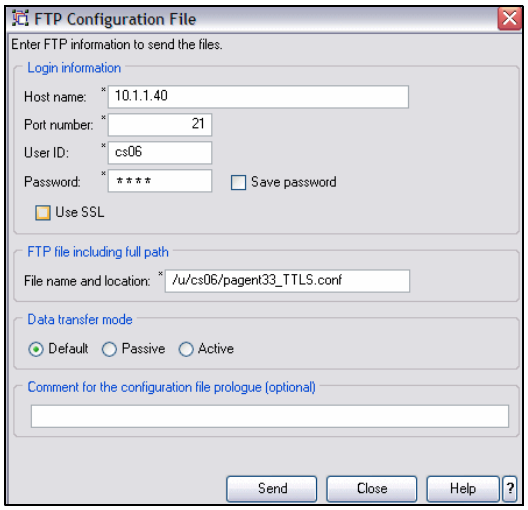


Figure 14-16 FTP Configuration File panel: transfer the configuration file to z/OS

4. When the FTP transfer is completed, the list in the Installation panel informs you of the transferred date and time in the Sent column; see Figure 14-17.

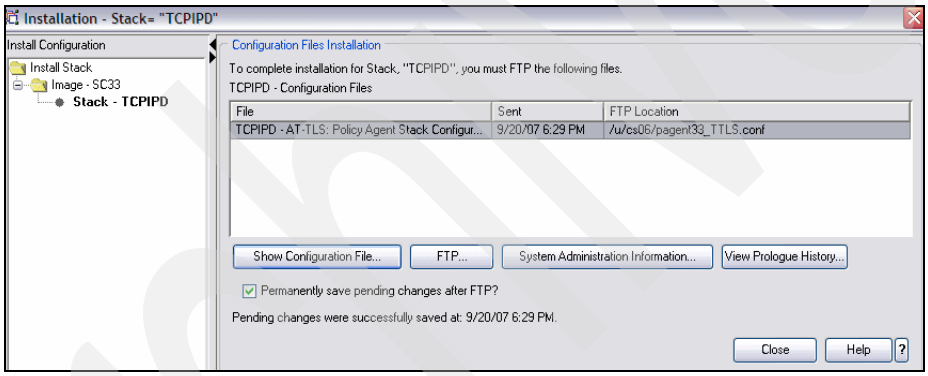


Figure 14-17 Completion of transfer through FTP

Modify the Policy Agent configuration file

Example 14-9 shows the main configuration file of the Policy Agent. It defines the stack-specific configuration file name for the TCPIPD stack.

Example 14-9 Main configuration file of Policy Agent

```
#  ******************************************************************
#  /SC33/etc/pagent33.conf
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH PURGE 600
```

Example 14-10 shows the stack-specific configuration file for THE TCPIPD stack. Add the TTLSConfig statement that points to the policy configuration file we created.

Example 14-10 Stack-specific configuration file of Policy Agent

```
#  ******************************************************************
#  /SC33/etc/pagent33_TCPIPD.conf
#  ******************************************************************
TTLSConfig     /etc/pagent33_TTLS.conf FLUSH PURGE
```

Define the AT-TLS port in the TN3270 configuration file

To enable AT-TLS security, specify the TTLSPORT statement. As shown in Example 14-11, TCP port 4992 is used for accepting the secure connections with AT-TLS.

Example 14-11 TELNETPARMS definition for AT-TLS port in TN3270 configuration file

```
TELNETPARMS
    TTLSPORT 4992                1 ; Port 4992 support AT-TLS
    CONNTYPE SECURE
;   KEYRING SAF TCPIP/SharedRing1 2 ; omit - defined in PAGENT
;   CLIENTAUTH NONE               2 ; omit - defined in PAGENT
;   ENCRYPT SSL_DES_SHA           2 ; omit - defined in PAGENT
;           SSL_3DES_SHA
;   ENDENCRYPT
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
ENDTELNETPARMS
;
BEGINVTAM
  PORT 4992
  DEFAULTLUS
      SC33DT01..SC33DT99
  ENDDEFAULTLUS

  USSTCP   USSTEST1         ; Use USSTABLE USSTEST1
  ALLOWAPPL SC3*            ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *       ; Allow all applications that have not been
                    ; previously specified to be accessed.
ENDVTAM
```

1 Port 4992 is used for the AT-TLS secure connection.

2 These security parameters are omitted from the TN3270 profile because they are defined in the Policy Agent.

14.5.2 Activation and verification

Perform the following tasks to activate and verify TN3270 AT-TLS support:

- Start the Policy Agent.
- Start the TN3270 server.
- Display PROF to show profile AT-TLS information.
- Display the AT-TLS profile using the **pasearch** command.
- Display CONN to show connection AT-TLS information.
- Display Netstat TTLS to show connection AT-TLS information.

Start the Policy Agent

Start the Policy Agent to enable the policy-based routing as shown in Example 14-12 on page 572. The message 1 shows the AT-TLS policy is processed and now in effect.

Example 14-12 Starting the Policy Agent

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS 1
```

If you have the Policy Agent started already, use the F *jobname*,REFRESH command as shown in Example 14-13. The message 2 informs you that the AT-TLS policy is loaded (or reloaded).

Example 14-13 Refreshing the Policy Agent

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS 2
```

Start the TN3270 server

Start the TN3270 server. Ensure that TN3270 server starts listening on the TTLS port 1.

Example 14-14 Starting the TN3270 Server

```
S TN3270D
$HASP100 TN3270D  ON STCINRDR
.....
EZZ6001I TELNET SERVER STARTED
.....
EZZ6003I TELNET LISTENING ON PORT  4992 1
EZZ6003I TELNET LISTENING ON PORT   992
EZZ6003I TELNET LISTENING ON PORT    23
```

Display PROF to show profile AT-TLS information

Verify that the profile is set up correctly. Use the Display Telnet,Profile command to display the profile.

Example 14-15 Telnet Profile Display

```
D TCPIP,TN3270D,T,PROF,PORT=4992,DETAIL
EZZ6080I TELNET PROFILE DISPLAY 132
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* CURR
.....
 SECURITY
   TTLSPORT             4992
   CONNTYPE           SECURE
   KEYRING            TTLS 1
   CRLLDAPSERVER      TTLS 1
   ENCRYPTION         TTLS 1
   CLIENTAUTH         TTLS 1
   NOEXPRESSLOGON
```

```
   NONACUSERID
   SSLV2             TTLS 1
 TIMERS
   INACTIVE             0 (OFF)
   PROFILEINACTIVE   1800
   KEEPINACTIVE         0 (OFF)
   PRTINACTIVE          0 (OFF)
   SCANINTERVAL       120
   TIMEMARK           600
   SSLTIMEOUT        TTLS 1
KEYRING          SAF TCPIP/SharedRing1 2
```

1 The AT-TLS policy is used for security parameters.

2 The key ring name specified in the AT-TLS policy configuration file is applied.

Display the AT-TLS profile using the pasearch command

The **pasearch** command can be used to display the AT-TLS policy configuration.

Example 14-16 The pasearch -t display

```
CS06 @ SC33:/u/cs06>pasearch -t

TCP/IP pasearch CS V1R9                  Image Name: TCPIPD
  Date:                 09/21/2007       Time:  17:33:58
  TTLS Instance Id:     1190328724

policyRule:             ATTLS_TN3270D_Rule1~1
  Rule Type:            TTLS
  Version:              3                Status:           Active
  Weight:               255              ForLoadDist:      False
  Priority:             255              Sequence Actions: Don't Care
  No. Policy Action:    3
  policyAction:         gAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Group
   Action Sequence:     0
  policyAction:         eAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Environment
   Action Sequence:     0
  policyAction:         cAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Connection
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00            To TimeOfDay:     24:00
   Fr TimeOfDay UTC:    04:00            To TimeOfDay UTC: 04:00
   TimeZone:            Local
  TTLS Condition Summary:                NegativeIndicator: Off
   Local Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   Remote Address:
    FromAddr:           0.0.0.0
    Prefix:             0
```

```
  LocalPortFrom:       4992             LocalPortTo:       4992
  RemotePortFrom:      1024             RemotePortTo:      65535
  JobName:             TN3270D          UserId:
  ServiceDirection:    Both

 TTLS Action:                 gAct1~ATTLS_TN3270D_4992
   Version:                   3
   Status:                    Active
   Scope:                     Group
   TTLSEnabled:               On
   CtraceClearText:           Off
   Trace:                     2
   TTLSGroupAdvancedParms:
    SecondaryMap:             Off
    SyslogFacility:           Daemon

 TTLS Action:                 eAct1~ATTLS_TN3270D_4992
   Version:                   3
   Status:                    Active
   Scope:                     Environment
   HandshakeRole:             Server
   TTLSKeyringParms:
    Keyring:                  SharedRing1
   TTLSEnvironmentAdvancedParms:
    SSLv2:                    Off
    SSLv3:                    On
    TLSv1:                    On
    ApplicationControlled:    Off
    HandshakeTimeout:         10
    ClientAuthType:           Required
    ResetCipherTimer:         0
   EnvironmentUserInstance:   0

 TTLS Action:                 cAct1~ATTLS_TN3270D_4992
   Version:                   3
   Status:                    Active
   Scope:                     Connection
   HandshakeRole:             Server
   TTLSConnectionAdvancedParms:
    ApplicationControlled:    On
   TTLSCipherParms:
    v3CipherSuites:
     0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
     2F  TLS_RSA_WITH_AES_128_CBC_SHA
```

Display CONN to show connection AT-TLS information

We used IBM Personal Communications (PComm) V5.7 to establish a secure session. Because we used self-signed certificate, we downloaded and installed the certificate of Certificate Authority into the PComm. We defined the PComm connection profile for the AT-TLS connection with exactly the same security parameters that we defined for the TN3270 native TLS secure connection.

Example 14-17 shows a secure connection.

Example 14-17 TLS secure connection using port 4992

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 754
         EN                                          TSP
```

```
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00009843 0A ::FFFF:10.1.100.224..2537
                                   SC33DT02 SC33TS06  TAE SNX32702
----- PORT:  4992 1 ACTIVE              PROF: CURR CONNS:      1
------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED
```

1 AT-TLS port 4992 is used.

Example 14-18 TLS secure connection using port 4992 - detail

```
D TCPIP,TN3270D,T,CONN,CONN=9843,DET
EZZ6065I TELNET CONNECTION DISPLAY 756
  CONNECTED: 10:50:47  09/25/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00009843   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..2537
    DESTIP..PORT: ::FFFF:10.1.1.40..4992
    LINKNAME: VIPA1L
  PORT:  4992 1 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  TTLSSECURE    ACCESS: SECURE  0A TLSV1 2
     TTLSRULE: ATTLS_TN3270D_Rule1~1 3
     TTLSGRPACTION: gAct1~ATTLS_TN3270D_4992
     TTLSENVACTION: eAct1~ATTLS_TN3270D_4992
     TTLSCONNACTION: cAct1~ATTLS_TN3270D_4992
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                        NEWENV FUNCTIONS: --
  LUNAME: SC33DT02
  APPL: SC33TS06
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                     OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                    >*DEFLUS*                  --------
    DEFLT APPL: **N/A**
    USS TABLE:  NL (NULL)
                    >USSTEST1                  P-------
    INT TABLE:  **N/A**
    PARMS:
   PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
  (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
   -------  ------------  ---  ---------  -------  ----
   *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
   -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
   -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
   *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* TP-CURR
   *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* <-FINAL 4
39 OF 39 RECORDS DISPLAYED
```

1 AT-TLS port 4992 is used.

2 The session is a secure connection. 0A (TLS_RSA_WITH_3DES_EDE_CBC_SHA) is used.

3 AT-TLS policy ATTLS_TN3270D_Rule1 is applied.

4 The security parameters defined in the AT-TLS policy are used.

Display Netstat TTLS to show connection AT-TLS information

The Display Netstat TTLS command is also helpful for obtaining information about AT-TLS, as shown in Example 14-19.

Example 14-19 Display Netstat TTLS command - AT-TLS information

```
D TCPIP,TCPIPD,N,TTLS,CONN=9843,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPD 718
CONNID: 00009843
  JOBNAME:      TN3270D
  LOCALSOCKET:  ::FFFF:10.1.1.40..4992
  REMOTESOCKET: ::FFFF:10.1.100.224..2537
  SECLEVEL:     TLS VERSION 1
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: ATTLS_TN3270D_RULE1~1
  PRIORITY:       255
  LOCALADDR:      0.0.0.0/0
  LOCALPORT:      4992
  REMOTEADDR:     0.0.0.0/0
  REMOTEPORTFROM: 1024              REMOTEPORTTO: 65535
  JOBNAME:        TN3270D
  DIRECTION:      BOTH
  TTLSGRPACTION:  GACT1~ATTLS_TN3270D_4992
    GROUPID:                    00000002
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
  TTLSENVACTION:  EACT1~ATTLS_TN3270D_4992
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    SHAREDRING1
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~ATTLS_TN3270D_4992
    HANDSHAKEROLE:              SERVER
    V3CIPHERSUITES:             0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
    APPLICATIONCONTROLLED:      ON
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

14.6 Problem determination for Telnet server security

If multiple Telnet servers are used (for example, multiple TCP/IP stacks in a Sysplex Distributor environment), ensure each server uses unique LU names. Otherwise, the second server that uses the same LU name will be unable to establish a session. Either the OPEN ACB request will fail or the cross-domain session request will fail.

If you have a problem with AT-TLS, verify that the Policy Agent is set up correctly and started. Also verify the policy is configured for a TCP/IP stack without syntax errors. Be sure Application Controlled is set On in TTLSEnvironmentAdvancedParms. The file generated by the IBM Configuration Assistant for z/OS Communication Server can be read with no syntax errors.

For further diagnosis, two traces can be used.

- AT-TLS trace

  Specify in the TTLSConnectionAction Trace parameter in flat file, or on the **Connectivity Rule** -> **Additional Settings** -> **Advanced** -> **Set Tracing, Timing, or Tuning** option in the IBM Configuration Assistant for z/OS Communication Server. The output is written to the syslog.

- TN3270 server trace

  TelnetParms Debug parameter

Refer to “Problem determination for Telnet server” in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533 and *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782 for further information.

14.7 Additional information sources for TN3270 AT-TLS support

Detailed information about the TN3270E Telnet server and protocol can be found in the following documents:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User’s Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781
- TN3270 is defined by RFC 1646
- TN3270E is defined by RFC 1647 and RFC 2355

Tip: For descriptions of security considerations that affect specific servers or components, refer to “UNIX System Services Security Considerations” in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

15

Secure File Transfer Protocol

System administrators can implement security precautions to protect data being transferred from one network device to another. Rules can be applied to client devices, server devices, applications platforms, and network firewalls to administer the security policies of the organization. This chapter focuses on the security measures that can be applied to the z/OS FTP client and server applications.

Note that the tasks, examples, and references in this chapter are based on the FTP chapter in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533, where basic, non-secure FTP scenarios are discussed. The assumption here is that you already have FTP clients and servers successfully executing in your environment.

The following topics are discussed in this chapter.

Section	Topic
15.1, “Conceptual overview of FTP security” on page 580	General security concepts for FTP
15.2, “FTP client with SOCKS proxy protocol” on page 581	How to enable a client to support an FTP SOCKS proxy server
15.3, “FTP with native TLS security support” on page 583	How to configure and test FTP’s native TLS security
15.4, “FTP with AT-TLS security support” on page 617	How to configure and test FTP’s support of AT-TLS through the policy agent
15.5, “Backing up the backing store file and policies” on page 649	Considerations for backing up your policy definitions
15.6, “Migrating from native FTP TLS to FTP AT-TLS” on page 650	Summarizes migration requirements for migrating from an existing native TLS environment to one with AT-TLS
15.7, “FTP TLS and AT-TLS problem determination” on page 652	Problem determination procedures for FTP security
15.8, “Additional information” on page 653	References to additional information about FTP security

15.1 Conceptual overview of FTP security

Security exposures can exist in the FTP environment, so it is imperative to analyze and correct them using available tools. The FTP application relies on the use of an external security manager, such as RACF, for certain levels of security. The client and server environments provide some built-in security functions for additional security. We identify a few of the common security exposures and tools that can be used to address the issues. The major emphasis is on Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Application Transparent-TLS (AT-TLS) features supported by the z/OS FTP application.

The following topics are discussed in this section:

- What is FTP security
- How FTP works
- How FTP security can be applied

15.1.1 What is FTP security

For the discussions in this book, we define the term *secure FTP* by using the following statements:

- FTP data transmissions can be secured by requiring the client to connect to the server indirectly by passing through an FTP SOCKS proxy server which can control the connections and provide an extra level of security itself on the path toward the final destination server.
- FTP clients can allow or require server authentication whereby the server identifies itself to the client by passing a digital certificate to the client who verifies the certificate against pre-established criteria. Any data exchanged between the client and server can then be encrypted using encryption keys established at connection time.
- FTP servers can allow or require the same server authentication process. Data encryption is usually the intent of this process, though not always required.
- FTP servers can require client authentication whereby the client identifies itself to the server by passing a digital certificate to the server who verifies the certificate against pre-established criteria. The certificate information can be used by the server to provide secure application logons on behalf of the client, thus avoiding the exchanges of user ID and password in clear text.

Note: This list of statements does not fully define *secure FTP*. However, because of the focus of this chapter, we have limited our definition to the statements listed.

15.1.2 How FTP security works

SOCKS proxy protocols enable an FTP client to access a remote FTP server, protected by some blocking mechanism, which is otherwise unreachable. The client does not have direct network access to the final destination server, but the intermediate (proxy) server does have the necessary access. So the client, having access to the proxy server, can connect to the proxy by using a special SOCKS protocol, pass through the proxy, and thereby gain access to the intended final destination.

Clients and servers can identify (authenticate) each other by using digital certificates, and can encrypt data exchanges by using encryption keys obtained from these certificates.

In addition to native TLS support, the FTP server and client can use AT-TLS to manage TLS security. TLS managed by AT-TLS (TLSMECHANISM ATTLS) supports more security

functions than TLS managed natively by the FTP (TLSMECHANISM FTP). Be aware of these AT-TLS capabilities and requirements when planning the preferred AT-TLS support for FTP:

- Specify the label of the certificate to be used for authentication instead of using the default.
- Support SSL session key refresh.
- Support SSL sysplex session ID caching.
- Trace decrypted SSL data for FTP in a data trace.
- Receive more detailed diagnostic messages in syslogd.
- There are no restrictions for the FTP ATTLS function.
- Policy Agent must be active for FTP ATTLS to work.
- TLS security defined natively to FTP will continue to be available in addition to AT-TLS.

15.1.3 How FTP security can be applied

FTP security can be applied by implementing restricted access to FTP servers with firewalls, and then requiring the use of a SOCKS proxy server to control client access to the intended destination server.

Digital certificates can be used for client and server authentication and to enable the use of data encryption.

If you already have TLS implemented for an existing instance of the FTP server, it is easy to migrate from TLS to AT-TLS support. AT-TLS support is implemented by moving most of the TLS definitions from the FTP server's FTP.DATA file into the policy agent's AT-TLS configuration profile section. The FTP server is defined as an AT-TLS controlling application to the policy agent, and can retain its control of the secure relationship with the client. The AT-TLS implementation provides more flexibility and functionality than basic TLS, and is therefore the preferred method of implementing digital certificate support for the FTP server. The z/OS Configuration Assistant GUI is used to create the appropriate policy agent statements for AT-TLS.

15.2 FTP client with SOCKS proxy protocol

In this section we discuss the configuration changes necessary to add SOCKS server support to the base FTP client we introduced in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

This section discusses the following topics:

- Description of the SOCKS proxy protocol
- Configuration of the SOCKS proxy protocol
- Activation and verification of the SOCKS proxy FTP
- Activation and verification of the SOCKS proxy FTP

15.2.1 Description of the SOCKS proxy protocol

An FTP client can be permitted to access an FTP server either directly, or it can be required to access the server indirectly by passing through a SOCKS proxy server to get to the FTP server. If the FTP client must pass through a SOCKS proxy, the client must use the SOCKS protocol to successfully navigate through the proxy server. The client must be able to determine which servers it must contact using the SOCKS protocol. These servers are specified in a SOCKS configuration file that the client reads to make this determination. The SOCKSCONFIGFILE statement in the client's FTP.DATA file is used to point the client to the configuration file where these servers are identified.

Dependencies

Note the following dependencies:

- The SOCKS protocol is supported for IPv4 only, not IPv6.
- The SOCKSCONFIGFILE is applicable to the client only; the server ignores the statement.
- If the SOCKSCONFIGFILE is not present in the client's FTP.DATA file, the client does not use SOCKS to contact any servers.
- The configuration file can be an HFS file or an MVS dataset.

Using the FTP client with SOCKS allows users to contact FTP servers that are protected by a SOCKS firewall, which are otherwise unreachable.

15.2.2 Configuration of SOCKS proxy protocol

To configure the client for SOCKS support, perform the following tasks:

- Configure the SOCKS configuration file.
- Add the SOCKSCONFIGFILE statement to FTP.DATA.

Configure the SOCKS configuration file

The configuration file can be an HFS file or an MVS dataset. You can code DIRECT or SOCKD statements in the configuration file. A DIRECT statement instructs the FTP client to access the FTP server without using SOCKS. A SOCKD statement directs the client to use SOCKS protocols and the specified SOCKS server to access the FTP server.

The order of statements in the SOCKS configuration is important. The client searches the statements in the order they are coded in the file. The first statement that matches the identification of the target FTP server is applied. Code statements that apply to specific FTP servers first, and a general statement for all other servers last.

Use the DIRECT statement to instruct the FTP client not to use SOCKS for the destinations that are included in the DIRECT statement.

Use the SOCKD statement to instruct the FTP client to use a SOCKS server for the destinations that are included in the SOCKD statement.

The statements in the file have the following syntax:

```
DIRECT ftp_server_ipaddr/number_of_subnetmask_bits
SOCKD @=proxy_server_ipaddr    ftp_server_ipaddr/number_of_subnetmask_bits
```

Example 15-1 shows some configuration statements.

Example 15-1 SOCKS configuration file, TCPIPB.TCPPARMS(FTPSOCKS)

```
1 SOCKD @=10.1.100.201 172.14.0.0/16
2 DIRECT 10.0.0.0/8
3 DIRECT 127.0.0.1/32
4 DIRECT 0.0.0.0 0.0.0.0
```

The following items explain the statements in Example 15-1:

1 Use SOCKS protocol to connect to the proxy server at 10.1.100.201 to pass through to any destination FTP servers in the subnet of 172.14.x.x.

2 Access all FTP servers within the 10.x.x.x network directly (no SOCKS protocol).

3 Access the local loopback address directly with no SOCKS protocol.

4 Access all other FTP servers directly that are not mentioned here.

For complete details on the use of the statement parameters, refer to the SOCKSCONFIGFILE section in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Add the SOCKSCONFIGFILE statement to FTP.DATA

The FTP client uses configuration information in the SOCKS configuration file to determine whether to access a given IPv4 FTP server directly or through a SOCKS server. The name of the SOCKS configuration file is specified by coding the SOCKSCONFIGFILE statement in the client's FTP.DATA file. Here are two examples:

```
SOCKSCONFIGFILE /etc/ftpsocks.conf
SOCKSCONFIGFILE 'TCPIPB.TCPPARMS(FTPSOCKS)'
```

15.2.3 Activation and verification of the SOCKS proxy FTP

To use SOCKS proxy protocol with your FTP client, do the following:

- Prepare the client to use the updated FTP.DATA file.
- Execute the FTP client with SOCKS enabled.
- Verify client log messages are as expected.

Prepare the client to use the updated FTP.DATA file

Allocate the FTP.DATA file to the TSO client as follows:

```
alloc ddn(sysftpd) dsn('tcpipb.tcpparms(ftpcb31)') shr
```

Allocate the FTP.DATA file to the batch job client as follows:

```
//SYSFTPD DD DSN=TCPIPB.TCPPARMS(FTPCB31),DISP=SHR
```

Execute the FTP client with SOCKS enabled

Connect to an FTP server in the protected network and the SOCKS server should be used to get there; for example:

```
FTP 172.14.1.20
```

Connect to your local loopback address and you should connect directly without SOCKS assistance; for example:

```
FTP 127.0.0.1
```

Verify client log messages are as expected

The expected client connection and login messages should be observed. If connecting to the proxy server, supply the appropriate user ID and password at the proxy server device. When connecting to the final destination FTP server, supply the user ID and password at *that* device.

15.3 FTP with native TLS security support

In this section we discuss the security parameters to be added to the basic FTP design described in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533, to enable native TLS. We need to enable various RACF profiles and then authenticate and encrypt the FTP sessions.

Both the FTP server and client in the z/OS Communications Server support TLS and Kerberos security. We show the use of TLS. FTP can also be made secure with z/OS Communications Server Application Transparent TLS (AT-TLS) or IP Security (IPSec). This section only discusses the TLS security built into FTP. AT-TLS and IPSec are discussed in detail in their respective chapters in this book.

Note: This section focuses on implementing SSL/TLS in FTP. The SSL/TLS implementation of FTP is known as secure FTP and invokes security in z/OS Communications Server using z/OS System SSL. It is based upon extensions to the base FTP RFC 959:

- RFC 2246, "The TLS Protocol Version 1"
- Internet Draft RFC, "On Securing FTP with TLS (draft 05)"
- RFC 2228, "FTP Security Extensions"
- RFC 4217, "Securing FTP with TLS"

"Secure FTP" should be distinguished from the OpenShell procedure known as "sftp". Unlike secure FTP in z/OS Communications Server, sftp operates over an encrypted secured shell (ssh) transport and does not use FTP protocols as described in RFC959 and its related RFCs.

The following topics are discussed in this section:

- Description of FTP native TLS security
- Configuration of FTP native TLS security
- Activation and verification of the FTP server without security
- Activation and verification of the FTP server with TLS security: Internet Draft protocols
- Activation and verification of the FTP server with TLS security: RFC4217 protocols

15.3.1 Description of FTP native TLS security

We now expand on our base FTP server (introduced in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533), and add additional security options. We set up the secure FTP server on one LPAR, and use a third-party FTP client and a z/OS client to demonstrate the secure FTP session.

Implicit versus explicit TLS connections

An *implicit* TLS connection is one in which the client connects to a nonstandard port (port 990) and immediately performs TLS negotiation. An *explicit* TLS connection is one where the client connects to the standard FTP server port (21) and issues a command (AUTH) indicating that TLS should be enabled. In our implementation of secure FTP we demonstrate the use of explicit TLS.

Dependencies

There are multiple dependencies, beginning with an appropriate understanding of the type of security that is acceptable to a particular business site, and ending with software and hardware prerequisites.

With TLS, SSLv2 is not supported.

Basic security audit information

Is TLS the appropriate security mechanism, or must another security mechanism be considered? The response to this question depends on the types and numbers of clients and servers, the types of file transfers that need to be invoked, the types of encryption algorithms

that need to be used, and so on. Many of these issues are discussed in standard product publications and in Chapter 3, “Certificate management in z/OS” on page 31.

When creating the certificates for the FTP server or the FTP client, the same considerations apply that apply to native TLS support for FTP:

- Does a shared key ring provide enough security for the environment?
- Does a shared site certificate for the server suffice, or is the granularity of individual server certificates more suitable?
- Is there any requirement for client certificates? If so, must these be available to individual clients or can a group of clients share a certificate for authentication purposes?
- How many and what type of certificates are necessary? How is certificate management to be handled? Is it important to engage a well known Certificate Authority to produce and manage certificates. Can the installation be its own Certificate Authority?

Software and hardware prerequisites

If FTP TLS is the appropriate security mechanism for an installation, an SAF environment must be available for necessary security authorizations. An FTP server that is already functioning in a non-secure mode should be available as a basis for the new security work with FTP. A decision should have been made about the type of hardware assists or cryptographic adapters necessary to ensure that service level agreements on performance can be met.

In addition to the FTP.DATA statements required for the basic FTP setup, FTP with security requires a key ring database with at least one certificate and additional statements in FTP.DATA. The server certificate for FTP must be the default certificate on the key ring.

FTP with security provides a safer environment for the transmission of data. It provides user authentication, encryption, and data integrity checking.

Considerations

The use of security requires more configuration and requires management of key rings. The use of TLS also adds overhead to the FTP transfer and requires an FTP client that supports TLS.

Important: Native TLS with FTP relies on the existence of a default certificate in a key ring. It cannot retrieve specific certificates by certificate label.

AT-TLS supports certificate labels, or can continue to use a default certificate in the absence of a certificate label specification.

FTP with TLS supports two different TLS standards that are built on top of RFC 2246, “The TLS Protocol Version 1” and RFC 2228, “FTP Security Extensions”.

- The DRAFT protocols defined with Internet Draft RFC, “On Securing FTP with TLS (draft 05)”
- The RFC 4217 protocols defined with RFC 4217, “Securing FTP with TLS“

RFC 4217 removes some of the restrictions that the Internet DRAFT imposes on FTP negotiations. In 15.3, “FTP with native TLS security support” on page 583, we test with both the DRAFT version of TLS and the RFC 4217 version of TLS.

15.3.2 Configuration of FTP native TLS security

To start FTP with TLS, we need to first perform all the implementation tasks for both client and server described in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

We create a server certificate, which might be a self-signed server certificate, a personal server, or site certificate that has been signed by a Certificate Authority. The client needs access to a key ring on the client's system, and may optionally even employ a client certificate for client authentication.

We need one or more key rings into which we can store the certificates for both client and server. Then we make changes to the FTP.DATA file for the server and optionally to the z/OS client. Finally, we need to **permit** the user ID of the z/OS FTP server and the z/OS FTP client to the appropriate RACF classes.

We tested two scenarios with FTP and native TLS security:

- z/OS client to z/OS server
  - We verified that the z/OS FTP client on TCPIPB still worked as expected when TLS was enabled at the server but not invoked by the client.
  - We verified that the client operated as desired when TLS was enabled and invoked.
    - We tested with both TLSRFCLEVEL DRAFT and TLSRFCLEVEL 4217.
- Workstation client to z/OS server
  - We verified that the workstation client still worked as expected when TLS was enabled at the server but not invoked by the client.
  - We verified that the client operated as desired when TLS was enabled and invoked.
    - We tested with both TLSRFCLEVEL DRAFT and TLSRFCLEVEL 4217.

To set up FTP server TLS support, perform the following tasks:

- Create key ring and certificates and update RACF permissions
- Update the FTP.DATA for the server

Create key ring and certificates and update RACF permissions

We strongly encourage using a digital certificate that is signed by a professional certificate authority (CA). However, to demonstrate the security features of FTP we created a self-signed CA certificate and then used it to sign our shared SITE certificate.

This approach enabled us to be our own, internal mju78iklCA. The commands may be executed from ISPF option 6, but because we wanted to keep a record of our security commands, we preferred to submit them from a batch job. Refer to Chapter 3, "Certificate management in z/OS" on page 31 for the details on setting up a shared ring, an internal CA certificate, and a shared SITE certificate with batch jobs.

Our shared SITE certificate is to be used by several servers and clients, including our FTP server. The FTP server is associated with the OMVS segment and user ID of TCPIP. In addition, we want the shared key ring on which the self-signed CA certificate and the server SITE certificate reside to be owned by the same user ID, TCPIP. Here are the RACF tasks we performed to set up the key ring and certificates.

- Create a shared key ring
- Generate a RACF CA certificate
- Generate a RACF SITE certificate
- Connect both certificates to the shared key ring

- Permit access to the private key of the shared SITE certificate in the key ring
- Permit access to LIST (retrieve) the certificates
- Permit access to the shared key ring
- Export the CA certificate to a flat file for prepopulating client devices
- Summary of shared key ring environment for FTP

Create a shared keyring

Certain RACF classes must be active before rings and certificates are added with the RACDCERT command.

```
SETROPTS CLASSACT(DIGTCERT)
SETROPTS CLASSACT(DIGTRING)
```

Create the shared key ring, refresh the relevant classes, and list the rings associated with the user ID of TCPIP.

```
RACDCERT ID(TCPIP) ADDRING(SharedRing1)
SETROPTS RACLIST(DIGTCERT) REFRESH
SETROPTS RACLIST(DIGTRING) REFRESH
RACDCERT LISTRING(*) ID(TCPIP)
```

Generate a RACF CA certificate

The default for the life of a certificate is one year. We executed the first RACDCERT command to assign an expiration date of September 11, 2008. You may wish to change the time frame in which the certificate is to be valid. After the certificate creation, refresh the facility and list all the generated CA certificates.

```
RACDCDERT CERTAUTH GENCERT -
   SUBJECTSDN( O('IBM Corporation') OU('ITSO Certificate Authority') C('US')) -
   NOTBEFORE(DATE(2007-09-11)) NOTAFTER(DATE(2008-09-11)) -
   KEYUSAGE(CERTSIGN) WITHLABEL('CS19 ITSO CA1')
SETROPTS RACLIST(FACILITY) REFRESH
RACDCERT CERTAUTH LIST
```

Generate a RACF SITE certificate

A USER certificate is associated with a user ID and may be used only by a single process: server process or client process. A SITE certificate is also associated with a user ID, but it may be shared among several users: either servers, or clients, or both. Some may consider a shared SITE certificate a security exposure because it is not distinctly associated with a single process; others may not consider this a problem.

Note: If your enterprise needs more stringent security controls than a shared site certificate offers, we urge you to use separate USER certificates for each server and client. You should create an environment that meets your organization's requirements.

This is the set of RACF commands we executed in order to create a shared SITE certificate for our implementation of FTP. Notice the SITE certificate is signed with the CA certificate just created.

```
RACDCERT SITE GENCERT -
   SUBJECTSDN(CN('ITSO.IBM.COM') OU('ITSO CS19 Shared Site') C('US')) -
   WITHLABEL('CS19 ITSO SharedSite1') -
   SIGNWITH(CERTAUTH LABEL('CS19 ITSO CA1')
SETROPTS RACLIST(FACILITY) REFRESH
RACDCERT SITE LIST
```

Connect both certificates to the shared key ring

Execute the RACF commands to connect the two certificates to the shared key ring.

```
RACDCERT ID(TCPIP) -
   CONNECT(CERTAUTH LABEL('CS19 ITSO CA1') RING(SharedRing1) USAGE(CERTAUTH))
RACDCERT ID(TCPIP) -
   CONNECT(SITE LABEL('CS19 ITSO SharedSite1') RING(SharedRing1) USAGE(PERSONAL) -
   DEFAULT)
SETROPTS RACLIST(DIGTCERT) REFRESH
SETROPTS RACLIST(DIGTRING) REFRESH
RACDCERT LISTRING(*) ID(TCPIP)
```

Important: The USAGE parameter for a SITE certificate must specify USAGE(PERSONAL).

The default is USAGE(SITE) and renders the certificate useless.

Permit access to the private key of the shared SITE certificate in the key ring

Every user, whether server or client, requires access to the private key of the shared SITE certificate. These commands provide that access to the relevant users. First we permit the user ID associated with the server and client procedures (TCPIP). Then we permit access by the general users, CS01 and CS02, who can FTP into the FTP server.

```
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(TCPIP) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS01) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS02) ACCESS(CONTROL)
SETROPTS RACLIST(FACILITY) REFRESH
SETROPTS RACLIST(DIGTCERT) REFRESH
```

Permit access to LIST (retrieve) the certificates

To access a certificate authority (CA) or site certificate, the user (server or client) must have control access to the IRR.DIGTCERT.LIST resource in the FACILITY class.

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(TCPIP) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(CS01) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(CS02) ACCESS(CONTROL)
SETROPTS RACLIST(FACILITY) REFRESH
SETROPTS RACLIST(DIGTCERT) REFRESH
```

Permit access to the shared key ring

The shared key ring in our environment is the repository for the CA certificate and the server SITE certificate. FTP clients also need a key ring in order to maintain Certificate Authority certificates and to receive server certificates they are sent at server connect time.

In our sample scenarios, we are using the same shared key ring for z/OS clients and z/OS servers. Therefore, both groups of participants in the FTP protocols need access to the shared ring.

Owners of a key ring need READ access to it. Non-owners of a key ring need UPDATE access to it. The FTP procedure is owned by the user ID TCPIP. z/OS TSO or UNIX FTP clients are associated with different user IDs, like CS01 and CS02. Therefore, we executed these commands to provide the appropriate authorizations to the FTP server.

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TCPIP) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS01)  ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(CS02)  ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
SETROPTS RACLIST(DIGTCERT) REFRESH
```

Export the CA certificate to a flat file for prepopulating client devices

Most vendors of client software (TN3270, FTP, and so on) usually prepopulate their client key rings with well-known Certificate Authority certificates. However, the CA certificate that we are using to sign all of our other SITE certificates and server certificates is a self-signed CA certificate, and client devices will not initially have a copy of it. We must EXPORT it from RACF to an MVS flat file in preparation for sending it to the clients. This is usually accomplished using FTP.

Note: To avoid this step, consider having a well-known Certificate Authority sign your SITE and Server certificates. You are not responsible for acquiring or sending copies of their CA certificates to your clients. The client devices and their software should already have copies of these public CA certificates. If not, they have the responsibility of acquiring them directly from the Certificate Authority.

RACF provides a number of different formats in which to export the certificate and its keys. The format chosen will be dictated by how the client and server use the certificate and the requirements of the level of TLS used by the negotiation process. Refer to *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683 and *z/OS Security Server RACF Command Language Reference*, SA22-7687, for explanations of the available formats and why each would be used. We used the following export command:

```
RACDCERT CERTAUTH EXPORT(LABEL('CS19 ITSO CA1')) -
    FORMAT(CERTB64) DSN('TCPIP.CS19.CACERT')
```

For our scenario, we chose format CERTB64 to export our CA certificate as a single certificate with only the public key included. Clients that require a copy of our certificate to be prepopulated into their key ring can import this copy and mark it as a trusted CA certificate. Using this format, we do not compromise the integrity of this certificate's private key.

Summary of shared key ring environment for FTP

Figure 15-1 depicts how the z/OS FTP client and the z/OS FTP server will use the certificates we just created in the shared RACF key ring.

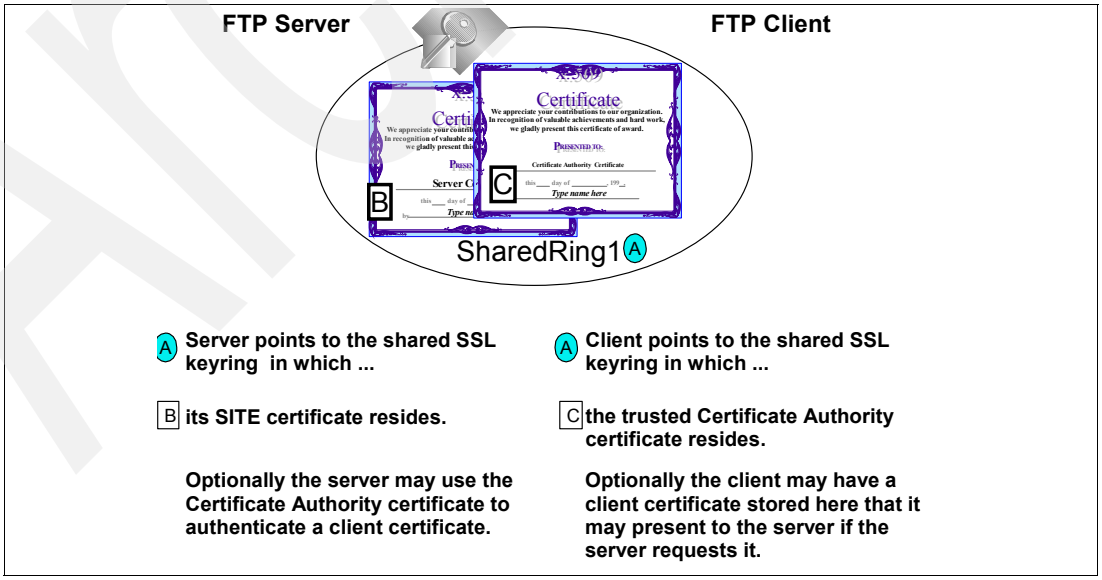


Figure 15-1 The shared key ring for FTP client and server

The server accesses the SITE certificate and any CA certificates that were involved in the signing of the SITE certificate when it had to identify itself to the client during the SSL negotiation process. A CA might use a root certificate and one or more intermediate certificates to sign another certificate. Likewise, the client needs access to the CA certificate (one or more) that has signed the FTP server's SITE certificate.

Optionally, the server may request client authentication. In this case the client will send its certificate signed by a trusted certificate authority to the server. The client certificate may be stored in the same key ring as that of the CA certificate.

The shared key ring is only available on nodes that share the same RACF database. If our client is a workstation, then there is no shared key ring. Instead, the workstation has a key ring separate from that of the z/OS server node.

Update the FTP.DATA for the server

We added a number of statements to the server FTP.DATA to provide for additional security. Most of the statements are documented in "Security options" of the hlq.SEZAINST(FTSDATA) member. Others are documented in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Example 15-2 shows the additional statements we added to our FTP.DATA data set to enable security.

Example 15-2 Additional statements in FTP.DATA to enable TLS security

```
1 EXTENSIONS   AUTH_TLS
2 TLSMECHANISM FTP
3 KEYRING      TCPIP/SharedRing1
4 CIPHERSUITE       SSL_RC4_MD5_EX     ; 03
4 CIPHERSUITE       SSL_RC4_MD5        ; 04
4 CIPHERSUITE       SSL_3DES_SHA       ; 0A
4 CIPHERSUITE       SSL_DES_SHA        ; 09
4 CIPHERSUITE       SSL_NULL_MD5       ; 01
4 CIPHERSUITE       SSL_NULL_SHA       ; 02
5 SECURE_FTP   ALLOWED
6 SECURE_CTRLCONN   PRIVATE
7 SECURE_DATACONN CLEAR
8 TLSTIMEOUT        100
9 TLSRFCLEVEL       DRAFT
```

1 Enable TLS with an EXTENSIONS AUTH_TLS statement.

2 Indicate native FTP TLS support (not AT-TLS)

Note: With FTP TLS, SSLv2 is not supported.

3 Add a KEYRING statement to identify the owner and name of the key ring file. By default, the owner ID is the user ID associated with the FTP server started task. In our case, the key ring is owned by the user ID TCPIP; technically it is not necessary to place the user ID in front of the label name of the key ring. However, it is good practice to do so in case other FTP server procedures not owned by TCPIP use this FTP.DATA file as a model.

4 Add a CIPHERSUITE statement for each encryption algorithm desired. The order in which they are coded here determines the order (priority) in which they are negotiated with the client.

5 Add a SECURE_FTP ALLOWED statement to allow for, but not require, TLS clients. In this fashion, the same FTP port can be used for non-TLS connections or for TLS connections, depending on the nature of the client request. If your implementation requires the use of TLS for every connection (not optional), then code SECURE_FTP REQUIRED instead.

6 Add a SECURE_CTRLCONN PRIVATE statement. This statement is ignored unless the client requests a secure connection. We recommend coding SECURE_CTRLCONN PRIVATE if the TLS mechanism is enabled, because the control connection carries user IDs and passwords. It is common practice to protect the user ID and the password even if the data connection need not be secure.

7 Add a SECURE_DATACONN CLEAR statement to allow the client to determine whether data traffic really needs encryption. Encryption carries a performance price and may not be necessary for every data transfer.

8 Take the default of 100 on TLSTIMEOUT or code it. This specifies the maximum time between full TLS handshakes.

9 Add or default TLSRFCLEVEL DRAFT. This indicates that the FTP TLS protocol is following the draft RFC named “On Securing FTP with TLS - revision 05”. It explains how to use RFC 2228 commands to implement TLS security.

15.3.3 Activation and verification of FTP server without security

In this section we enable the FTP server for TLS security, but the client does not request security. We do this in order to verify that the chosen clients still work as expected with the revised FTP server.

The scenario we depict first is illustrated in Figure 15-2.

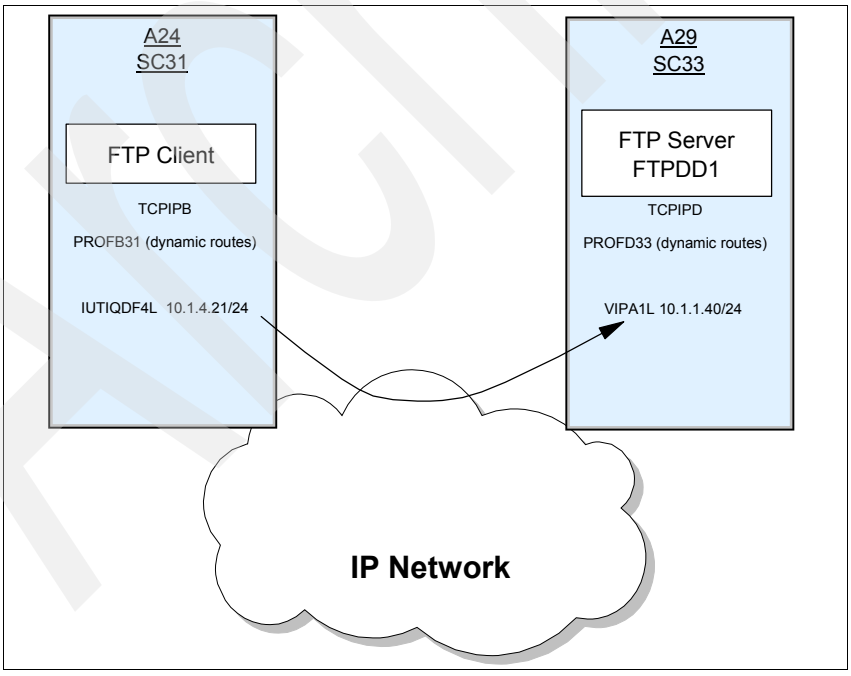


Figure 15-2 FTP Client connection to server: no security

There are a number of ways to verify that the FTP server has started and is working correctly.

Perform the following tasks:

- Check FTP server job log messages
- Verify the non-TLS z/OS client can connect to the server
- Display system job tasks for control and data connections
- Use a workstation client to verify the FTP server

Check FTP server job log messages

First, we stop the current FTP server at TCPIPD and restart it while pointing to the correct FTP.DATA file that has been reconfigured for TLS:

- `p ftpdd1`
- `s ftpdd`

Our FTP procedure is copied after 'hlq.TCPPARMS(FTPSPROC)' and begins like Example 15-3.

Example 15-3 FTP procedure without _BPX_JOBNAME specified

```
//FTPDD  PROC MODULE='FTPD',PARMS='',
//        TCPDATA=DATAD&SYSCLONE.,FTPDATA=FTPSTLS
//FTPD   EXEC PGM=&MODULE,REGION=0M,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPD"',
//            '"TZ=EST5EDT4")/&PARMS')
//*************************************************************
```

Look for message EZY2702I on the system console. Example 15-4 shows the EZY2702I message we received shortly after FTP was started.

Example 15-4 EZY2702I message received on successful startup of FTP

```
EZY2702I Server-FTP: Initialization completed at 10:13:32 on 09/20/07.
```

Example 15-5 shows that the FTP server has initialized in an address space named FTPDD1 and that it is owned by user ID TCPIP.

Example 15-5 D OMVS,A=ALL: FTPDD1 as a Unix Address Space

```
D OMVS,A=ALL
 BPXO040I 10.13.42 DISPLAY OMVS 799
 OMVS     000F ACTIVE             OMVS=(9A)
 USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS

TCPIP    FTPDD1   0060   17039371          1 1FI--- 10.13.32      .022
  LATCHWAITPID=         0 CMD=FTPD
```

Verify the non-TLS z/OS client can connect to the server

To verify that the FTP server was functional, we used an FTP client on another TCPIP, TCPIPB, that was not TLS-enabled. Our FTP server at TCPIPD was set up with TLS as optional (SECURE_FTP ALLOWED).

Example 15-6 on page 593 shows the output of our FTP client on TCPIPB after we connected to our FTP server. We did not invoke TLS in the client FTP.DATA file, and we did not invoke TLS with a parameter on the ftp command itself. Therefore, the connection we tested was not running with TLS.

From ISPF, Option 6, we entered the client command at TCPIPB on LPAR A24. TCPIPB is running in a CINET environment; therefore we needed to designate appropriate stack affinity for the FTP client:

- ftp 10.1.1.40 (TCP TCPIPB)
- The resulting output, prior to signing on with the user ID and password, is shown in Example 15-6.

Example 15-6 FTP from one z/OS system to another z/OS system

```
FTP: using TCPIPB
Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS at WTSC40.ITSO.IBM.COM, 10:55:49 on 2007-09-20.
220 Connection will close if idle for more than 5 minutes.
NAME (10.1.1.40:CS02):
```

Display system job tasks for control and data connections

After we login, but prior to entering the user ID and password, we see output at TCPIPD (Example 15-7) from the D OMVS,A=ALL command that shows that we have address space #7B, which has been forked off to represent the client control connection to the server. At this point the client connection in address space (ASID of 7B) is still running under the security context of TCPIP, because the user ID and password have not been entered. The JOBNAME of the client name has been forked off as FTPDD2.

Example 15-7 OMVS address spaces and connection IDs for FTP control connections

```
D OMVS,A=ALL
BPXO040I 10.58.15 DISPLAY OMVS 952
OMVS     000F ACTIVE              OMVS=(9A)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
......
TCPIP    FTPDD1   0060   33816587          1 1FI--- 10.53.31      .021
   LATCHWAITPID=         0 CMD=FTPD TCPIP
.....
TCPIP    FTPDD2   007B   17039439   33816587 1FI--- 10.55.49      .013
  LATCHWAITPID=         0 CMD=FTPD


D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS      TCPIPD 943
USER ID  CONN     STATE
FTPDD1   00007EC6 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00007ED2 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1034
```

At TCPIPD in Example 15-7 we also see for this connection the output of the D TCPIP,TCPIPD,N,CONN netstat command. Jobname FTPDD1 shows two connections: one connection on port 21 in a listen state that represents the listener task, and one established connection between local port 21 and 10.1.4.21 that represents the established connection for our client.

At the client, TCPIPB, we next enter the user ID and password for our client user; see Example 15-8 on page 594.

Example 15-8 Entering user ID and password after initial connection

```
FTP: using TCPIPB
Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS at WTSC40.ITSO.IBM.COM, 10:55:49 on 2007-09-20.
220 Connection will close if idle for more than 5 minutes.
NAME (10.1.1.40:CS02):
cs02 >>> USER cs02
331 Send password please.
PASSWORD:
>>> PASS
230 CS02 is logged on.  Working directory is "CS02.".
Command:
```

Example 15-9 shows the output after our user, CS02, has initially established a connection and entered a user ID and password.

Example 15-9 Change to security context of control connection

```
D OMVS,A=ALL
BPXO040I 10.58.15 DISPLAY OMVS 952
OMVS     000F ACTIVE             OMVS=(9A)
USER     JOBNAME  ASID       PID       PPID STATE   START     CT_SECS
TCPIP    FTPDD1   0060   33816587          1 1FI--- 10.53.31      .030

  LATCHWAITPID=         0 CMD=FTPD

CS02a    CS02b    007B   17039439   33816587 1FI--- 10.55.49      .058
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2136795344
```

Notice, in Example 15-9, that address space 7B is now running under the security context of CS02 (a) and no longer under that of the superuser TCPIP. The JOBNAME has also changed to that of the user: CS02 (b). (In this version of the FTP server JCL, we did not specify _BPX_JOBNAME as an environment variable, as you see in Example 15-3 on page 592.)

Our user, CS02, enters the client subcommand **stat** to verify the parameters under which the FTP server is operating. The FTP server is using the draft level of the FTP code; see a in Example 15-10.

Example 15-10 Client command, STAT, shows operating parameters of server

```
>>> STAT
211-Server FTP talking to host ::ffff:10.1.4.21, port 1034
211-User: CS02  Working directory: CS02.
211-The control connection has transferred 2577 bytes
211-There is no current data connection.
211-The next data connection will be actively opened
....
211-TLS security is supported at the DRAFT level a
```

Example 15-11 shows output at TCPIPD from the D TCPIP,TCPIPD,N,CONN command after we logged onto the FTP server from TCPIPB. Jobname FTPDD1 shows two connections: one connection on port 21 in a listen state that represents the listener task, and one established connection between local port 21 and 10.1.2.21 that represents the established connection for our client.

Example 15-11 Control port connections: listening socket and client connection

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS      TCPIPD 344
```

```
USER ID  CONN     STATE
FTPDD1   00007EC6 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00007ED2 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.2.21..1026
```

Next, we want the server to open the data connection between client and server. The FTP client at TCPIPB enters a **dir** command to request a directory listing. This creates a forked task for the client at TCPIPD, which is visible after a connection display at TCPIPD is executed; see Example 15-12.

Example 15-12 FTP data connection opened

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS      TCPIPD 183
USER ID  CONN     STATE
CSO2     00007ED8 FINWT2
  LOCAL SOCKET:   ::FFFF:10.1.1.40..20
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1039
FTPDD1   00007ED2 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1038
FTPDD1   00007EC6 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
```

FTP started with _BPX_JOBNAME

We want to illustrate how the displays would change if we started the FTP procedure with the Language Environment variable of _BPX_JOBNAME; see the JCL **a** in Example 15-13.

Example 15-13 _BPX_JOBNAME Environment Variable setting

```
//FTPDD  PROC MODULE='FTPD',PARMS='',
//        TCPDATA=DATAD&SYSCLONE.,FTPDATA=FTPSTLS
//FTPD   EXEC PGM=&MODULE,REGION=0M,TIME=NOLIMIT,
//        PARM=('POSIX(ON) ALL31(ON)',
//            'ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPD"',
//            '"_BPX_JOBNAME=FTPDDDD"', a
//            '"TZ=EST5EDT4")/&PARMS')
//*   PARM=('POSIX(ON) ALL31(ON)',
```

When our user at TCPIPB connects to the FTP server at TCPIPD, but before signing in, the display of the address space yields the output in Example 15-14.

Example 15-14 Output from DA when FTP Server is started with _BPX_JOBNAME

```
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     FTPDD1   STEP1             STC03458 TCPIP      LO  FF  729   0.00   0.02
b FTPDDDD  STEP1             STC03772 TCPIP c IN  C1 1170   0.00   0.00
```

You see at **b** in Example 15-14 that the user address space has a jobname of FTPDDD and it is still associated with the security context of user ID TCPIP **c**. The **netstat conn** displays the listening connection and the established connection to the server at FTPDD1 in Example 15-15.

Example 15-15 FTP listening and established socket connections

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS      TCPIPD 054
```

```
USER ID  CONN     STATE
FTPDD1   0000B061 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   0000B0A9 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1026
```

Our user at TCPIPB logs in, at which point the security context of the user connection changes from TCPIP to the user's ID of CS02. The jobname of the user's address space remains FTPDDD; see **d** in Example 15-16.

Example 15-16 DA output: change of security context when user logs in

```
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     FTPDD1   STEP1             STC03458 TCPIP      LO  FF  729   0.00   0.00
     FTPDDDD  STEP1             STC03772 CS02  d LO  FF 1334   0.00   0.00
```

The output from the **netstat conn** display when the FTP server opens a data connection at port 20 for the user shows that the user's address space for the data connection remains the constant name of FTPDDDD; see **e** in Example 15-17.

Example 15-17 Data connection for user: address space name

```
USER ID  CONN     STATE
FTPDDDD  0000B0E2 FINWT2 e
  LOCAL SOCKET:   ::FFFF:10.1.1.40..20
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1028
FTPDD1   0000B0DC ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1027
FTPDD1   0000B061 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
```

Many installations find it useful to keep the name of the user address spaces constant, because it helps with job accounting routines and with WLM service class assignments. For TLS jobs, this has special significance. Without _BPX_JOBNAME, the data connection for the TLS user has a constantly changing address space name that starts with the procedure name. The name would be, for example, FTPDD2 or FTPDD3, and so on. With _BPX_JOBNAME, the data connection for the TLS user remains the fixed jobname of FTPDDDD. In either case the security context reveals who the actual user is (CS02, in our examples). Figure 15-3 on page 597 lays out in tabular format the variations you can see using the various startup procedure methods.

FTP Procedure Startup Scenario	***Type of FTP Connection***	***Name of User Address Space for Client Data Connection***	***Example: FTP procedure name is ... Userid is ... Address Space name is ...***
No _BPX_JOBNAME in JCL	non-secure	Client userid	FTP Procedure: FTPDD/FTPDD1 Userid: CS02; Address Space: CS02
"	secure (TLS or AT-TLS)	Server procedure name with suffix	FTP Procedure: FTPDD/FTPDD1 Userid: CS02 Address Space: FTPDD2, etc.
_BPX_JOBNAME environment variable in JCL	non-secure	Fixed name designated with _BPX_JOBNAME	FTP Procedure: FTPDD/FTPDD1 Userid: CS02 Address Space: FTPDDDD
"	secure (TLS or AT-TLS)	Fixed name designated with _BPX_JOBNAME	FTP Procedure: FTPDD/FTPDD1 Userid: CS02 Address Space: FTPDDDD

Figure 15-3 Derivation of FTP user address space name

Use a workstation client to verify the FTP server

We now know that the z/OS client at TCPIPB can perform a successful FTP connection to TCPIPD. We need to verify that the non-SSL workstation FTP client can also connect to TCPIPD even though we have enabled certain security parameters in the FTP.DATA file of procedure FTPDD.

We tested with the FTP client on a Windows workstation and had no difficulties connecting to FTPDD at TCPIPD. However, we also needed to test standard FTP with the GUI workstation client that we planned to use for testing TLS in a later step. We chose to use Filezilla, an open source TLS-enabled FTP client that is available at:

`http://filezilla.sourceforge.net`

Our scenario is depicted in Figure 15-4 on page 598.

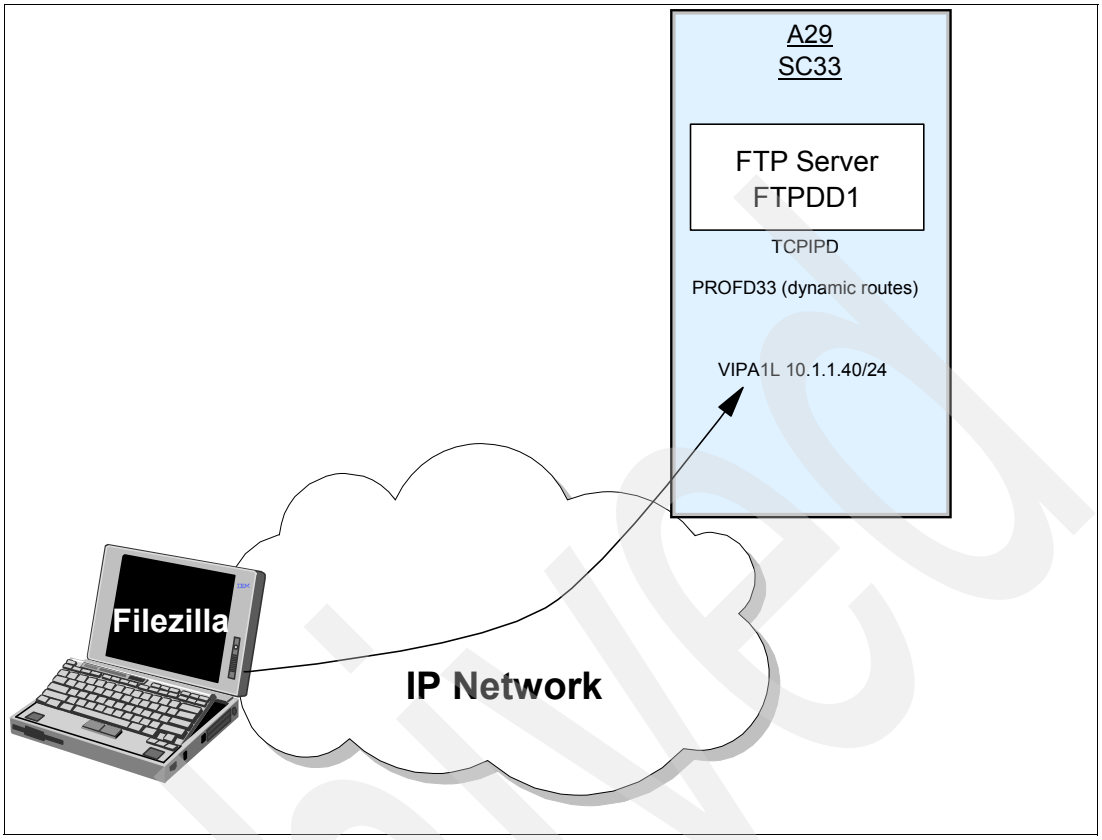


Figure 15-4 Workstation client ftps to z/OS FTP: no security

Filezilla can be configured for secure or non-secure FTP. After installing the package on our workstation, we used the file pull-down for Filezilla to enter the Site Manager panel. We then created a site for the standard FTP on this panel; see Figure 15-5.

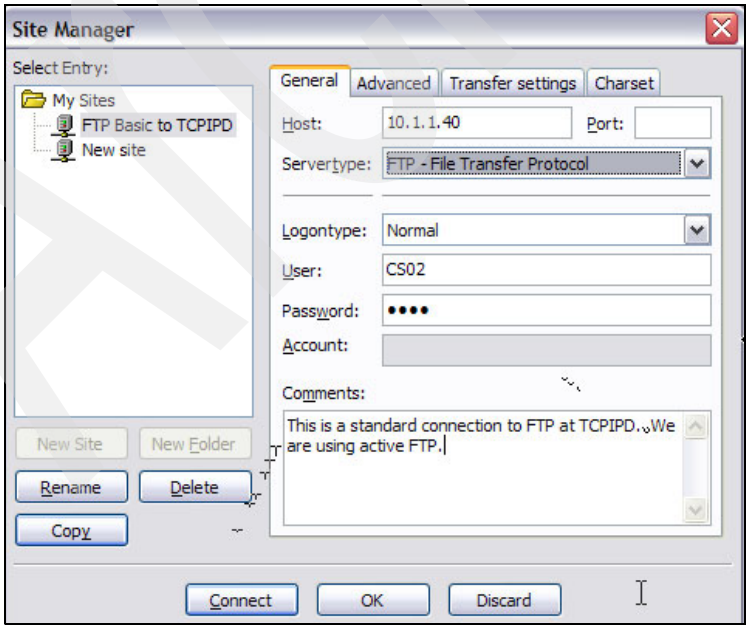


Figure 15-5 Filezilla: Basic FTP client configuration

We clicked **Connect** on the Filezilla panel and opened both a control and data connection with the FTPDD server at TCPIPD. The results are shown in Figure 15-6.

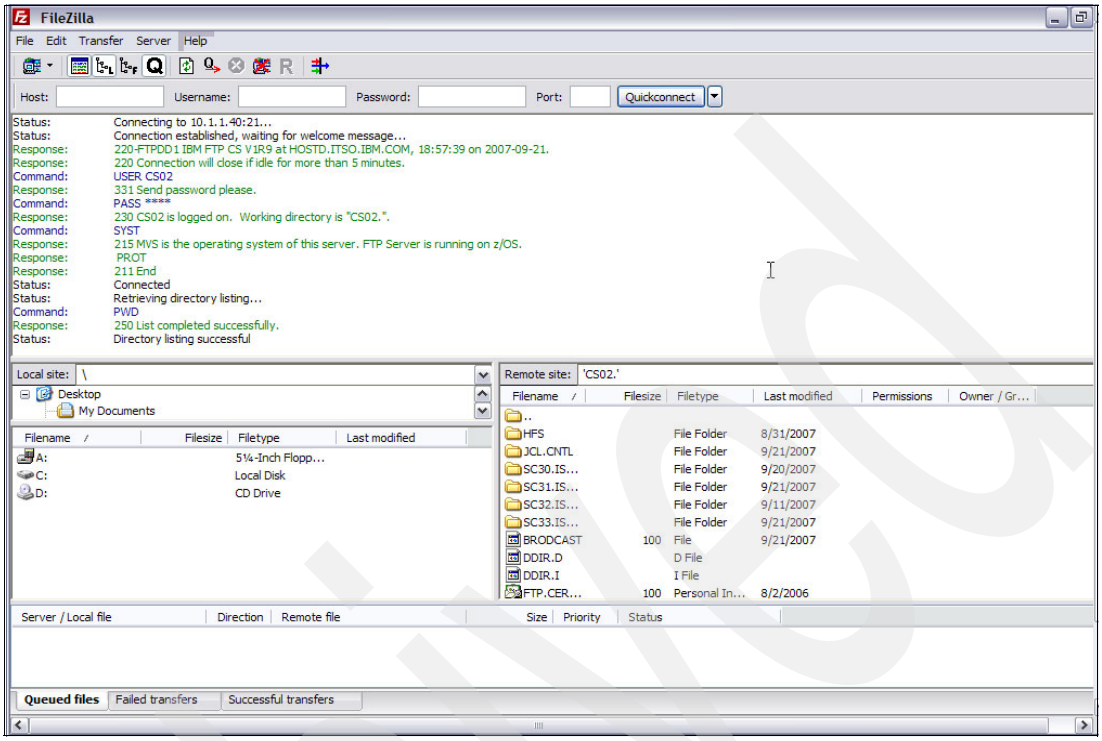


Figure 15-6 Successful FTP from Filezilla into TCPIPD

15.3.4 Activation and verification of the FTP server with TLS security: Internet Draft protocols

In this section we enable the FTP server for TLS security and we allow the client to request secure connections through the command line options. The Internet Draft RFC, “On Securing FTP with TLS (draft 05)”, represents the original TLS version introduced with the z/OS Communications Server FTP server and client. The eleven revisions of the draft since revision 05 added many enhancements that are included in RFC 4217. For now we test TLS at the TLS Internet DRAFT level. We test FTP TLS at the RFC 4217 level later in this chapter.

Assume that the tasks listed here have already been performed as described in “Create key ring and certificates and update RACF permissions” on page 586 and “Update the FTP.DATA for the server” on page 590. And use the same server startup verification process as described previously in 15.3.3, “Activation and verification of FTP server without security” on page 591:

- “Check FTP server job log messages” on page 592
- “Verify the non-TLS z/OS client can connect to the server” on page 592
- “Display system job tasks for control and data connections” on page 593
- “Use a TLS-enabled z/OS client to verify server TLS (Internet Draft)” on page 599
- “Use a TLS-enabled workstation client to verify server TLS (Draft)” on page 605

Use a TLS-enabled z/OS client to verify server TLS (Internet Draft)

We now allow the clients to request secured communication with the server FTPDD, which is depicted in Figure 15-7 on page 600.

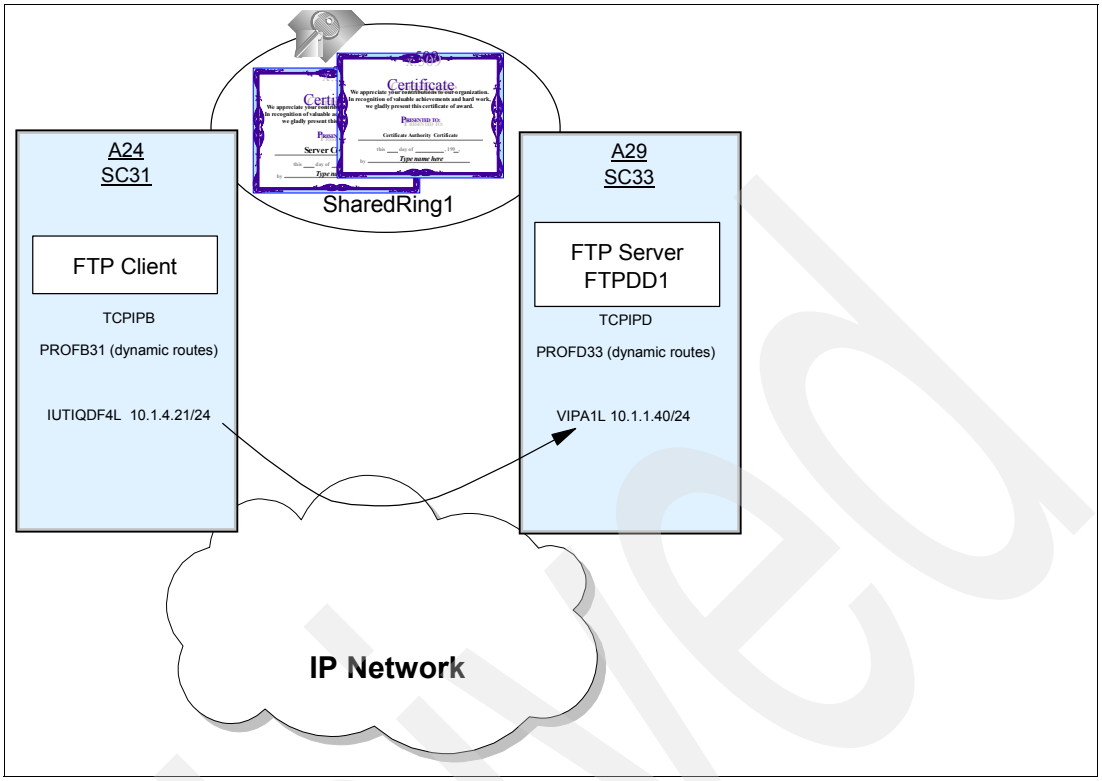


Figure 15-7 z/OS Client to FTP server: TLS Security

From ISPF, Option 6, we entered the client command at TCPIPB on LPAR A24. TCPIPB is running in a CINET environment; therefore we needed to establish stack affinity to the TCPIPB stack. In addition, we used the -a parameter to request a TLS connection:

```
ftp 10.1.1.40 -a TLS (TCP TCPIPB
```

Through the -a parameter, the FTP client attempts to authenticate to the FTP server by sending the AUTH command. The -a parameter indicates that TLS is optional for the client, implying that if the server rejects the AUTH request for TLS, the client will proceed with a normal non-TLS connection process.

Example 15-18 FTP from one z/OS system to another z/OS system requesting TLS

```
ftp 10.1.1.40 -a TLS (TCP TCPIPB
FTP: using TCPIPB
Connecting to:   10.1.1.40 port: 21.
recv error from getNextReply - EDC8128I Connection refused. (errno2=0x769F044
2)
Connection with 10.1.1.40 terminated
Command:
```

We examined the FTPDD log in syslogd, but there was no information about why the connection had failed. We also examined the syslog.log at the client side to understand why the connection failed. Again, there was no information there. We decided to turn on client logging and tracing at the client side and also enable tracing at the server side.

At the server side we issued a **modify** command for debugging at the system console; see a in Example 15-19 on page 601.

Example 15-19 Enabling tracing at the FTP server

```
F FTPDD1,DEBUG=ALL a
+EZYFT82I ACTIVE SERVER TRACES - FLO CMD PAR INT ACC UTL SEC FSC(1)
SOC(1) JES SQL
```

At the client side we entered the **trace** subcommand b and re-issued the request for a TLS connection:

```
ftp 10.1.1.40 -a TLS (TRACE TCP TCPIPB b
```

We could also have specified the -d parameter instead of **(trace** to produce client tracing. The resulting trace activity at the client as displayed in Example 15-20 helped show what was wrong.

Example 15-20 Enabling tracing at the client

```
EP1062 read_ftpdata: entered
 EP0457 init_config_defaults: entered c
 Using FTP configuration defaults. c
 FTP: using TCPIPB
 Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
 CY1337 ftpStart: client operating in dual mode
 CZ0267 ftpOpen: entered
 SC0453 initConnection: entered
 SC0898 initIPv4Connection: entered
 CY3091 access_via_socks_server: entered
 Connecting to:   10.1.1.40 port: 21.
 220-FTPDD1 IBM FTP CS      at HOSTD.ITSO.IBM.COM, 15:51:02 on 2007-09-24.
 220 Connection will close if idle for more than 5 minutes.
 FC0284 ftpAuth: no keyring d
 Authentication negotiation failed
 GV0534 seq_stat_file(2): lrecl=0 recfm=0 blksize=0 mode=1
 CZ1466 rnetrc:(3) file CS02.NETRC does not exist
 NAME (10.1.1.40:CS02):
```

In Example 15-20, at the two lines marked with c, note that we have used configuration defaults for the client FTP.DATA file. Most of the time the client defaults work fine. In the case of TLS, however, we need to override some of these defaults, either through command line options or through FTP.DATA options.

Next, look at the trace message marked with d. The client is missing a key ring. Recall that the key ring is used to store the certificates of the signing authority: the Certificate Authority. It is also used to store any client certificates that might be necessary or any received server certificates. However—this client has no key ring.

The name of the key ring can only be specified in the client FTP.DATA file. We are sharing the ring with all servers at this site. The ring we defined is owned by the user ID TCPIP and is named SharedRing1. It is stored in a RACF repository. We copy the sample for a client FTP.DATA ('hlq.SEZAINST(FTCDATA) into our TCPIPB.TCPPARMS dataset and rename it FTPCB31. Focus on the security options in the sample file, and edit the client FTP.DATA file to look like the following in Example 15-21.

Example 15-21 Client FTP.DATA file changes

```
; ----------------------------------------------------------------------
; 7. Security options
; ----------------------------------------------------------------------

;SECURE_MECHANISM  GSSAPI e; GSSAPI (Kerberos) or TLS
```

```
;SECURE_FTP        ALLOWED          ; Authentication indicator
;SECURE_CTRLCONN   CLEAR            ; Minimum level of security for
;SECURE_DATACONN   CLEAR            ; Minimum level of security for
;SECURE_HOSTNAME   OPTIONAL         ; Authentication of hostname in
;SECURE_PBSZ       16384            ; Kerberos maximum size of the
; Name of a ciphersuite that can be passed to the partner during
; the TLS handshake.
;CIPHERSUITE       SSL_NULL_MD5     ; 01 f
;CIPHERSUITE       SSL_NULL_SHA     ; 02
;CIPHERSUITE       SSL_RC4_MD5_EX   ; 03
;CIPHERSUITE       SSL_RC4_MD5      ; 04
;CIPHERSUITE       SSL_RC4_SHA      ; 05
;CIPHERSUITE       SSL_RC2_MD5_EX   ; 06
;CIPHERSUITE       SSL_DES_SHA      ; 09
;CIPHERSUITE       SSL_3DES_SHA     ; 0A
;CIPHERSUITE       SSL_AES_128_SHA  ; 2F
;CIPHERSUITE       SSL_AES_256_SHA  ; 35

KEYRING           *AUTH*/* g        ; Name of the keyring for TLS
;TLSTIMEOUT        100              ; Maximum time limit between full
;SECUREIMPLICITZOS TRUE             ; Specify whether client will
; ---------------------------------------------------------------------
```

In Example 15-21 on page 601 we have left the parameters specifying the type of security we want (SECURE_...) commented out e. These parameters, like the request for TLS or the request for a private data connection, can be specified on the ftp command itself. We have also left the CIPHERSUITES commented out f, because during the TLS negotiation with the server, the intersection of the default CIPHERSUITES from the client and the defined CIPHERSUITES at the server results in a list of cipher suites that will work for both client and server. The order in which the ciphers are coded in the server determines the order of negotiation. The first matching cipher that both the client and server support is chosen for the connection.

Focus on the KEYRING designation g. When we coded the key ring for the server we used the syntax KEYRING TCPIP/SharedRing1. Coding TCPIP/ in front of the key ring name allows multiple users to share the key ring. The coding you see here is also valid: KEYRING *AUTH*/*. This is a special case where RACF permits the client to specify a virtual key ring instead of a real one. Certain conditions must exist in order for the specification of a virtual key ring to be accepted, as explained in the discussion of virtual key rings in *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

Now that we have added the designation for the client's key ring to the client FTP.DATA file and referenced it with the -f parameter on the ftp command, we can re-execute the command to establish the secure connection:

```
ftp 10.1.1.40 -a TLS -f "//'TCPIPB.TCPPARMS(FTPCB31)'" (TCP TCPIPB
```

Example 15-22 Establishing a secure connection using a specific FTP.DATA file

```
Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration parameters.
FTP: using TCPIPB
Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS      at HOSTD.ITSO.IBM.COM, 15:21:35 on 2007-09-24.
220 Connection will close if idle for more than 5 minutes.
>>> AUTH TLS
234 Security environment established - ready for negotiation
Authentication negotiation succeeded h
NAME (10.1.1.40:CS02):
cs02
```

```
>>> USER cs02
331 Send password please.
PASSWORD:
>>> PASS
230 CS02 is logged on.  Working directory is "CS02.".
***
```

You see in Example 15-22 on page 602 that the negotiation has succeeded this time. Next, we entered the **status** subcommand from the client to verify that we truly had a TLS connection.

Example 15-23 FTP Client messages

```
status
 >>> STAT
 211-Server FTP talking to host ::ffff:10.1.4.21, port 1033
211-User: CS02  Working directory: CS02.
211-The control connection has transferred 182 bytes
211-There is no current data connection.
211-The next data connection will be actively opened
211-to host ::ffff:10.1.4.21, port 1033,
.............
211-Authentication type: TLS a
211-Control protection level: Private b
211-Data protection level: Clear c
211-TLS security is supported at the DRAFT level d
...
211 *** end of status ***
Command:
```

In Example 15-23, the connection has successfully negotiated TLS with the server a. The control connection has been designated as private b. The data protection level is clear c. TLS is operating at the DRAFT level d.

To determine which encryption algorithm was chosen during the negotiation, look at SMF records or at a network management product that interfaces with the NMI. Alternatively, you can enable DEBUG SEC in the client FTP.DATA file, as shown in Example 15-24, prior to starting the ftp file transfer:

```
ftp 10.1.1.40 -a TLS -f "//'TCPIPB.TCPPARMS(FTPCB31)'" (TRACE TCP TCPIPB
```

Example 15-24 The client FTP.DATA file with DEBUG SEC option enabled

```
DEBUG SEC
```

Now that debug for security (SEC) is in effect, the client log shows the selected encryption option. See the client display at TCPIPB on the line denoted with a in Example 15-25.

Example 15-25 Debug messages in client trace output

```
.........
Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS      at HOSTD.ITSO.IBM.COM, 17:53:53 on 2007-09-24.
220 Connection will close if idle for more than 5 minutes.
FCO232 ftpAuth: security values: mech=TLS, tlsmech=FTP, sFTP=A, sCC=C, sDC=C
FCO279 ftpAuth: ........ cipherspecs =
FCO320 ftpAuth: environment_open()
FCO438 ftpAuth: environment_init()
FCO447 ftpAuth: environment initialization complete
>>> AUTH TLS
```

```
234 Security environment established - ready for negotiation
........
FCO828 authServer: secure_socket_open()
FCO895 authServer: secure_socket_init()
FUO9O3 tlsLevel: using TLSV1 with SSL_RC4_MD5_EX (03) a
Authentication negotiation succeeded
GVO534 seq_stat_file(2): lrecl=0 recfm=0 blksize=0 mode=1
CZ1466 rnetrc:(3) file CSO2.NETRC does not exist
NAME (10.1.1.40:CSO2):
```

After logging in on the encrypted control connection, we transmit a file from client to server in the clear over the data connection, as shown in Example 15-26.

Example 15-26 File transfer over data connection in the clear

```
Command:
get 'cs02.ioacmd.OUTPUT.QUERY' 'cs02.myioacmd'
 >>> PORT 10,1,4,21,4,43
 200 Port request OK.
 >>> RETR 'cs02.ioacmd.OUTPUT.QUERY'
 125 Sending data set CS02.IOACMD.OUTPUT.QUERY FIXrecfm 80
 250 Transfer completed successfully.
 26925 bytes transferred in 0.010 seconds.  Transfer rate 2692.50 Kbytes/sec.
 Command:
```

Now we want to transmit a file and secure the transfer with encryption and integrity checking. Therefore, we need to enter the command to change the data connection to a private, encrypted one by entering the subcommand **private**. The command **protect private** would have accomplished the same thing. Example 15-27 shows setting the data connection encryption option to private and then transferring the file.

Example 15-27 Setting the data connection encryption option to private

```
private
 >>> PBSZ 0
 200 Protection buffer size accepted
 >>> PROT P
 200 Data connection protection set to private
 Data connection protection is private
 Command:
get 'cs02.jcl.cntl(certauth)' 'cs02.jcl.cntl(certcryp)'
 >>> PORT 10,1,4,21,4,46
 200 Port request OK.
 >>> RETR 'cs02.jcl.cntl(certauth)'
 125 Sending data set CS02.JCL.CNTL(CERTAUTH) FIXrecfm 80
 250 Transfer completed successfully.
 911 bytes transferred in 0.010 seconds.  Transfer rate 91.10 Kbytes/sec.
 Command:
```

The status subcommand (**stat**) from the client shows that the data connection is indeed private; see Example 15-28.

Example 15-28 STAT shows the Data protection level set to Private

```
211-Checkpoint interval is 0
211-Authentication type: TLS
211-Control protection level: Private
211-Data protection level: Private
211-TLS security is supported at the DRAFT level
```

Now we reset the data connection to clear, as shown in Example 15-29.

Example 15-29 Setting the data protection level back to clear

```
protect clear
>>> PROT C
200 Data connection protection set to clear
Data connection protection is clear
Command:
```

Disabling tracing

We disabled tracing at the z/OS server and at the z/OS client before continuing.

- At the server on TCPIPD, we issued the following command:

  ```
  F FTPDD1,DEBUG=NONE
  ```

You see the results in Example 15-30.

Example 15-30 Turning server debug tracing off

```
F FTPDD1,DEBUG=NONE
+EZYFT82I ACTIVE SERVER TRACES - NONE
```

At the client we commented out the DEBUG SEC option in the client FTP.DATA file named TCPIPB.TCPPARMS(FTPDB31), as you see in Example 15-31.

Example 15-31 Turning client debug tracing off

```
; DEBUG SEC
```

Use a TLS-enabled workstation client to verify server TLS (Draft)

In order to verify that our FTP server supports TLS with a non-z/OS client platform, we needed a non-z/OS FTP client that supports TLS. Filezilla, which we used in non-TLS mode, does support TLS. Therefore we continued to use it, and configure it for TLS support. Remember that Filezilla is an open source TLS-enabled FTP client available at:

http://filezilla.sourceforge.net

Our scenario with the separate key rings is depicted in Figure 15-8 on page 606.

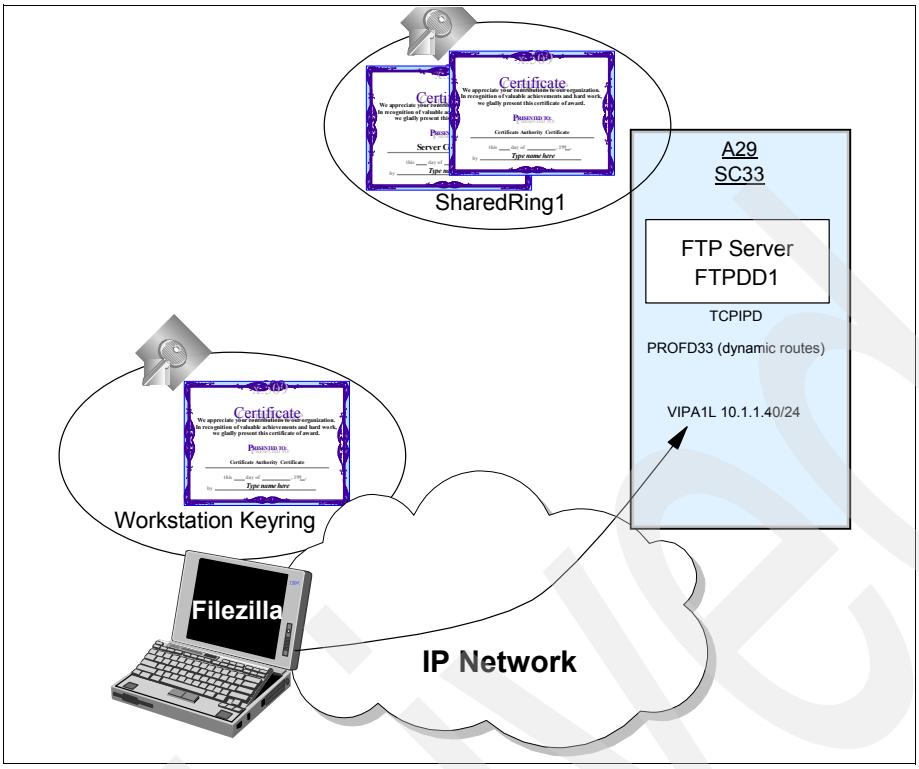


Figure 15-8 Use of a Client key ring separate from the Server key ring

The Filezilla client needs a key ring in which to store the Certificate Authority certificates and any server certificates it may want to receive. (It cannot share the RACF key ring with the server.) However, Filezilla does not include a key management utility. It is necessary to use the Microsoft® Management Console (MMC) to create the key ring for Filezilla.

Key ring creation and certificate management at Windows for Filezilla

Follow these steps:

1. FTP the EXPORTed Certificate Authority Certificate in ASCII into the workstation from which you are planning to run Filezilla. Receive the file as ITSOCACERT.cer.
   This is the certificate that you exported into an MVS dataset in “Export the CA certificate to a flat file for prepopulating client devices” on page 589.

   The certificate label is CS19 ITSO CA1 and was stored in our examples as TCPIP.CS19.CACERT.
2. From the Windows Start menu, select **Run**.
3. Enter `mmc` in the Open field. Click **OK** to start the MMC.
4. In the Console 1 window, click **File** from the menu bar. From the pull-down menu, click **Add/Remove Snap-in**.
5. In the Add/Remove Snap-in window, click **Add**.
6. In the Add Standalone Snap-in window, select **Certificates** and click **Add**.
7. In the Certificates Snap-in window, select **Computer account** and click **Next**.
8. Select **Local computer** and click **Finish**.
9. Click **OK**.
10. In the MMC window, click the plus sign (**+**) next to Certificates (Local Computer) to show the list of available tasks.
11. Right-click **Trusted Root Certification Authorities** and choose **All Tasks** from the pull-down menu. Choose **Import** from the next pull-down menu.

12. Click **Browse** and specify the Trusted Root CA file name that you imported into the workstation. In our case we downloaded this file in ASCII and renamed it to ITSOCACERT.cer. Click **Next**.
13. Specify that you are placing the certificate “in the following store.”
14. Click **Finish** and you receive a message that the import was successful.
15. Double-click the certificates and browse through the list of CA certificates to find the one you just imported. It is represented by the `ou` name that you assigned to it when you created it with the **`racdcert certauth`** command. The one that we created was identified by: `ou('ITSO Certificate Authority')`. See “Generate a RACF CA certificate” on page 587.

Filezilla configuration for TLS

The following figures show the configuration of Filezilla for TLS. In Figure 15-9 we show the configuration settings needed to connect to our TLS-enabled secure FTP server.

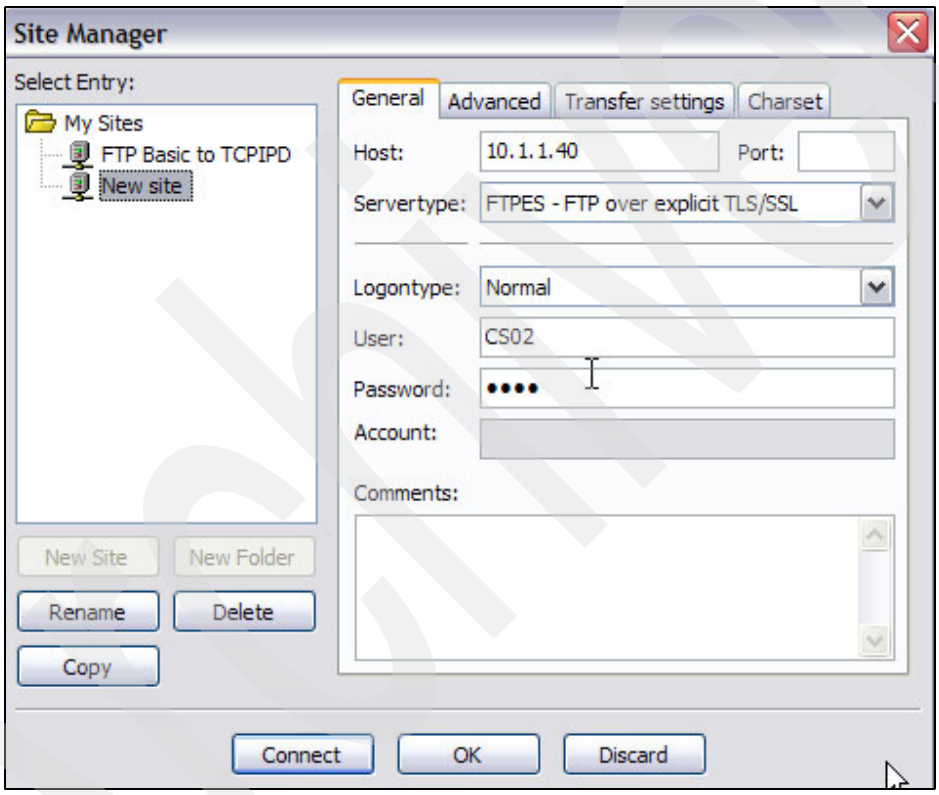


Figure 15-9 Configuration of Filezilla to connect to our TLS-enabled secure FTP server

Observe that in the Servertype pull-down we selected **FTPES - FTP over explicit TLS/SSL**. We fill in the other fields and click **Connect**. Next, we see the AUTH flow from the client that is requesting a TLS connection in Figure 15-10 on page 608.

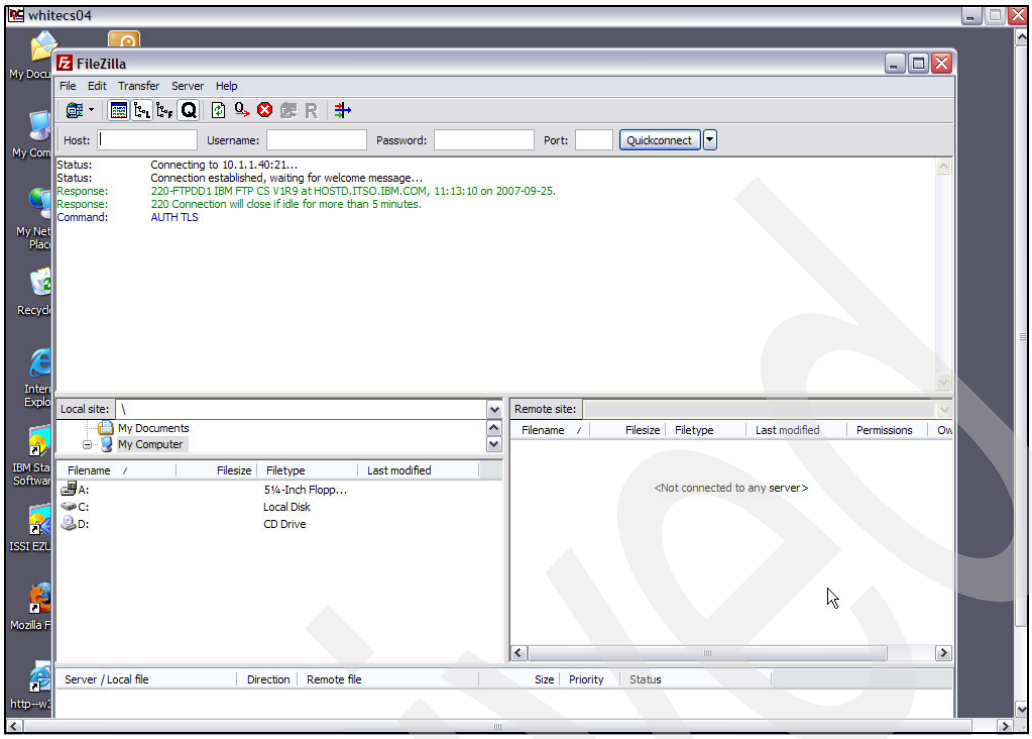


Figure 15-10 Client AUTH flow for a TLS connection request

In Figure 15-11 we show Filezilla asking for confirmation to accept or reject the unknown certificate received from the server.

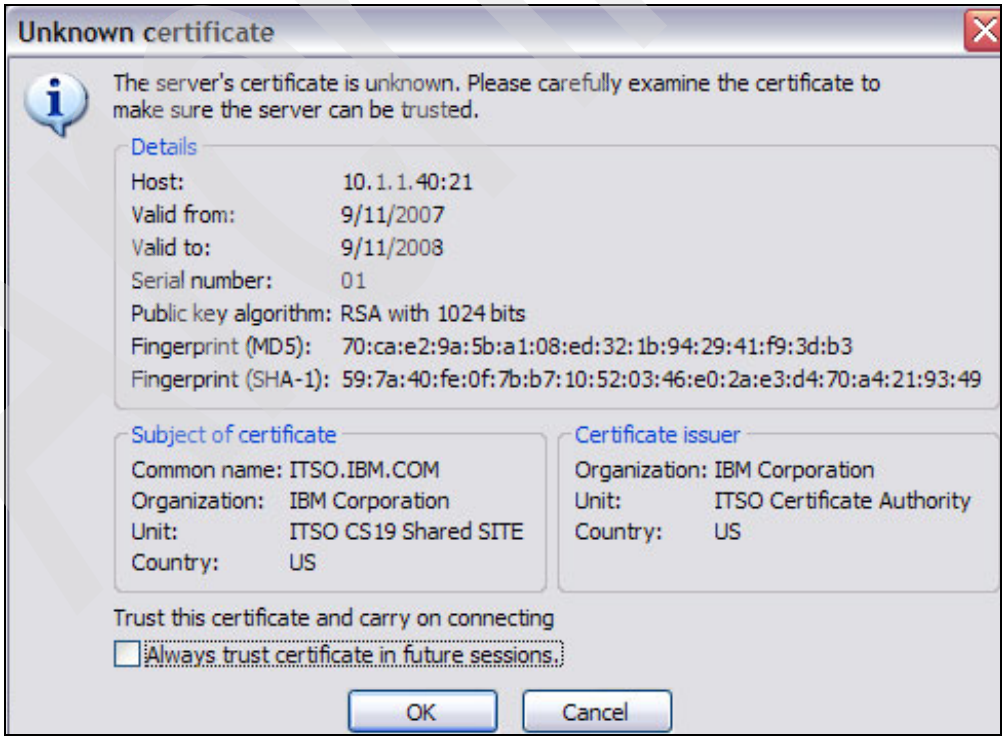


Figure 15-11 Filezilla confirming that the received server certificate is acceptable

We click **OK** to confirm that we accept this server certificate. Finally we had confirmation that Filezilla was able to connect to our secure FTP server; we received messages indicating that the TLS/SSL connection had been established and the data connection protection was set to private (**`PROT P`**).

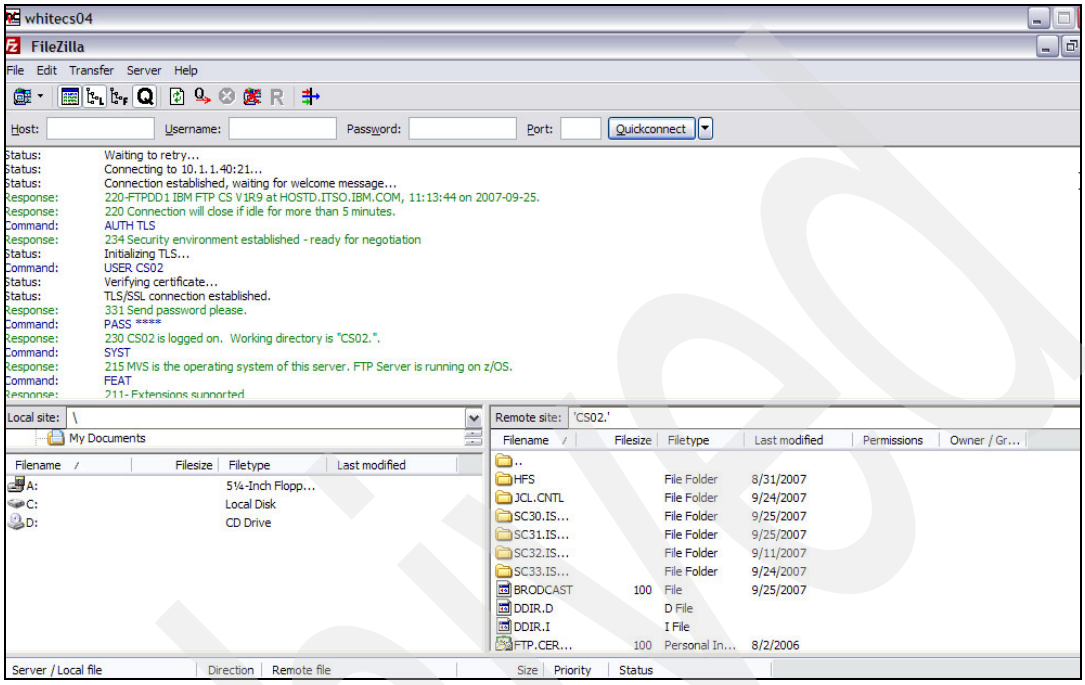


Figure 15-12 Confirmation that Filezilla connected to our system using TLS

15.3.5 Activation and verification of FTP server with TLS security: RFC4217 protocols

In 15.3.4, “Activation and verification of the FTP server with TLS security: Internet Draft protocols” on page 599 we show how to enable the Internet Draft version of FTP TLS in both a z/OS-to-z/OS environment and in a workstation-to-z/OS environment. In this section we enable z/OS at the RFC 4217 level and test again.

We assume that the tasks listed here have already been performed as described in “Create key ring and certificates and update RACF permissions” on page 586 and “Update the FTP.DATA for the server” on page 590.

We use the same processes we used previously to confirm that the FTP server has started with TLS and is working correctly:

- “Check FTP server job log messages” on page 592
- “Verify the non-TLS z/OS client can connect to the server” on page 592
- “Display system job tasks for control and data connections” on page 593
- “Use a TLS-enabled z/OS client to verify server TLS (RFC 4217)” on page 609
- “Use a TLS-enabled workstation client to verify server TLS (RFC 4217)” on page 615

Use a TLS-enabled z/OS client to verify server TLS (RFC 4217)

We now enable the z/OS client at the RFC 4217 level. This permits the client to enable or disable security on the control connection through the **`ccc`** command or its equivalent, **`cprotect clear`**. With the Internet Draft, the client cannot change the nature of the control

connection after the control connection has been negotiated as private. With the Draft level, the data connection could always be manipulated from the client side as long as the server agreed with the client request.

Our first step is to change the z/OS FTP server and the z/OS FTP client to conform to RFC 4217. We change the TLSRFCLEVEL statement in the server FTP.DATA in Example 15-32.

Example 15-32 Setting TLSRFCLEVEL

```
EXTENSIONS   AUTH_TLS
TLSMECHANISM FTP
KEYRING      TCPIP/SharedRing1
CIPHERSUITE       SSL_RC4_MD5_EX    ; 03
CIPHERSUITE       SSL_RC4_MD5       ; 04
CIPHERSUITE       SSL_3DES_SHA      ; 0A
CIPHERSUITE       SSL_DES_SHA       ; 09
CIPHERSUITE       SSL_NULL_MD5      ; 01
CIPHERSUITE       SSL_NULL_SHA      ; 02
SECURE_FTP   ALLOWED
SECURE_CTRLCONN   PRIVATE
SECURE_DATACONN CLEAR
TLSTIMEOUT        100
a TLSRFCLEVEL       RFC4217
```

TLSRFCLEVEL a in TCPIPD.TCPPARMS(FTPSTLS) has been changed from DRAFT to RFC4217. Otherwise this is the same FTP.DATA file we used when we originally started the FTP server for either non-TLS or TLS connections.

Next, we change the TLSRFCLEVEL for the z/OS client on TCPIPB in TCPIPB.TCPPARMS(FTPCB31) a to reflect the RFC 4217 level in Example 15-33:

Example 15-33 FTP.DATA for the client at TLSRFCLEVEL RFC4217

```
;*********************************************************************
; -------------------------------------------------------------------
; 7. Security options
; -------------------------------------------------------------------
;SECURE_MECHANISM  GSSAPI          ; Name of the security mechanism
;SECURE_FTP        ALLOWED         ; Authentication indicator
;SECURE_CTRLCONN   CLEAR           ; Minimum level of security for
;SECURE_DATACONN   CLEAR           ; Minimum level of security for
;SECURE_HOSTNAME   OPTIONAL        ; Authentication of hostname in
;SECURE_PBSZ       16384           ; Kerberos maximum size of the
 TLSRFCLEVEL       RFC4217 a
;CIPHERSUITE       SSL_NULL_MD5      ; 01
;CIPHERSUITE       SSL_NULL_SHA      ; 02
;CIPHERSUITE       SSL_RC4_MD5_EX    ; 03
;CIPHERSUITE       SSL_RC4_MD5       ; 04
;CIPHERSUITE       SSL_RC4_SHA       ; 05
;CIPHERSUITE       SSL_RC2_MD5_EX    ; 06
;CIPHERSUITE       SSL_DES_SHA       ; 09
;CIPHERSUITE       SSL_3DES_SHA      ; 0A
;CIPHERSUITE       SSL_AES_128_SHA   ; 2F
;CIPHERSUITE       SSL_AES_256_SHA   ; 35
 KEYRING      TCPIP/SharedRing1     ; Name of the keyring for TLS
 ;EYRING       *AUTH*/*             ; Name of the keyring for TLS
;TLSTIMEOUT        100             ; Maximum time limit between full
```

```
;SECUREIMPLICITZOS TRUE              ; Specify whether client will
```

We restart the FTP server (FTPDD) at TCPIPD. From ISPF, Option 6, at TCPIPB we enter the client command. TCPIPB is running in a CINET environment; therefore we needed to establish stack affinity to the desired stack for the FTP. We use the -a parameter to request a TLS connection and the -f parameter to establish affinity to the correct client FTP.DATA file:

```
ftp 10.1.1.40 -a TLS -f "//'TCPIPB.TCPPARMS(FTPCB31)'" (TCP TCPIPB
```

In Example 15-34, the connection succeeds.

Example 15-34 TCPIPB FTP client view of the connection at the RFC 4217 level

```
211-Authentication type: TLS
211-Control protection level: Private
211-Data protection level: Clear
211-TLS security is supported at the RFC4217 level a
. . . .
211 *** end of status ***
Command:
```

The client is operating under the protocols of RFC 4217 a. If you miss the 211 response about the TLS support level, you may query for the level with the command **locstat tlsrfclevel**, as in Figure 15-13.

```
 Command:
locstat tlsrfclevel
 local site variable TLSRFClevel is set to RFC4217
 Command:
```

Figure 15-13 Querying client TLSRFCLevel support

Likewise, you can investigate the RFC level at the server with the **stat (tlsrfclevel** subcommand; see Figure 15-14.

```
 Command:
stat (tlsrfclevel
 >>> XSTA (tlsrfclevel
 211-TLS security is supported at the RFC4217 level
 211 *** end of status ***
```

Figure 15-14 Querying server TLSRFCLevel support

From an OEM client, you could enter **quote xsta (tlsrfclevel** to do the same thing. Note that you must use the interpreted version of the command (**xsta**) when you execute the command in this fashion.

Figure 15-15 on page 612 shows how to set the local client TLSRFCLEVEL back to DRAFT mode.

```
 Command:
locsite tlsrfclevel=draft
 Command:
locstat tlsrfclevel
 local site variable TLSRFClevel is set to DRAFT
 Command:
```

Figure 15-15 Using LOCSITE client command to set the TLSRFCLEVEL

The full **status** command output in Figure 15-16 shows that the server is also operating under RFC 4217 **b**. In addition, it shows the protection levels for the control connection and the data connection.

```
211-Authentication type: TLS
211-Control protection level: Private
211-Data protection level: Clear
211-TLS security is supported at the RFC4217 level b
211-Port of Entry resource class for IPv4 clients is: TERMINAL
211-Record format FB, Lrecl: 128, Blocksize: 6144
211-Server site variable LISTSUBDIR is set to TRUE
211 *** end of status ***
Command:
```

Figure 15-16 Output from status command: FTP server operating parameters

Next we need to see if we can change the control protection level from Private to Clear, one of the capabilities of RFC 4217. Figure 15-17 on page 613 shows how to use the client command **ccc** to perform this task. (There is an equivalent command: **cprotect clear**.)

When TLSRFCLEVEL DRAFT is configured, these subcommands (**ccc** and **cprotect clear**) are not allowed during a TLS session. We connect to the server with the **-a TLS** specification as before; see Figure 15-17 on page 613.

```
 Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration parameters.
 FTP: using TCPIPB
 Connecting to:   10.1.1.40 port: 21.
 220-FTPDD1 IBM FTP CS      at HOSTD.ITSO.IBM.COM, 12:21:16 on 2007-09-25.
 220 Connection will close if idle for more than 5 minutes.
 >>> AUTH TLS
 234 Security environment established - ready for negotiation
Authentication negotiation succeeded
 NAME (10.1.1.40:CS02):
cs02
 >>> USER cs02
 331 Send password please.
 PASSWORD:
>>> PASS
 230 CS02 is logged on.  Working directory is "CS02.".
 Command:
ccc
 >>> CCC
 200 CCC command successful
 Control connection protection is clear
 Command:
status
 >>> STAT
 211-Server FTP talking to host ::ffff:10.1.4.21, port 1090
 . . . .
 211-Authentication type: TLS
 211-Control protection level: Clear
 211-Data protection level: Clear
 211-TLS security is supported at the RFC4217 level
 . . .
211 *** end of status ***
 Command:
```

Figure 15-17 Clearing the control connection: `ccc`

After you clear the control connection, you cannot alter the state of the data connection. Therefore, if for some reason you wanted to have a protected data connection and a clear control connection, the data connection state must be protected *prior* to issuing the **ccc** command.

Next, we reset the control connection to private with the **auth tls** command. When we attempt the **ccc** command again, we are told that we must enter user ID and password again as in a of Figure 15-18 on page 614. RFC 2228, upon which RFC 4217 is built, specifies that after an auth command is issued to change a previous state, the user must reauthorize. This is why we were required to enter our user ID and password again.

```
 Command:
auth tls
 >>> AUTH TLS
 234 Security environment established - ready for negotiation
Authentication negotiation succeeded
 Command:
status
 >>> STATUS
.............
 211-Authentication type: TLS
 211-Control protection level: Private
 211-Data protection level: Clear
 211-TLS security is supported at the RFC4217 level
............
 211 *** end of status ***
 Command:
ccc
 >>> CCC
 530 You must first login with USER and PASS. a
 Cannot set protection level to clear
 Control connection protection is private
 Command:
user cs02
 >>> USER cs02
 331 Send password please.
 PASSWORD:
>>> PASS
 230 CS02 is logged on.  Working directory is "CS02.".
 Command:
ccc
 >>> CCC
200 CCC command successful
Control connection protection is clear
Command:
```

Figure 15-18 Protecting the control connection and clearing it again: auth tls and ccc

We next test a non-secure connection to the FTP server from the same client. We want to determine whether we are able to convert an unprotected connection into a secure one with the **auth tls** command if we are operating at the RFC 4217 support level. We ftp into the server with the following command:

```
ftp 10.1.1.40 -f "//'TCPIPB.TCPPARMS(FTPCB31)'"  (TCP TCPIPB
```

```
. . . . . .
 Connecting to:   10.1.1.40 port: 21.
 220-FTPDD1 IBM FTP CS V1R9 at HOSTD.ITSO.IBM.COM, 13:46:02 on 2007-09-25.
 220 Connection will close if idle for more than 5 minutes.
 NAME (10.1.1.40:CSO2):
cs02                                a
 >>> USER cs02
 331 Send password please.
 PASSWORD:
 >>> PASS
 230 CSO2 is logged on.  Working directory is "CSO2.".
 Command:
stat (tlsrfclevel
 >>> XSTA (tlsrfclevel
 211-TLS security is supported at the RFC4217 level   b
 211 *** end of status ***
 Command:
status
 >>> STAT
 211-Server FTP talking to host ::ffff:10.1.4.21, port 1028
 211-User: CSO2  Working directory: CSO2.
 211-The control connection has transferred 209 bytes
 211-Authentication type: None    c
 211-TLS security is supported at the RFC4217 level
```

Figure 15-19 An FTP connection without protection

In Figure 15-19 at a, note that the **auth tls** command does not flow because we did not invoke it and did not include it in the client FTP.DATA file. Observe at b that the server supports RFC 4217. You can verify at c that this connection has been established with no authentication mechanism in place.

When we now issue the **auth tls** command, as shown in Figure 15-20, our negotiation fails. This is expected behavior because System SSL cannot support this activity because the FTP daemon changes identity when the user logs in.

```
 Command:
auth tls
 >>> AUTH TLS
 234 Security environment established - ready for negotiation
 Authentication negotiation failed
 Command:
```

Figure 15-20 Negotiation fails: an initially clear control connection cannot be set to private

Use a TLS-enabled workstation client to verify server TLS (RFC 4217)

We now use FileZilla to test the TLS-enabled server operating with RFC 4217 protocols.

We are testing with the same Certificate Authority certificate and do not need to use MMC again to establish the key ring that Filezilla needs as we did in “Filezilla configuration for TLS” on page 607. The configuration for Filezilla is the same one we set up in “Use a TLS-enabled workstation client to verify server TLS (Draft)” on page 605.

We connected the FTP client to the server at TCPIPD successfully. We used the Server pull-down to select **custom commands** and entered **stat** to confirm the use of the TLS protocols, as shown in Figure 15-21 on page 616.

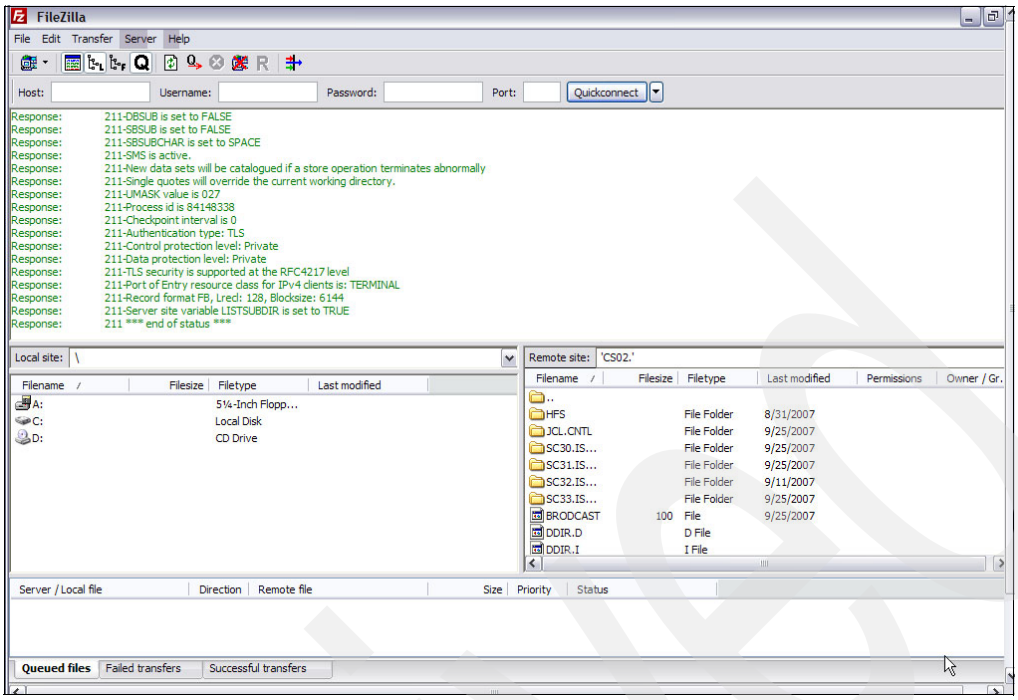


Figure 15-21 Results of client stat command: TLS Level 4217 at the server

We then used the server pull-down to enter the command **`prot c`** in order to obtain a clear data connection. The results are shown in Figure 15-22.

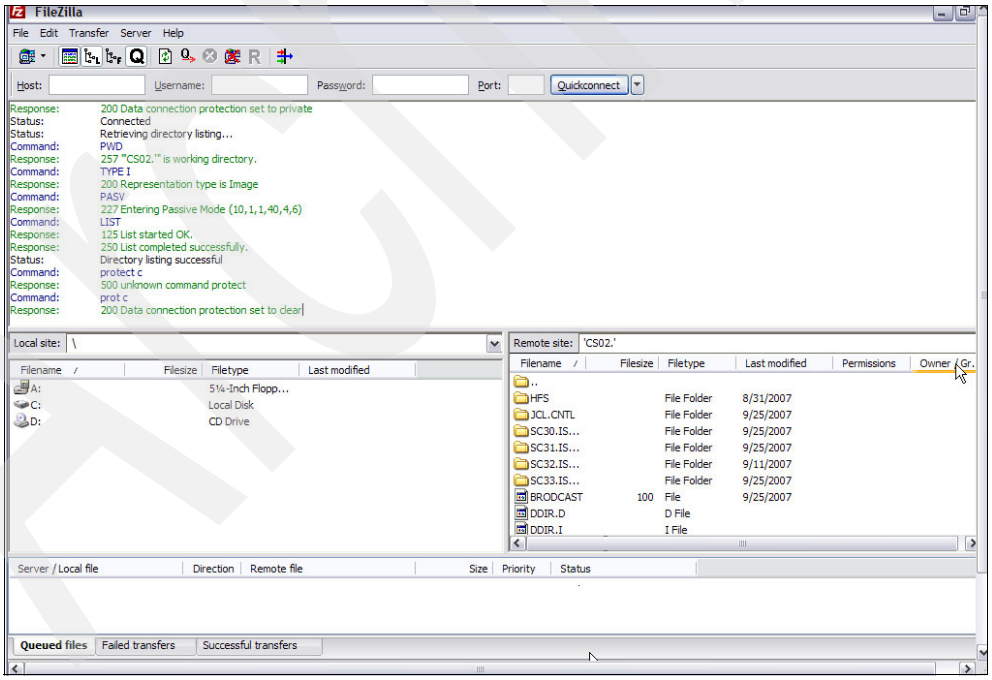


Figure 15-22 Results of client entering prot c to clear the data connection

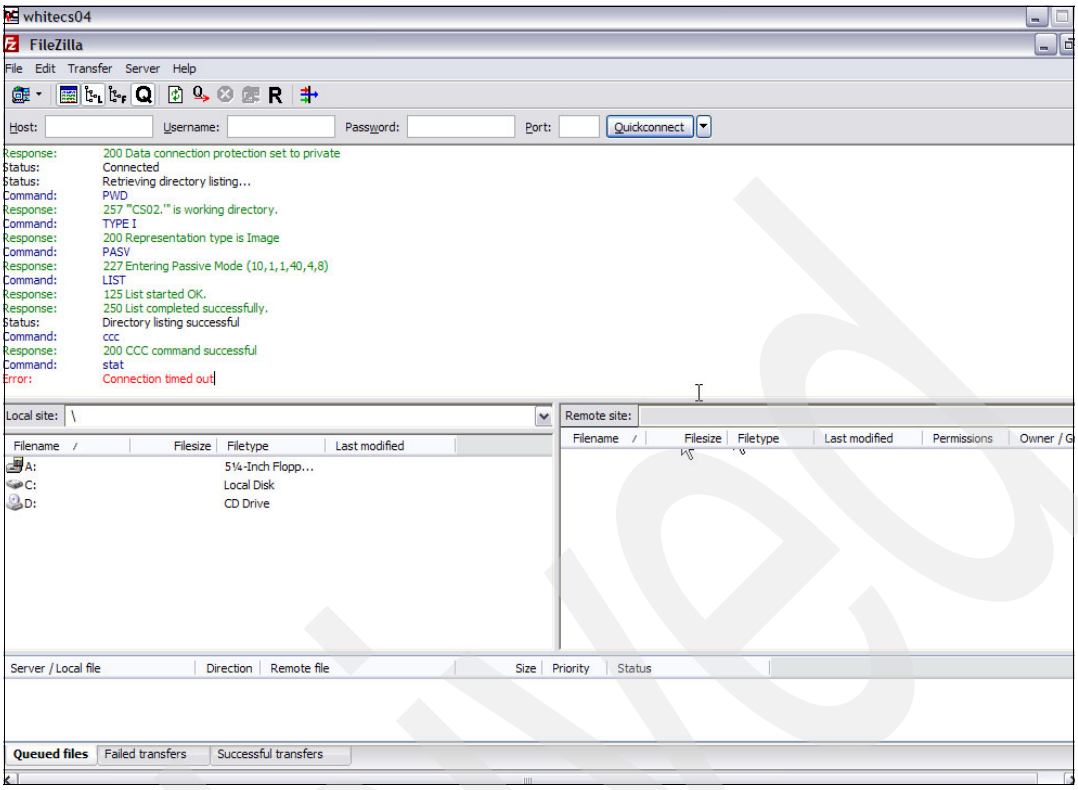


Figure 15-23 Entry of ccc command by client; subsequent failure with stat command

Although Filezilla sends the **ccc** command, it cannot support a clear control connection. Therefore when the next command, **stat**, needs to flow over the control connection, the connection is terminated.

Recommendation: Unless both the client and the server are RFC 4217-compliant, the advanced functions, like ccc, cannot be used on the connection without experiencing a disconnect or some other unpredictable occurrence.

Although a z/OS server that is running with RFC 4217 may still communicate with downlevel clients using standard TLS protocols, our recommendation is to use the DRAFT level unless you are certain that your FTP clients also support the RFC 4217 level, or that your clients will refrain from executing commands like **ccc** or even **REIN**, which we have not tested here.

15.4 FTP with AT-TLS security support

This section discusses the following topics:

- Description of FTP AT-TLS support
- Configuration of FTP AT-TLS support
- Activation and verification of FTP AT-TLS support

An invaluable resource for implementing FTP with AT-TLS is *z/OS Communications Server: IP Configuration Guide*, SC31-8775. Always refer to it for additional detail or for clarification of the individual implementation steps.

15.4.1 Description of FTP AT-TLS support

Note: For information about AT-TLS in general, consult Chapter 11, “Application Transparent Transport Layer Security” on page 443.

The FTP client and server can be configured to use AT-TLS to support TLS connections.

There are three types of AT-TLS applications:

- Those that are *not aware* of using AT-TLS because they require no code changes to the application itself
- Those that are *aware* of AT-TLS and can query the TCPIP stack but cannot control it or affect the choices that AT-TLS makes
- Those that are AT-TLS *controlling*, meaning that the application starts and stops security on the connection, through the AT-TLS mechanism

FTP belongs to the third category: it is a controlling AT-TLS application that requires the ApplicationControlled On statement in the AT-TLS policy. The FTP protocol relies on two connections: the control connection and the data connection. Therefore, for AT-TLS, FTP must also be defined with the SecondaryMap On statement in the policy file.

Dependencies of AT-TLS

You must have an appropriate understanding of the type of security that is acceptable to a particular business site, and be aware of the software and hardware prerequisites. In fact, the dependencies we presented for native TLS in 15.3, “FTP with native TLS security support” on page 583 apply to AT-TLS, as well. We urge you to review those dependencies in that section.

Additional dependencies for AT-TLS exist because it is implemented as part of policy-based networking. AT-TLS for FTP relies on the definition of AT-TLS policies that Policy Agent downloads into the TCP/IP stack where they are enforced. It also relies on FTP configuration changes in the server and in the client. Finally, since it is invoking TLS protocols, AT-TLS requires at least a server x.509 certificate and key, possibly together with a certificate from a signing authority (a Certificate Authority certificate). Depending on a site’s requirements, a client certificate for client authentication may be necessary.

With AT-TLS, SSLv2 is possible, but it is disabled by default in the AT-TLS policy.

Recommendation: Do not implement with SSLv2, even though it is permitted. You should turn on SSLv2 only if you must support connections with older applications that do not support the more secure protocols of SSLv3 or TLSv1.

Benefits of using AT-TLS

Just as with native TLS, FTP with AT-TLS security provides a more secure environment for the transmission of data. It provides user authentication, encryption, and data integrity checking. Application-Transparent TLS (AT-TLS) has considerable advantages over standard TLS/SSL for the FTP environment. AT-TLS can exploit:

- Certificate labels, thus eliminating the need to rely only on the presence of a default certificate in a key ring; key ring administration is simplified because many certificates can reside in the keyring
- Session key refresh during the lifetime of a session, reducing the possibility of a key compromise
- Certificate revocation lists (CRLs) that are maintained in LDAP servers

If you implement AT-TLS, the security policies can be placed under the control of a Centralized Policy Server, also known as a Distributed Policy Server. This subject is the topic of Chapter 6, “Central Policy Server” on page 257.

Considerations for AT-TLS

Just as with native TLS in FTP, the use of AT-TLS security requires more configuration and requires management of key rings. Keep in mind that a native SSL or TLS client can communicate with a server that has been implemented with AT-TLS. Likewise, an AT-TLS client can communicate with a server that is implemented with native TLS. The underlying TLS protocols are the same in either case.

Important: Although native TLS with FTP relies on the existence of a default certificate in a keyring, AT-TLS imposes no such requirement. AT-TLS supports certificate labels, or can continue to use a default certificate in the absence of a certificate label specification. This can simplify the administrative burden of maintaining multiple key rings, because one key ring can be used to hold a Certificate Authority certificate, a default certificate, and many others that are referenced with their label names.

15.4.2 Configuration of FTP AT-TLS support

It is always good practice to begin with a tested FTP client and server that work without security definitions. The tasks for a basic FTP server and client are described in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

We also need to create a server certificate in the server LPAR. The server certificate might be a self-signed server certificates, or it might be a personal or site certificate that has been signed by a Certificate Authority. We need one or more key rings into which we can store the certificates.

For AT-TLS, we make changes to the FTP.DATA file for the server and optionally, to the z/OS client. Finally, we need to **`permit`** the user ID of the FTP server to the appropriate RACF classes.

We tested two scenarios with FTP and AT-TLS security:

1. z/OS client to z/OS server
   a. We verified that the standard z/OS TLS client requesting TLS worked as expected when AT-TLS was enabled at the server.
   b. We verified that a z/OS AT-TLS client at TCPIPB could establish connections to the AT-TLS FTP server at TCPIPD.
2. Workstation client to z/OS server
   a. We verified that the workstation client invoking standard TLS could operate with the FTP server configured for AT-TLS on the TCPIPD stack.

Perform the following tasks to set up FTP server TLS support:

- Create key ring and certificates for AT-TLS.
- Update the FTP.DATA for the server for AT-TLS.
- Create the Policy Agent procedure and build the FTP AT-TLS policy.
  You may define the policy manually or with the z/OS Communications Server Configuration Assistant GUI. (We chose to use the GUI.)
- Update the TCP/IP stack for AT-TLS.
- TCP/IP stack changes.
- (Optional) Create certificates for client authentication.
- (Optional) Update the client’s TCP/IP stack to support AT-TLS if using it with the client.

Create key ring and certificates for AT-TLS

In “Create key ring and certificates and update RACF permissions” on page 586, we configured a shared key ring and a Certificate Authority certificate together with a shared server SITE certificate for use with TLS. Here we are using the same key ring and certificates to test our AT-TLS scenarios.

Update the FTP.DATA for the server for AT-TLS

The FTP server at startup needs to request AT-TLS capability. In the TLS scenarios, we defined or defaulted TLSMECHANISM FTP. With AT-TLS, this statement needs to be changed to TLSMECHANISM ATTLS. The same configuration statement is valid for the FTP client in its FTP.DATA file if the client chooses not to specify the value in the FTP request itself. There is a sample of the server FTP.DATA file in hlq.SEZAINST(FTPSDATA).

Refer to Example 15-35 to see a sample of the server’s security section in the FTP.DATA file when AT-TLS is being used. Note the reference to TLSMECHANISM ATTLS at a. Note also that many of the security definitions are commented out b. The security statements that are commented out apply to native TLS. With AT-TLS, the definitions that are not required in the server FTP.DATA file move into the AT-TLS policy configuration file. If the definitions remain in the server FTP.DATA file, they are ignored when TLSMECHANISM ATTLS is coded.

Example 15-35 FTP.DATA for AT-TLS implementation

```
1 EXTENSIONS   AUTH_TLS
2 TLSMECHANISM ATTLS a
3 ;;KEYRING      TCPIP/SharedRing1 b
4 ;;CIPHERSUITE       SSL_RC4_MD5_EX     ; 03 b
4 ;;CIPHERSUITE       SSL_RC4_MD5        ; 04 b
4 ;;CIPHERSUITE       SSL_3DES_SHA       ; 0A b
4 ;;CIPHERSUITE       SSL_DES_SHA        ; 09 b
4 ;;CIPHERSUITE       SSL_NULL_MD5       ; 01 b
4 ;;CIPHERSUITE       SSL_NULL_SHA       ; 02 b
5 SECURE_FTP   ALLOWED
6 SECURE_CTRLCONN   PRIVATE
7 SECURE_DATACONN CLEAR
8 ;;TLSTIMEOUT        100 b
9 TLSRFCLEVEL       DRAFT
```

We examine each statement in Example 15-35 to understand what exactly should be in the server FTP.DATA file for AT-TLS, and what should be defined instead in the AT-TLS policy. This look at individual parameters helps you understand how to configure a completely new secure FTP server and also how to migrate from an existing TLS server to an AT-TLS server.

The following items explain the statements in Example 15-35.

1 Enable a secure server with TLS by coding EXTENSIONS AUTH_TLS.

2 Indicate ATTLS support (not native FTP TLS support)

3 and b The KEYRING statement which identifies the location of our key ring file is commented out. The keyring definition must move to the AT-TLS policy file.

4 and b The CIPHERSUITE statements identifying each desired encryption algorithm are commented out. They also move to the AT-TLS policy file.

5 Add (or leave) a SECURE_FTP ALLOWED statement to allow for, but not require, TLS clients. In this fashion, the same FTP port can be used for non-secure or secure connections.

6 Add (or leave) a SECURE_CTRLCONN PRIVATE statement. This statement is ignored unless the client requests a secure connection. We recommend coding SECURE_CTRLCONN PRIVATE if the TLS mechanism is enabled, because the control connection carries user IDs and passwords. It is common practice to protect the user ID and the password even if the data connection need not be secure.

7 Add (or leave) a SECURE_DATACONN CLEAR statement to allow the client to determine whether data traffic really needs encryption. Encryption carries a performance price and may not be necessary for every data transfer.

8 and b The TLSTIMEOUT statement moves to the AT-TLS policy file and is thus commented out in the FTP.DATA file. This specifies the maximum time between full TLS handshakes.

9 Add or default TLSRFCLEVEL DRAFT. Alternatively, you may specify TLSRFCLEVEL RFC4217. DRAFT indicates that the FTP TLS protocol is following the draft RFC named “On Securing FTP with TLS - revision 05”. RFC4217 indicates that the FTP TLS protocol is following RFC 4217, “Securing FTP with TLS“. Both explain how to use RFC 2228 commands to implement TLS security. Refer to “Considerations” on page 585 and 15.3.5, “Activation and verification of FTP server with TLS security: RFC4217 protocols” on page 609 for a description of the significance of the DRAFT versus RFC4217 specifications.

Create the Policy Agent procedure and build the FTP AT-TLS policy

The following topics are discussed in this section:

- Policy Agent procedure
- The PAGENT standard environment variable file
- Policy Agent configuration files
- AT-TLS policy for z/OS FTP server and z/OS FTP client
- Using the Configuration Assistant to create an FTP server AT-TLS policy
- Modifying a traffic descriptor

Policy Agent procedure

There are many ways to code the procedure explained in the sample file in hlq.SEZAINST(PAGENT). Refer to Chapter 5, “Policy Agent” on page 221 for instructions on setting up the PAGENT procedure for Policy Agent. We structured our PAGENT procedure as you see in Example 15-36, ensuring that the STDENV dataset for the Language Environment was allocated with a Variable format, as is required by Language Environment.

Example 15-36 Sample PAGENT procedure

```
//PAGENT   PROC
//PAGENT   EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
//            etc/pagent&SYSCLONE..conf -l SYSLOGD -d1'
//STDENV   DD DSN=TCPIP.SC&SYSCLONE..STDENV(PAGENV&SYSCLONE),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

The PAGENT standard environment variable file

The Language Environment file, which resides in a dataset with Record Format V, and which is referenced in the procedure, is depicted in Example 15-37.

Example 15-37 Policy Agent environment variable file

```
PAGENT_CONFIG_FILE="//'TCPIPD.TCPPARMS(PAGCFG33)'"
PAGENT_LOG_FILE=/tmp/pagentd.log
```

```
LIBPATH=/usr/lib
TZ=EST5EDT4
```

Note: The specification of the configuration file in the JCL overrides the PAGENT_CONFIG_FILE definition in the Language Environment file.

Policy Agent configuration files

These are the configuration files that we built in the zFS.

Example 15-38 Main PAGENT configuration file

```
;; pagent33.conf = main configuration file stored on /etc directory of zFS
   Loglevel 15
   TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH PURGE 600
```

We modeled the main file after the samples file in /usr/lpp/tcpip/samples/pagent.conf. Our TCPIMAGE file for TCPIPD is depicted in Example 15-39.

Example 15-39 Image configuration file for stack TCPIPD

```
;; pagent33_TCPIPD.conf = Image configuration file stored on /etc directory of zFS
;RoutingConfig /etc/pagent33_PBR.conf
QoSConfig     /etc/pagent33_QoS.conf
TTLSConfig     /u/cs02/TCPIPDTTLS.policy     FLUSH PURGE
```

Our TCP/IP AT-TLS policy file is the one we are going to create using the IBM Configuration Assistant in the next steps. Refer to “Using the Configuration Assistant to create an FTP server AT-TLS policy” on page 624 for more information.

After we have tested it thoroughly we will merge it with the other AT-TLS policies using the IBM Configuration Assistant tools, thus creating one “master” AT-TLS policy file for TCPIPD. Then we will move the merged file into the /etc directory of the zFS and rename it to the shop standard: /etc/pagent33_TTLS.policy.

AT-TLS policy for z/OS FTP server and z/OS FTP client

The most important points to keep in mind when defining a policy for an FTP server or client, as explained in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, are the following:

- The FTP server
  - Requires the **`ApplicationControlled On`** statement in the AT-TLS policy. This parameter allows FTP to start and stop TLS security on a connection.
  - Requires the **`SecondaryMap On`** statement in the AT-TLS policy. This parameter enables active or passive data connections to use the same policy that is in place for the control connection. You do not need to code any additional TTLSRule statements for the data connections.
  - Requires the Direction parameter with the value Inbound on the TTLSRule statement for the FTP server
  - If the SECURE_LOGIN statement is coded in FTP.DATA with the parameters REQUIRED or VERIFY_USER, then the HandshakeRole parameter value must be ServerWithClientAuth.
- The FTP client
  - Requires the HandshakeRole parameter with the value Client to be coded on the TTLSEnvironmentAction statement.
  - Requires the Direction parameter with the value Outbound on the TTLSRule statement for the FTP client.

Guideline: The FTP server and client do not support SSLv2 when using TLSMECHANISM ATTLS. AT-TLS defaults to disabling SSLv2. SSLv2 should not be enabled in AT-TLS unless explicitly required by a remote system.

If SSLv2 is required by a remote system, use a specific TTLSRule statement for the remote system that points to a TTLSConnectionAction statement enabling SSLv2.

You have a few choices about how to build an AT-TLS policy. One is to create the policy file manually; another is to use the IBM Configuration Assistant for z/OS Communications Server, a GUI that you can download from the z/OS Communications Server support Web site.

If you choose to create the policy file manually you can code it “from scratch” or you can use the AT-TLS sample provided in /usr/lpp/tcpip/samples/pagent_TTLS.conf of the zFS or HFS file system on z/OS.

After you become familiar with the parameters and keywords of a policy file, it can be simple to manipulate one created manually. However, if you are not familiar with the syntax, it can be daunting, as you can see from the excerpt from pagent_TTLS.conf depicted in Example 15-40

Example 15-40 FTP Server AT-TLS rules and actions

```
1TTLSRule                   Secure_Ftpd
{
 LocalPortRange           21
 Direction                Inbound
 TTLSGroupActionRef       grp_Production a
 TTLSEnvironmentActionRef Secure_Ftpd_Env b
}
# Common Production Group that all Rules use
2TTLSGroupAction grp_Production      a
{
 TTLSEnabled On
 Trace 2             # Log Errors to syslogd
}
# Environment Shared by all secure FTP server connections.
3TTLSEnvironmentAction Secure_Ftpd_Env b
{
 HandshakeRole Server c
 TTLSKeyRingParms d
 {
   Keyring FTPDsafkeyring # The keyring must be owned by the FTP server
 }
TTLSEnvironmentAdvancedParms
 {
   ApplicationControlled On e
   SecondaryMap  On   # include data connections f
   SSLV2         On   # Allow SSLv2 connections (Default is Off) g
   SSLV3         On   # Allow SSLv3 connections (Default is On ) g
   TLSV1         On   # Allow TLSv2 connections (Default is On ) g
 }
}
```

In Example 15-40, we can see three major sections for the AT-TLS policy rule:

1 is a **TTLSrule** named **Secure_ftpd**. The rule references two items:

- TTLSGroupActionRef named grp_Production a
- TTLSEnvironmentActionRef named Secure_Ftpd_Env b

2 defines a common TTLSGroupAction named Grp_Production.This is the group action that is referenced in the TTLSrule a.

3 defines the specific Actions for the FTP environment. It is the TTLSEnvironmentAction named Secure_Ftpd_Env.

- This is the environment action that is referenced in the TTLSrule b.
- This environment action contains three types of definitions: the handshake role c, the key ring parameters d, and the advanced parameters e, f, g.

After looking at this example with its complex syntax, you can appreciate why we prefer to use the Configuration Assistant GUI to define our AT-TLS policies. Not only does it produce a flat file like the one in the sample but it also performs a “health check” that warns of logic errors that you may have introduced into the policy.

Using the Configuration Assistant to create an FTP server AT-TLS policy

Download the Configuration Assistant from the Internet at:

`www.ibm.com/software/network/commserver/zos/support/`

Search for IBM Configuration Assistant for z/OS Communications Server. Install it on your workstation and initialize it. At startup, you are presented with a Main Perspective panel, as shown in Figure 15-24.

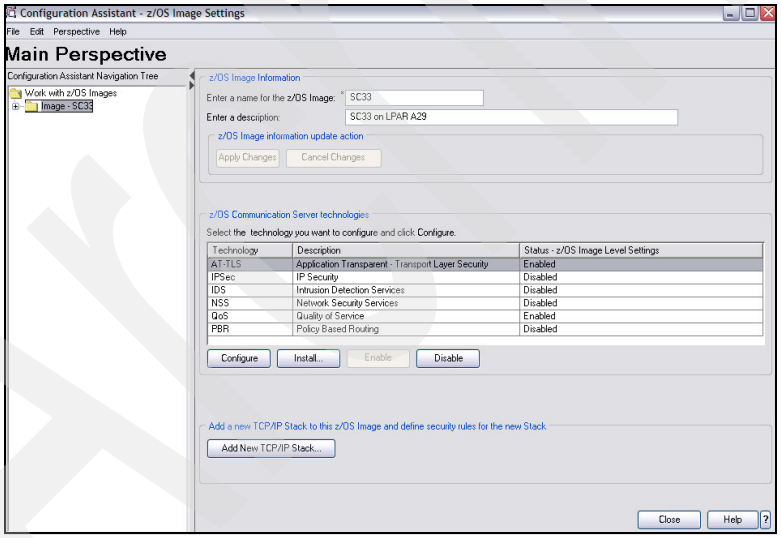


Figure 15-24 Main Perspective panel - Configuration Assistant

From the File pull-down, select **Open --> Create a new backing store file**. We named ours LPARA29.TTLS and allowed it to be stored in the IBM Configuration Assistant default directory.

Note: The term *backing store file* is common in the Java environment. The main repository of the policy definitions is stored as a collection of Java objects. The policies themselves are translated to ASCII before they are shipped via FTP to the target TCPIP stack for Policy Agent processing.

Our Configuration Assistant panel then looked as shown in Figure 15-25.

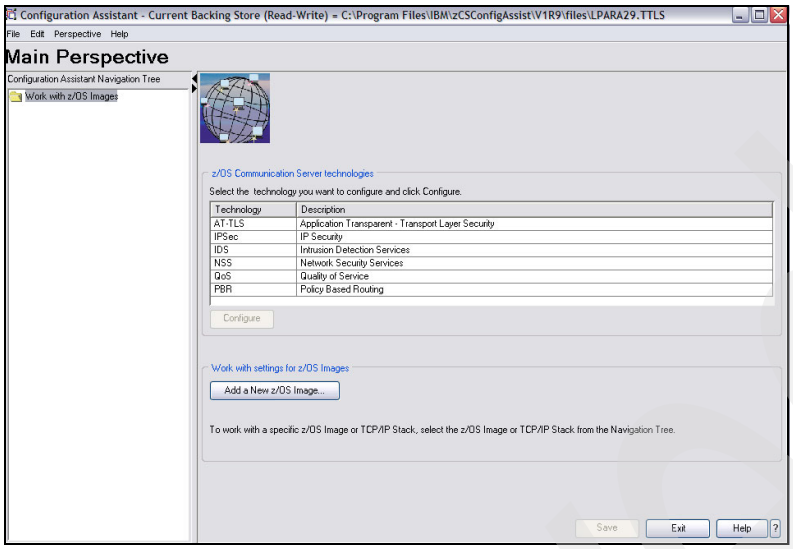


Figure 15-25 Main Perspective - Current Backing Store panel

This is the main perspective of the GUI tool. In the left pane we highlighted Work with z/OS Images and clicked **Add a New z/OS Image**. We were presented with the panel in Figure 15-26.

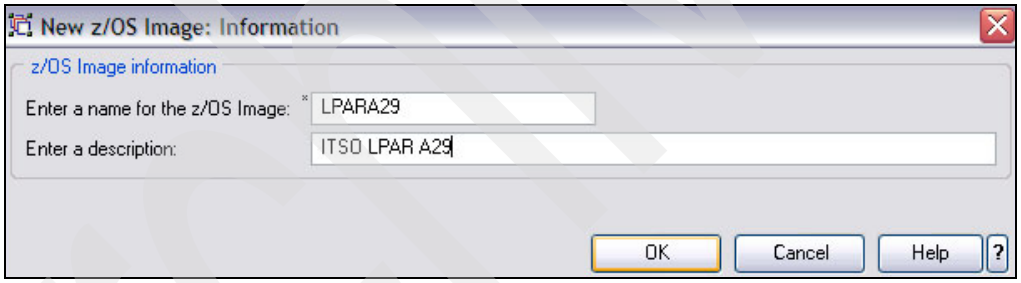


Figure 15-26 New z/OS image information for IBM Configuration Assistant

We entered the information as shown and clicked **OK**. The next image presented us with the confirmation prompt shown in Figure 15-27.

Figure 15-27 Confirmation prompt requiring a TCP/IP stack definition

We responded with **Yes**, because we want to add a TCP/IP stack to the z/OS image. The next panel, shown in Figure 15-28 on page 626, enabled us to enter the name of the TCP/IP stack.

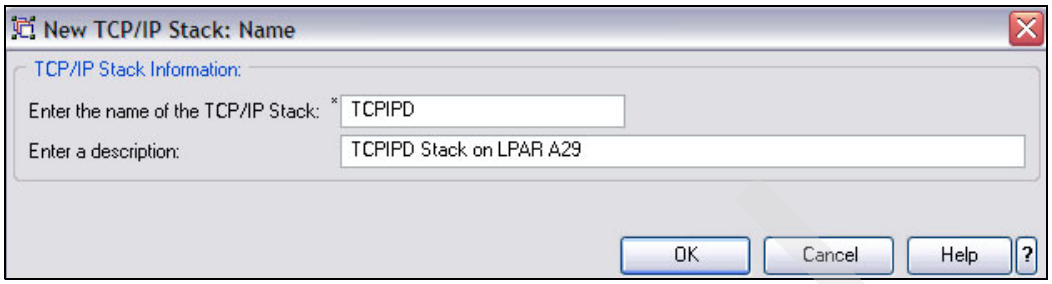


Figure 15-28 Providing the name and description of the TCP/IP stack

We clicked **OK** after filling out the fields and were presented with a new panel that enables the selection of AT-TLS policy definitions; see in Figure 15-29.

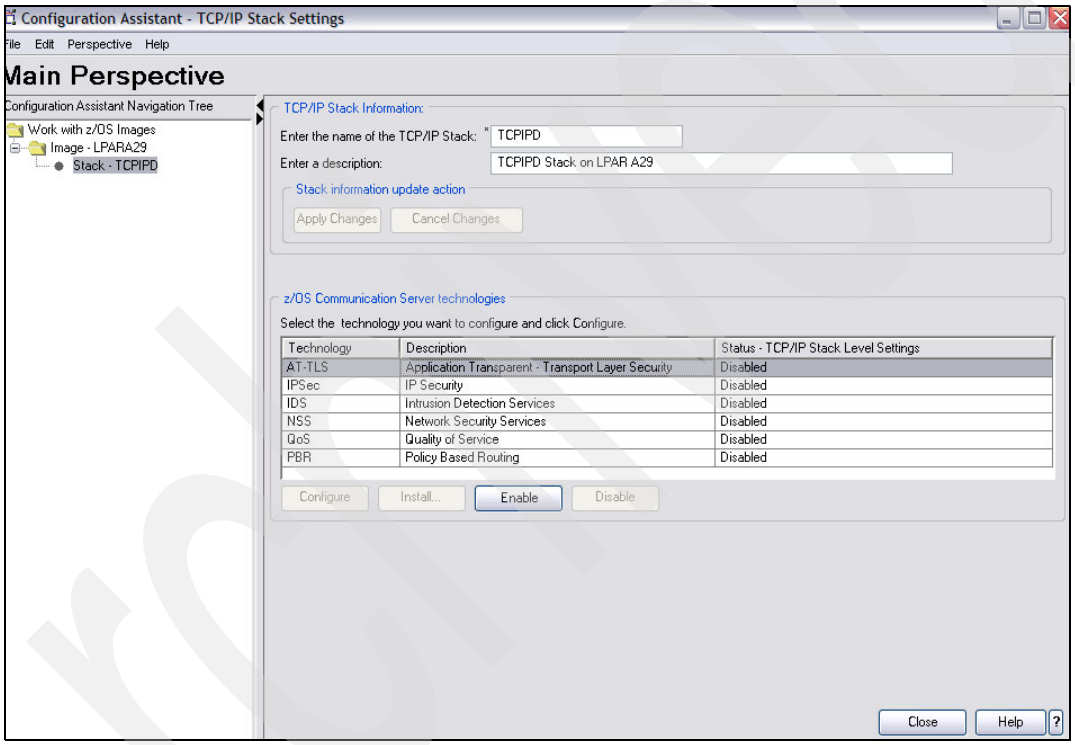


Figure 15-29 Begin to define AT-TLS policy for Stack TCPIPD on z/OS Image LPAR A29

At this point, we had defined a z/OS image and a TCP/IP stack. We were ready to begin the definition of the AT-TLS policy for the FTP server. We had highlighted the technology named AT-TLS. We clicked **Enable**, because AT-TLS is currently disabled on the TCPIPD stack in this backing store file. The status changed to `Incomplete`, because we still needed to create the traffic descriptors to build the Policy Rule and Policy Action.

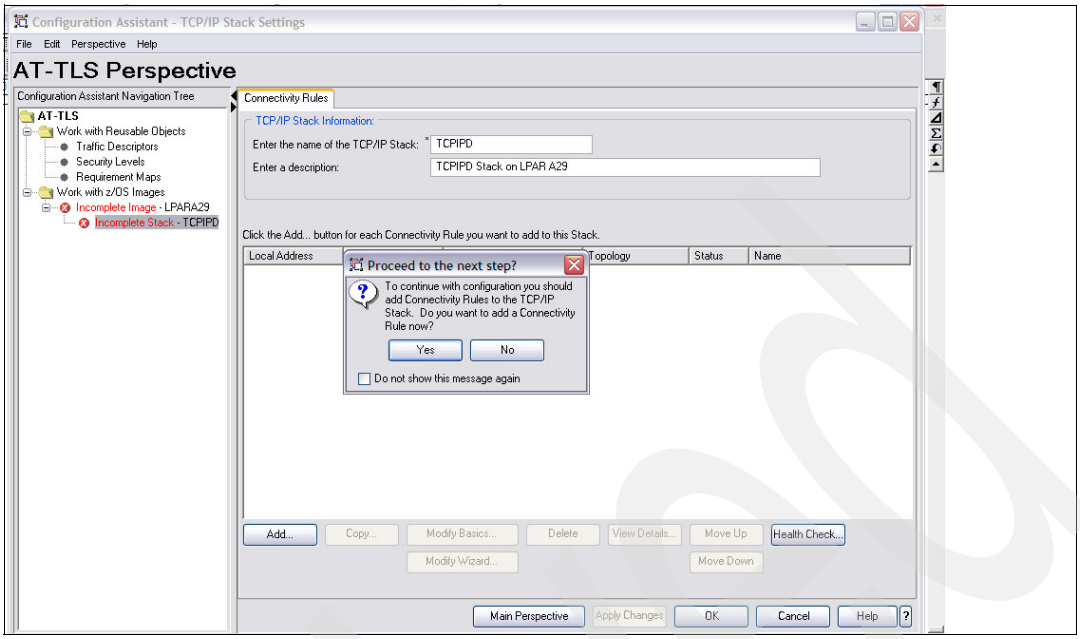


Figure 15-30 Incomplete Image and Stack

Note, in the left pane, that the z/OS image and the TCP/IP stack are listed as `Incomplete`. We need to define connectivity rules for them. Therefore, we select **Yes**, and then **Next** to start defining reusable objects that can be used as the basis for our FTP server connectivity rules. We define the data endpoints for the connectivity rule first. We have selected the IP address settings for both endpoints in Figure 15-31.

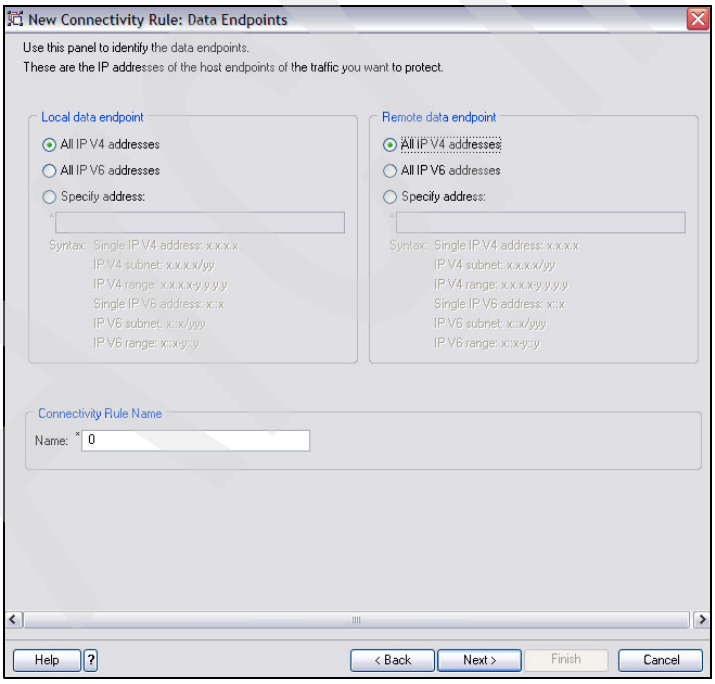


Figure 15-31 Connectivity rule: data endpoint definition

We decided to accept the Connectivity Rule name that is dynamically generated, so we did not fill in the field for rule name in Figure 15-31. (We eventually decided this was a mistake and had to change the rule name. We demonstrate how to do this at a later point in this

scenario.) We pressed **Next** to specify what the Requirement Map is for the new Connectivity Rule, and were presented with the panel in Figure 15-32.

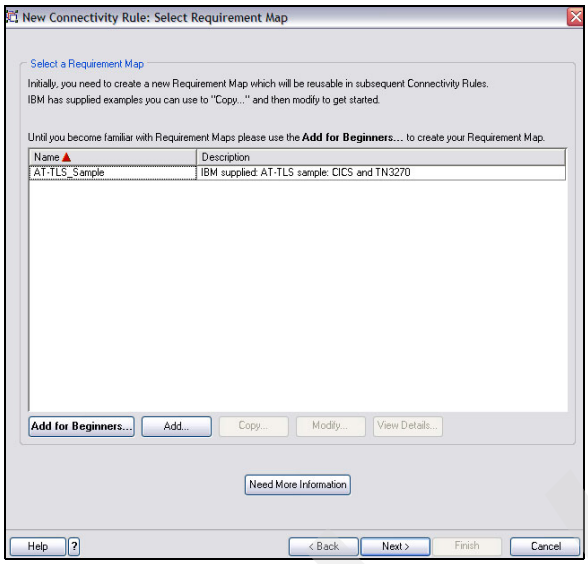


Figure 15-32 Selecting a Requirement map

On this panel we selected **Add for Beginners**. This took us to an introductory panel on defining requirement maps, where we press **Next**. The resulting panel is shown in Figure 15-33.

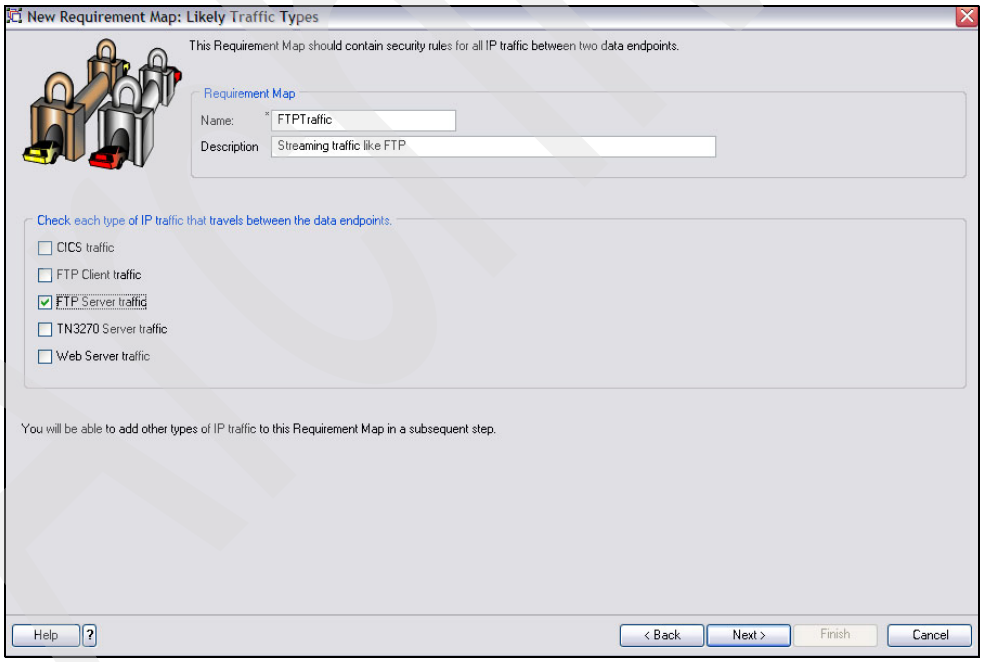


Figure 15-33 Requirement map for FTP server traffic

We clicked **Next** to take us to the panel shown in Figure 15-35 on page 629, where we can define traffic descriptors for FTP traffic.

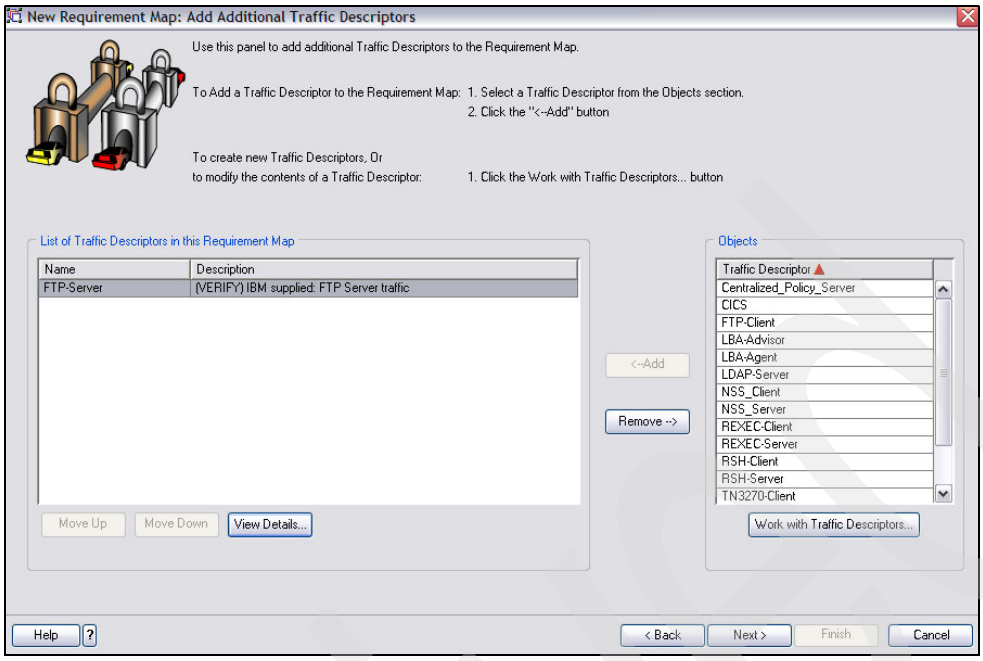


Figure 15-34 Requirement map definition

The FTP traffic descriptor has already been selected for us based upon our previous choices. However, we want to see the contents of the traffic descriptor, and so we click **View Details**, which presents the panel in Figure 15-35.

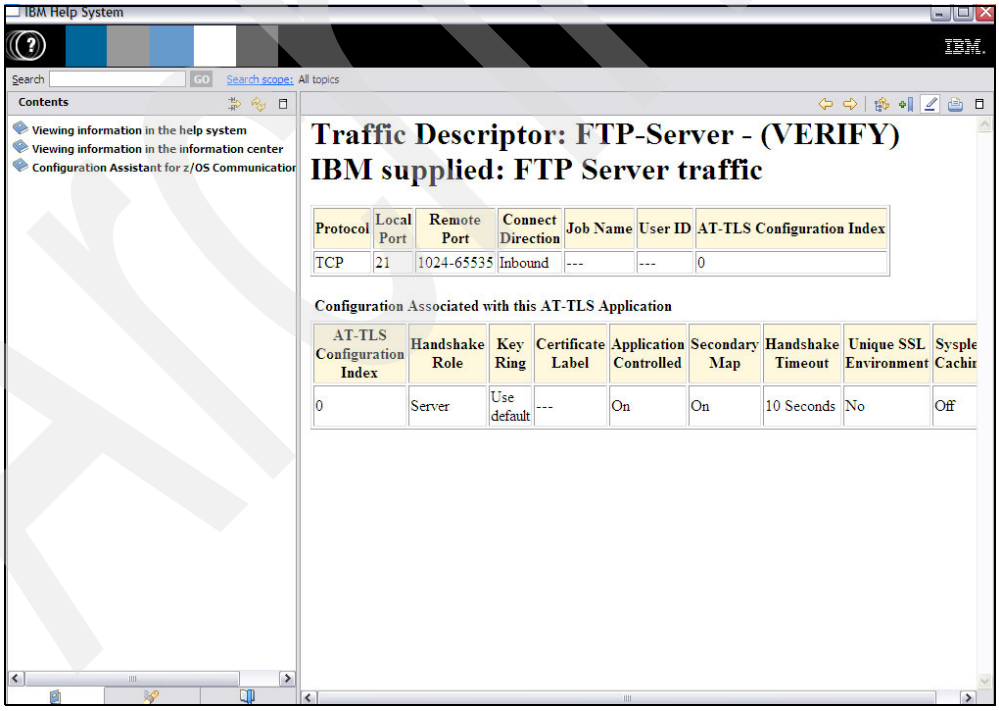


Figure 15-35 Traffic Descriptor for FTP Server

We close the information window, which takes us back to the traffic descriptor panel. When we select **Next**, this takes us to the security levels panel, where we highlight the FTP selection in order to assign a security level, shown in Figure 15-36 on page 630.

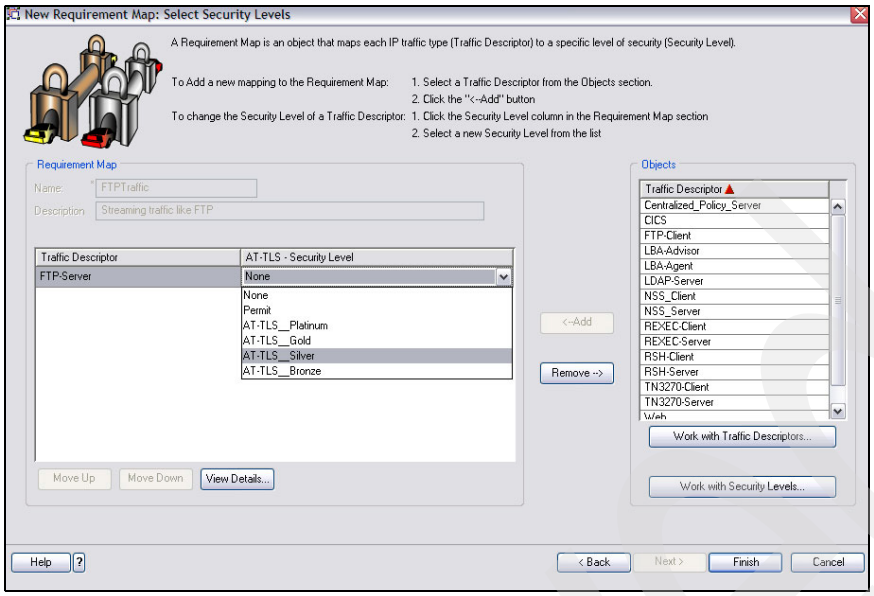


Figure 15-36 Selecting security levels

The pull-down on this panel permits us to highlight and select a Security Level. We chose the Silver level, and view the details first in Figure 15-37.

Selected Row

Traffic Descriptor	AT-TLS Security Level
FTP-Server - (VERIFY) IBM supplied: FTP Server traffic	AT-TLS__Silver - IBM supplied: Medium level of protection

Selected Row Details

Traffic Descriptor					AT-TLS Security Level	
Name	Protocol	Local port	Remote Port	Connect Direction	Name	Ciphers
FTP-Server	TCP	21	1024-65535	Inbound	AT-TLS__Silver	TLS V1 / SSL V3: 0x09 -TLS_RSA_WITH_DES_CBC_SHA 0x0A - TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F - TLS_RSA_WITH_AES_128_CBC_SHA

Figure 15-37 Viewing a Security Level definition

For our example we were satisfied with the Silver Level of encryption. However, you should consider creating your own Security Levels to reflect what is realistic for your environment.

Recommendation: Build your own Security Level with IBM Configuration Assistant. Keep in mind that there are export restrictions on certain encryption algorithms. Furthermore, there are performance issues with some encryption algorithms, depending on the type of cryptographic hardware available on the platform you are working with.

Therefore, you may want to resequence the Security Level contents and add or subtract some of the encryption levels in the IBM-supplied Security Levels to accommodate your particular situation. The IBM Configuration Assistant allows you to create new Security Levels for your own purposes.

Select **Finish** on the Requirement Map panel and **Proceed** when you see the Requirement Map Tip that a requirement map should contain more than one traffic descriptor. You can always add other traffic descriptors later.

Your new Requirement Map is built, as shown in Figure 15-38.

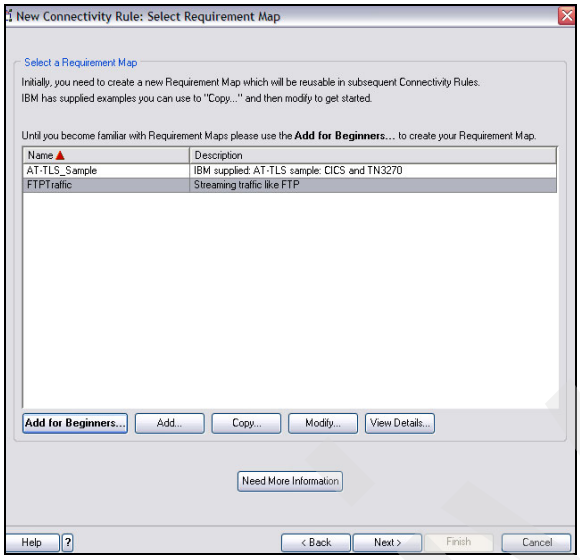


Figure 15-38 Requirement Map

Highlight the Requirement Map if it is not already highlighted and press **Next** to proceed to the next task, shown in Figure 15-39.

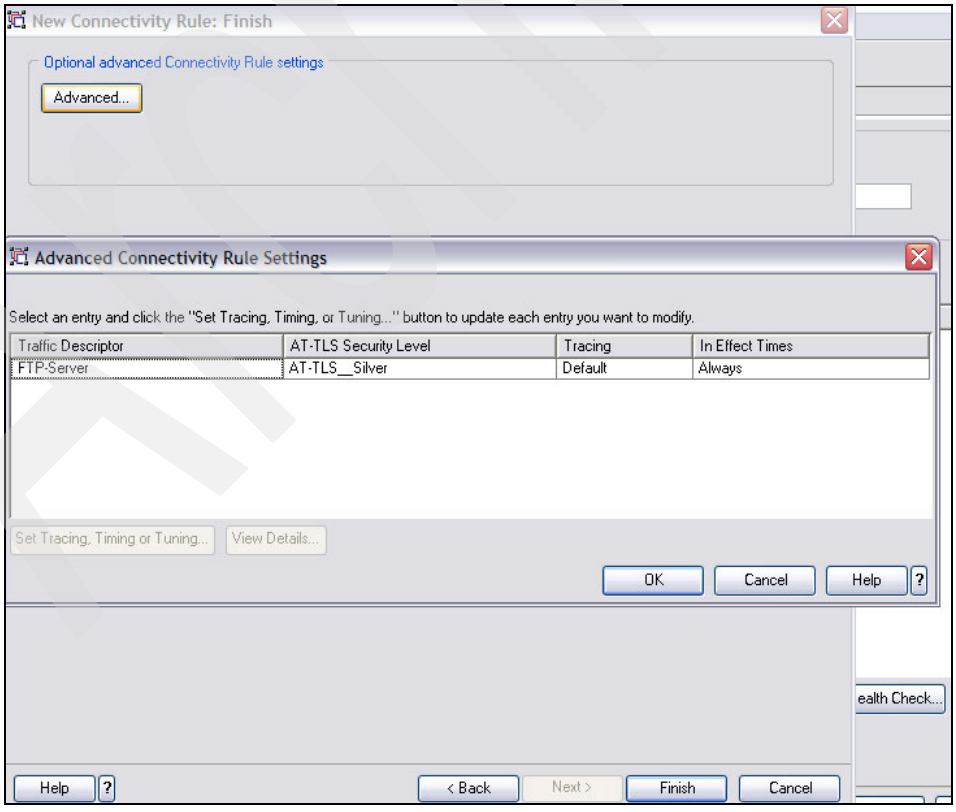


Figure 15-39 Connectivity Rule: Finish

You may press the **Advanced...** button to take you to tracing, timing and tuning options. Or you can select Finish, as we did. In Figure 15-40 you now see a new Connectivity Rule.

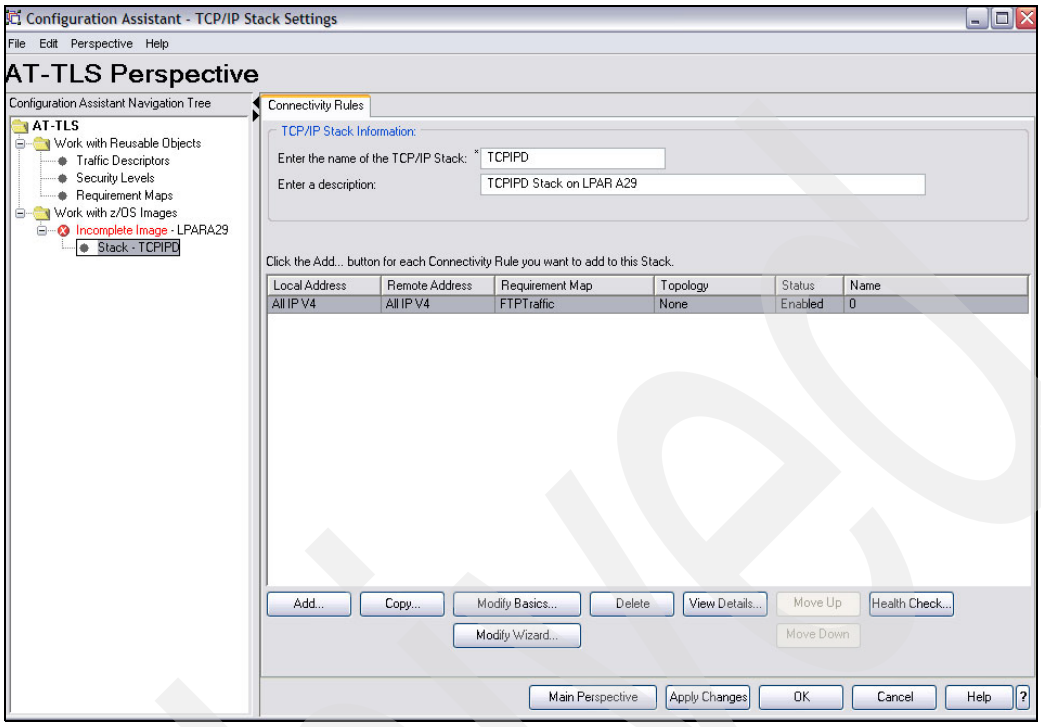


Figure 15-40 TCP/IP Stack Settings

Run **Health Check...** from this panel and verify that there are no mistakes or inconsistencies in the definitions. If there are, correct them. Then click **Apply Changes**. Note, however, that the z/OS image in the left-hand pane is `Incomplete`. Highlight the image and a new panel appears, as shown in Figure 15-41.

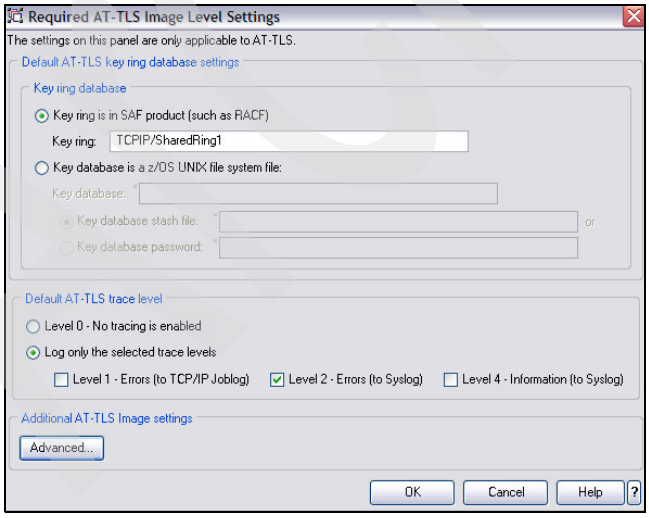


Figure 15-41 Specifying the key ring name and logging level for AT-TLS

We enter the label of the SAF shared key ring at TCPIPD. We include the owner (TCPIP) of the key ring in front of the forward slash (/). We keep the default log levels for now and click **OK**.

When we are returned to the AT-TLS perspective panel (), we note in the left pane that the z/OS image is now complete and that we can install the configuration files.

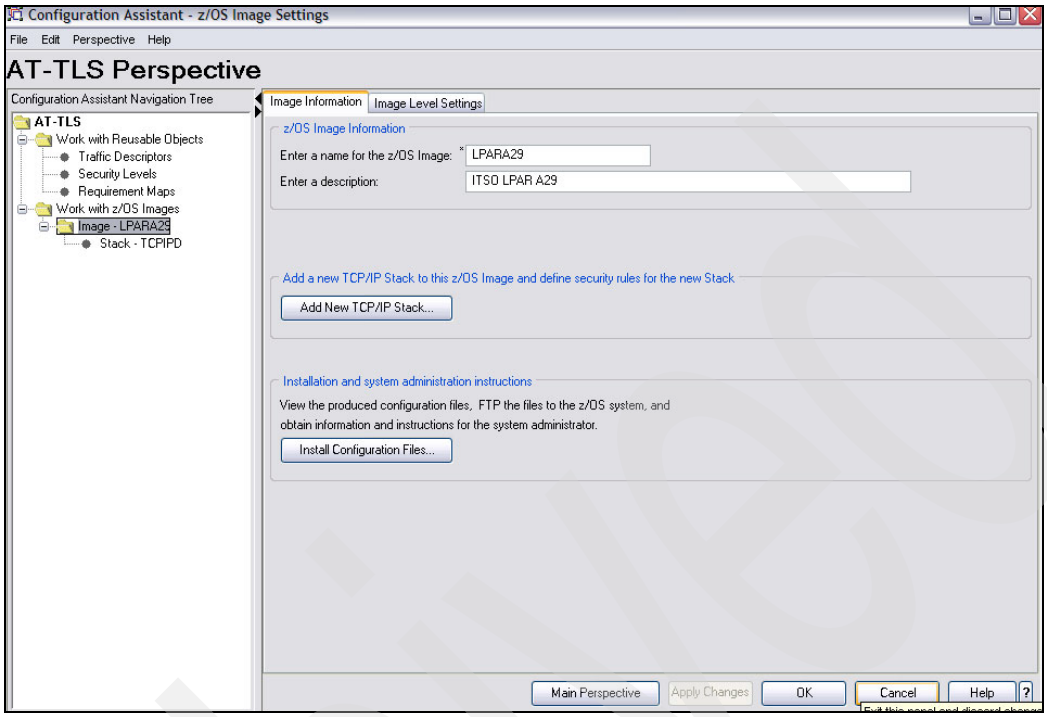


Figure 15-42 z/OS Image Settings

However, before we install the files, we want to share information about creating an FTP policy rule. There is more than one way to create a policy rule, and up until now we have shown you the way for the FTP server.

Behind the GUI panels, however, important details are inserted into the traffic descriptor for the FTP server. Therefore, select **Traffic Descriptor** from the panel you see in Figure 15-42. You are presented with Figure 15-43 on page 634.

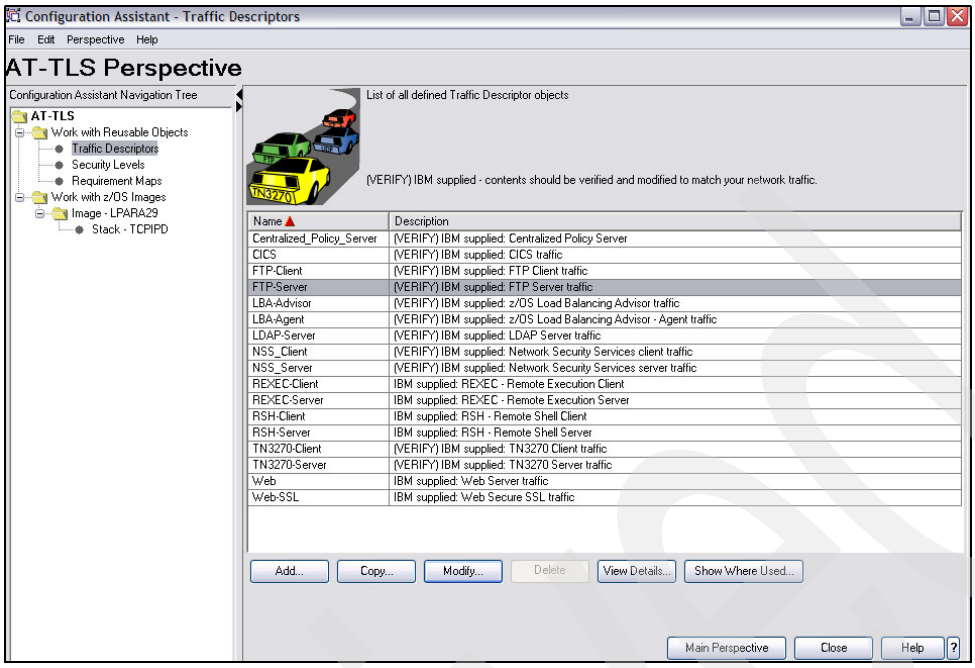
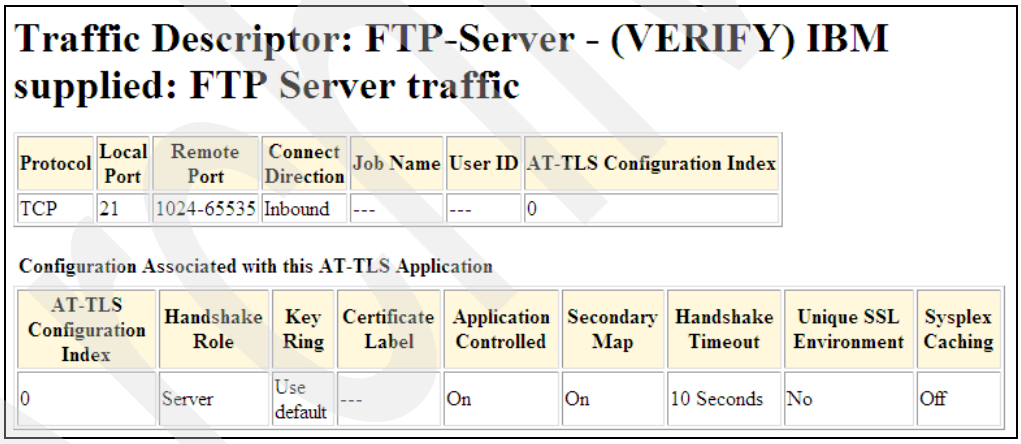


Figure 15-43 Traffic Descriptors

Highlight the FTP Server line as depicted and click **View Details**.



Traffic Descriptor: FTP-Server - (VERIFY) IBM supplied: FTP Server traffic

Protocol	Local Port	Remote Port	Connect Direction	Job Name	User ID	AT-TLS Configuration Index
TCP	21	1024-65535	Inbound	---	---	0

Configuration Associated with this AT-TLS Application

AT-TLS Configuration Index	Handshake Role	Key Ring	Certificate Label	Application Controlled	Secondary Map	Handshake Timeout	Unique SSL Environment	Sysplex Caching
0	Server	Use default	---	On	On	10 Seconds	No	Off

Figure 15-44 Viewing a Traffic Descriptor for the FTP server

Note that our configuration specifies that we are to use the default certificate in the key ring file. If we have decided to exploit the full functionality of AT-TLS, we may choose at some point to populate a key ring with many certificates and then to provide a certificate label for a certificate that is not the default.

We see that the FTP server is set to “Application Controlled”. You may recall from a description of AT-TLS for FTP that this server is indeed an application-controlled server for AT-TLS.

“Secondary Map” has been enabled for the FTP server; this is required because a data connection must be associated with the control connection. With Secondary Map On, the data connections have the same policy as the control connection. In this case, it is unnecessary to code any additional TTLSRule statement for the data connection because the FTP protocol itself negotiates the security level for the data connection.

Also notice that the SSL handshake time-out is set to 10 seconds. This is quite different from the SSL time-out default of 100 seconds for standard FTP TLS.

Important: If you are migrating from native TLS to AT-TLS, you probably want to consider changing the TLSTIMEOUT value to approximate what it was under native TLS to avoid any surprises. At the very least, be aware of the situation in order to establish appropriate TLSTIMEOUT values for your installation.

Next, we demonstrate how to modify the FTP traffic descriptor.

Modifying a traffic descriptor

On the panel shown in Figure 15-44 on page 634. highlight the FTP Server traffic descriptor and click **Modify**. On the next panel, also press **Modify**. This takes you to the panel depicted in Figure 15-45.

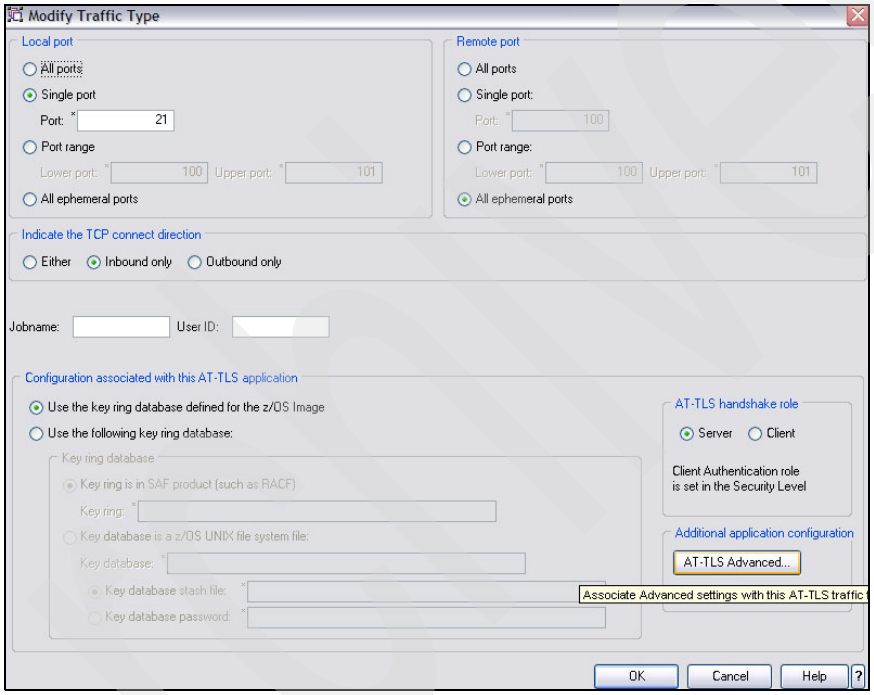


Figure 15-45 Modifying the Traffic Type

Click **AT-TLS Advanced...** to reach the panel shown in Figure 15-46 on page 636.

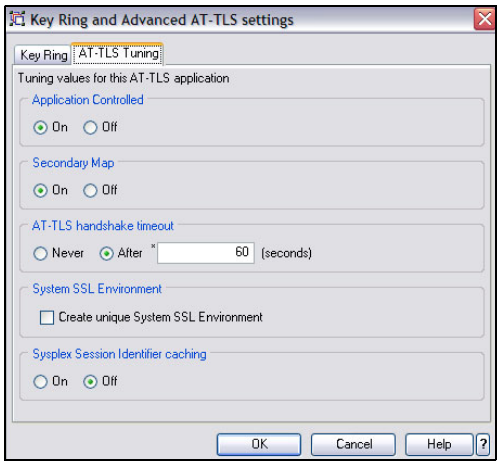


Figure 15-46 Advanced key ring settings

You are asked to specify a key ring label on one tab, or to specify AT-TLS tuning on the other. We bypass showing you the key ring label panel here because we do not have to specify a label. (Our key ring has only the Certificate Authority certificate and the default server certificate.)

Therefore, choose **AT-TLS tuning** on this panel. In Figure 15-46 we have overridden the AT-TLS default of 10 to specify the AT-TLS maximum of 60. This is not as much as we allowed for native TLS for FTP (100 seconds), but it is still a sizeable number.

Focus on the settings for Application Controlled and for Secondary Map. These settings are correct for FTP, as indicated in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Click **OK**. Continue clicking **OK** or **Proceed** until you arrive back at the AT-TLS Perspective panel of the GUI. In our case, we decided at this point that we did not like the default rule name for our connectivity rule. Therefore, we pressed the button to Modify Basics and we entered a rule name of FTPTLSRule. When we returned to the AT-TLS Perspective panel, we saw that we now had a suitable rule name, as shown in Figure 15-47 on page 637.

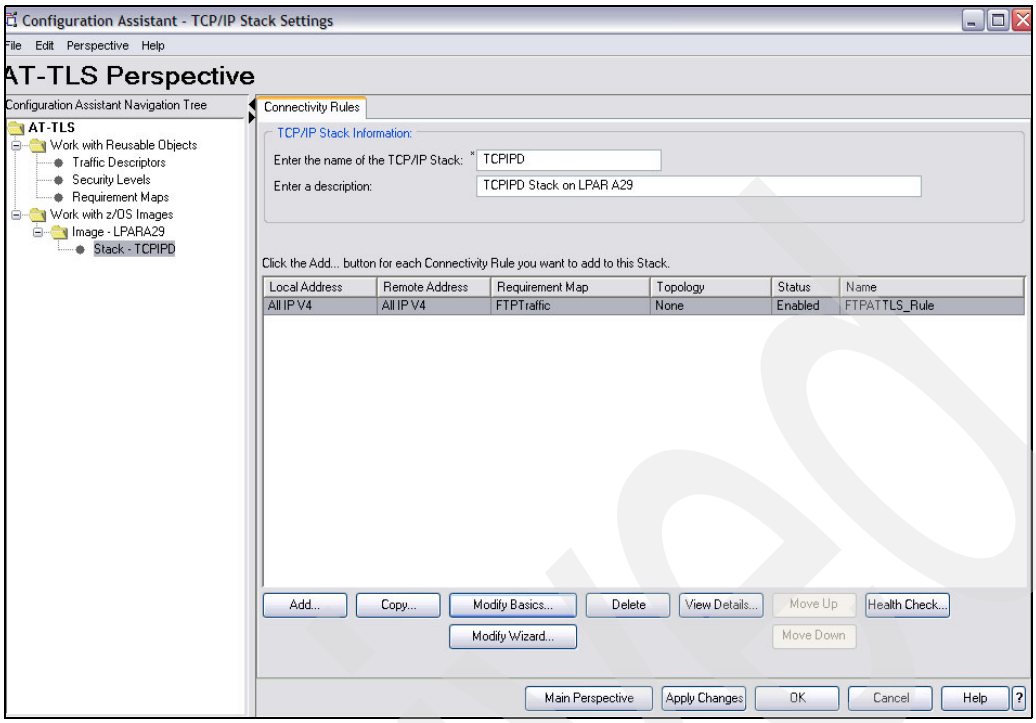


Figure 15-47 Completed AT-TLS definitions

Click **Apply Changes** as shown in Figure 15-47 and click **Health Check...** again. The file that is saved is comprised of Java objects. Its technical name is “the backing store file”. Before file transfer to the mainframe target, the policy files are converted to ASCII text files, but you also have the opportunity to save the policy files as ASCII to your workstation.

In our case, we were ready to send the policy file to the TCP/IP target. We closed the health check window and highlighted the z/OS image in the left pane so that we could begin the installation of the created policy file on the mainframe target.

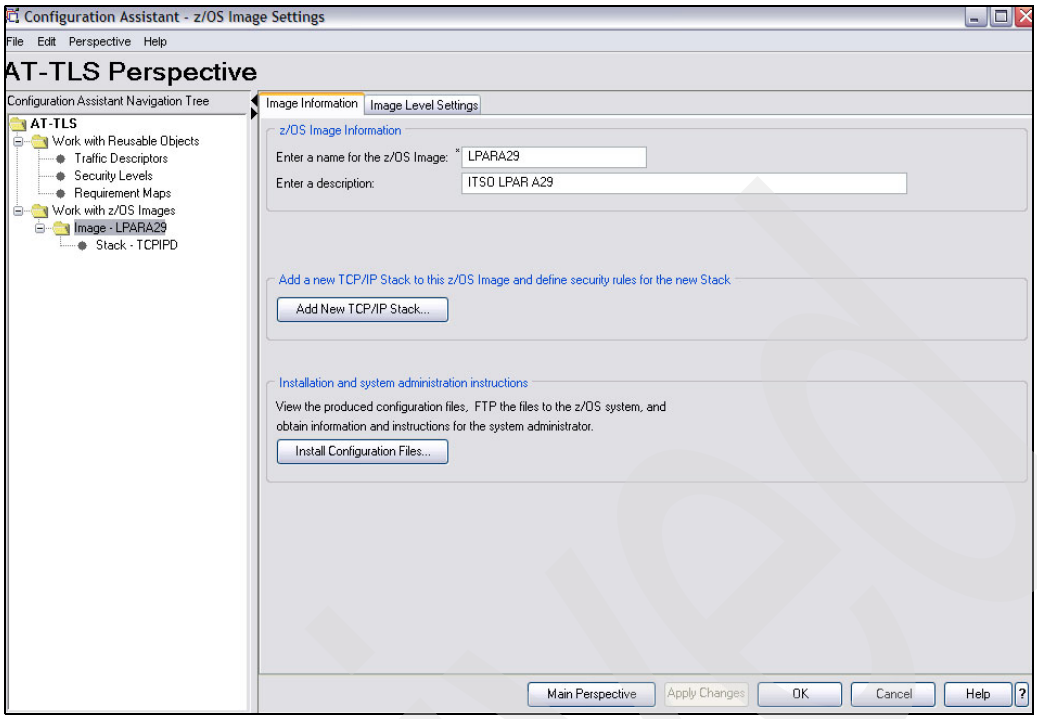


Figure 15-48 AT-TLS Main Perspective panel

Click **Install Configuration Files**. From this panel, you can look at the generated configuration file that you want to FP into LPAR A29. You can also look at the System Administration information that tells you about the RACF authorizations and TCP/IP stack changes that you must make to enable AT-TLS. For more detailed information about this topic, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

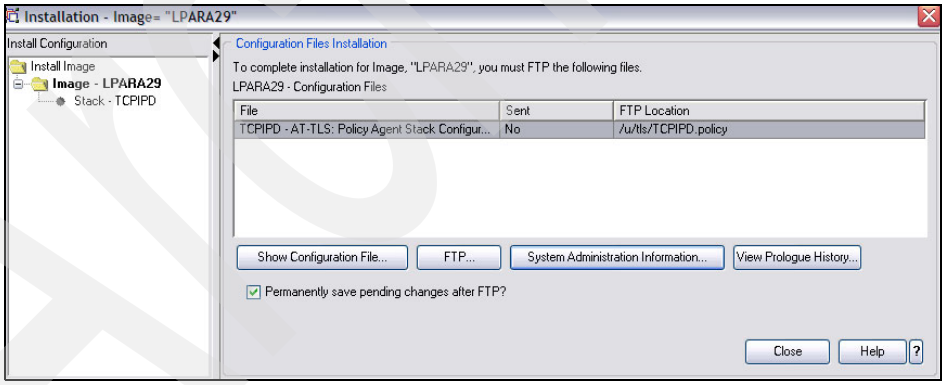


Figure 15-49 FTP preparation for sending configuration file to mainframe

The next display, shown in Figure 15-50 on page 639, contains a view of the generated configuration file and shows that you can even save it in ASCII to your workstation disk.

Figure 15-50 Generated AT-TLS policy file

We closed this view of the configuration file and chose **FTP** from the panel depicted in Figure 15-50. This brought us to Figure 15-51.

E

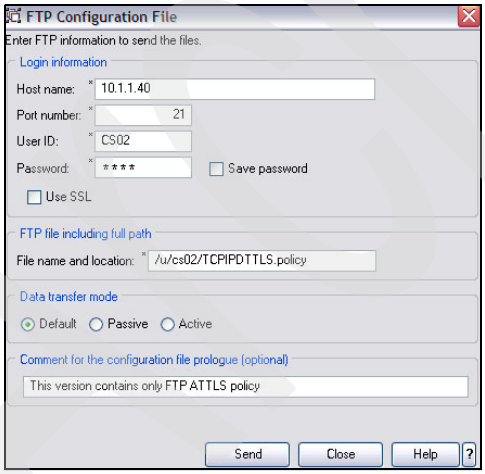


Figure 15-51 FTP Configuration File panel

We filled in the fields on the panel to point to our LPAR and its TCPIPD stack. We overrode the default directory to place the configuration file in /u/cs02/TCPIPDTTLS.policy. We entered a comment to identify this transfer setup. We then pressed **Send** and were soon notified that the transfer was successful. (Obviously we had an FTP server running in the TCPIP stack.)

We closed the window and skipped the opportunity to enter a history comment. We then closed the remaining window and the GUI itself by selecting **Exit** from the File pull-down.

Update the TCP/IP stack for AT-TLS

This section discusses the setup tasks to be performed in preparing the TCP/IP stack for AT-TLS support.

TCP/IP stack initialization access control

When AT-TLS is enabled and before Policy Agent itself initializes, there is a window of time in which applications or users could open sockets on the stack without being subjected to any kind of security controls. For this reason, there is a RACF resource called EZB.INITSTACK.sysname.tcpname that can be defined to limit access to the stack until the Policy Agent has finished initializing.

Important: User IDs that are permitted to this resource may open sockets prior to PAGENT initialization; those not permitted to this resource experience failures until PAGENT finishes initializing.

The INITSTACK SAF profile definition and sample permission statement are included in the hlq.SEZAINST(EZARACF) member and are illustrated in Example 15-41.

Example 15-41 Sample RACF INITSTACK permissions

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.sysname.tcpprocname UACC(NONE)
PERMIT   EZB.INITSTACK.sysname.tcpprocname -
         CLASS(SERVAUTH) ID(*) ACCESS(READ) -
         WHEN(PROGRAM(PAGENT,EZAPAGEN)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

We defined this profile for all the system names and TCP/IP stacks in our installation.

Important: User IDs who are not permitted to this resource will not be able to open sockets on this stack until the TTLS policy is installed. When the resource is not defined, no stack access is permitted until Policy Agent finishes initializing.

In Example 15-42 you see the details for adding authorization for two of the TCP/IP stacks in the installation: for TCPIPD **a** in LPAR A29 (SC33), and for TCPIPB **b** in LPAR A24 (SC31).

Example 15-42 Set up TTLS stack initialization access control for SC31 and SC33

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
RDEFINE  SERVAUTH EZB.INITSTACK.SC33.TCPIPD UACC(NONE) a
PERMIT   EZB.INITSTACK.SC33.TCPIPD CLASS(SERVAUTH) ID(*) c -
          ACCESS(READ) WHEN(PROGRAM(PAGENT,EZAPAGEN))
RDEFINE  SERVAUTH EZB.INITSTACK.SC31.TCPIPB UACC(NONE) b
PERMIT   EZB.INITSTACK.SC31.TCPIPB CLASS(SERVAUTH) ID(*) c -
          ACCESS(READ) WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

We permitted access for any user IDs that might be associated with the running program PAGENT, also known as EZAPAGEN **c**. The process of establishing stack initialization access control is described in hlq.SEZAINST(EZARACF) and in Chapter 11, “Application Transparent Transport Layer Security” on page 443.

Note: When we brought up the stacks, we noticed that it might be useful to authorize more than what is contained in the EZARACF member. Refer to 15.4.3, “Activation and verification of FTP AT-TLS support” on page 641 for more information about this topic.

TCP/IP stack changes

To activate AT-TLS, the TTLS parameter must be added to the TCPCONFIG profile config statement, as shown in Example 15-43.

Example 15-43 Profile statement to enable AT-TLS

```
TCPCONFIG TTLS
```

When AT-TLS is enabled in this fashion and a newly established connection is first used, the TCP layer of the stack searches for a matching AT-TLS policy installed from the Policy Agent. If no policy is found, the connection is made without the AT-TLS involvement.

Important: Do not insert this change into the TCP/IP profile before you have made the necessary RACF authorizations for INITSTACK processing. Otherwise, until PAGENT has fully initialized, any servers or users that attempt to open sockets on the TCP/IP stack will be denied access unless their associated user IDs are authorized to the INITSTACK SERVAUTH profile.

15.4.3 Activation and verification of FTP AT-TLS support

This section discusses the activation and startup validation of the FTP AT-TLS environment. The topics are:

- Starting TCP/IP enabled for AT-TLS
- Initializing PAGENT with AT-TLS policies
- Analyzing the FTPD daemon records in syslogd log

Starting TCP/IP enabled for AT-TLS

First we start the TCPIPD stack that has been enabled for AT-TLS. We started TCPIPD in LPAR A29 and received the messages shown in Example 15-44.

Example 15-44 Sample symptoms when SAF INITSTACK permissions have not been set

```
S TCPIPD
$HASP100 TCPIPD   ON STCINRDR
IEF695I START TCPIPD   WITH JOBNAME TCPIPD   IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
.................
*EZZ4248E TCPIPD WAITING FOR PAGENT TTLS POLICY d
 EZZ4202I Z/OS UNIX - TCP/IP CONNECTION ESTABLISHED FOR TCPIPD
 BPXF206I ROUTING INFORMATION FOR TRANSPORT DRIVER TCPIPD HAS BEEN
 INITIALIZED OR UPDATED.
 EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
 EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPD ARE AVAILABLE.
S OMPD
$HASP100 OMPD     ON STCINRDR
IEF695I START OMPD     WITH JOBNAME OMPD     IS ASSIGNED TO USER
```

```
TCPIP   , GROUP TCPGRP
$HASP373 OMPD     STARTED
ICH408I USER(OMVSKERN) GROUP(OMVSGRP ) NAME(####################) 581
  EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) e
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
EZZ7800I OMPD STARTING
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(####################) 585
  EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) e
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
EZZ7814I UNABLE TO CREATE SOCKET TYPE 1, ERRNO=112:EDC5112I RESOURCE
TEMPORARILY UNAVAILABLE., ERRNO2=74580296
EZZ8074I OMPD PROCESSING ERROR
EZZ7805I OMPD EXITING ABNORMALLY - RC(11) e
$HASP395 OMPD     ENDED
 ICH408I USER(IBMUSER ) GROUP(SYS1    ) NAME(                    ) 268
   EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) e
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
 HZS1002E CHECK(IBMCS,CSTCP_SYSPLEXMON_RECOV_TCPIPD): 269
 AN ERROR OCCURRED, DIAG: 0000007A_12B20381
 ICH408I USER(IBMUSER ) GROUP(SYS1    ) NAME(                    ) 270
   EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) e
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

We see in message EZZ4248E that TCPIPD is waiting for the PAGENT TTLS policy to initialize; this message d tells us that there is a window of time when there is no AT-TLS policy in effect that could represent a security exposure. And then the messages show us that the z/OS Health Checker, associated with the user ID of IBMUSER, could not run because it was not authorized to the profile for EZB.INITSTACK.SC33.TCPIPD. OMPRoute could not run at all either. Refer to all messages denoted with e.

We started up TN3270 so that we could access the system to determine what was wrong, and saw the messages displayed in Example 15-45.

Example 15-45 TN3270 server startup when Policy Agent not completely initialized

```
S TN3270D
$HASP100 TN3270D  ON STCINRDR
IEF695I START TN3270D  WITH JOBNAME TN3270D  IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270D  STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 356
          TCPIPD.TCPPARMS(TN3270D)
EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 357
          TCPIPD.TCPPARMS(TN3270D)
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(####################) 358
  EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) f
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(####################) 359
  EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) g
   INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
EZZ6011I TELNET BPX1SOC FAILED, RC = 00000070 RSN = 74580296
EZZ6035I TELNET DEBUG DETAIL PORT 361
```

```
  CONN: 00000000  LU: TELNET    MOD: EZBTTINI h
  RCODE: 102C-00  Socket initialization failed.  Will retry.
  PARM1: 00000000 PARM2: 00000000 PARM3: 00000000
EZZ6047I TELNET WAITING FOR TCPIPD ON PORT    23 i
```

In Example 15-45 on page 642 in the messages marked with **f**, **g**, **h**, and **i**, we see that the TN3270 server cannot open a socket on the stack while it is waiting for PAGENT to be started and to complete its initialization.

This is not necessarily a problem. After we bring up PAGENT with the AT-TLS policy, the messages will stop and the failed processes that are waiting will continue to initialize. The processes that terminated, like OMPROUTE (OMPD), can be started again without difficulty. In fact, as soon as we brought up PAGENT with the AT-TLS policy, TN3270D completed and was ready to accept connections. We restarted OMPROUTE and it came up successfully. The next time that z/OS Health Checker ran, it also completed successfully.

However, if you want to avoid some of these error messages to begin with, consider adding some RACF definitions, at least for OMPROUTE and the SNMP subagent. Up until now our RACF definitions from hlq.SEZAINST(EZARACF) have only authorized user IDs associated with the PAGENT program. If you decide you want to authorize the other programs, you can do so by authorizing their user IDs to the SERVAUTH profile as depicted in Example 15-46.

Example 15-46 Stack initialization access for OMPROUTE, TN3270, SNMP

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
PERMIT  EZB.INITSTACK.SC33.TCPIPD CLASS(SERVAUTH) ID(OMVSKERN,IBMUSER,TCPIP) c -
          ACCESS(READ)) a
PERMIT  EZB.INITSTACK.SC31.TCPIPB CLASS(SERVAUTH) ID(OMVSKERN,IBMUSER,TCPIP) c -
          ACCESS(READ)) a
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

In our example, the processes we care about are associated with the user IDs OMVSKERN, IBMUSER, and TCPIP. Note how we omitted the WHENPROGRAM specification **a** on the RACF PERMIT command.

When we stopped the running processes, including TCP/IP, and brought up the TCPIPD stack again, everything that was authorized started even prior to the initialization of PAGENT.

Initializing PAGENT with AT-TLS policies

The PAGENT startup messages in Example 15-47 show that additional policies have been loaded, and not just our AT-TLS policies.

Example 15-47 PAGENT initialization loads the policies

```
S PAGENT
. . . .
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
IAZ0511I NETSRV1 Server Port number could not be resolved, DEFAULT
assumed: 175
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : QOS
EZD1133I IKE STATUS FOR STACK TCPIPD   IS ACTIVE WITHOUT IPSECURITY
SUPPORT
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS
```

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : IPSEC
EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER
```

We want to examine whether the AT-TLS policies that loaded are really ours. We move to the UNIX System Services environment using the SDSF panel with the following command:

```
tso omvs
```

We want to execute the **pasearch** command, which has already been authorized to us by RACF. From the UNIX shell we entered the command with the **-t** option in order to examine only the AT-TLS policies. We redirected the output to a file in our user directory.

```
pasearch -t > myttlspolicies
```

Example 15-48 Executing pasearch -t from UNIX System Services

```
. . .
CS02 @ SC33:/u/cs02>pasearch -t > myttlspolicies
CS02 @ SC33:/u/cs02>
```

The file to which we redirected the output from the pasearch command now shows the loaded AT-TLS policies. They are indeed the ones we defined with IBM Configuration Assistant, as shown in Example 15-49.

Example 15-49 pasearch output from UNIX System Services

```
BROWSE    /u/cs02/myttlspolicies                  Line 00000000 Col 001
. . .
******************************** Top of Data ******************************
TCP/IP pasearch CS                    Image Name: TCPIPD
  Date:                 09/28/2007    Time:  12:15:04
  TTLS Instance Id:     1190995000

policyRule:             FTPATTLS_Rule~1   a
  Rule Type:            TTLS
  Version:              3               Status:           Active
  Weight:               255             ForLoadDist:      False
  Priority:             255             Sequence Actions: Don't Care
  No. Policy Action:    3  b
  policyAction:         enableGrpAct~LPARA29  c
   ActionType:          TTLS Group
Action Sequence:     0
 policyAction:          eAct1~FTP-Server      d
  ActionType:           TTLS Environment
  Action Sequence:      0
 policyAction:          cAct1~FTP-Server      e
  ActionType:           TTLS Connection
  Action Sequence:      0
 Time Periods:
  Day of Month Mask:
  First to Last:        11111111111111111111111111111111
  Last to First:        11111111111111111111111111111111
  Month of Yr Mask:     111111111111
  Day of Week Mask:     1111111  (Sunday - Saturday)
  Day of Week Mask:     1111111  (Sunday - Saturday)
  Start Date Time:      None
  End Date Time:        None
  Fr TimeOfDay:         00:00             To TimeOfDay:      24:00
  Fr TimeOfDay UTC:     04:00             To TimeOfDay UTC:  04:00
  TimeZone:             Local
```

```
TTLS Condition Summary:                 NegativeIndicator: Off
 Local Address:
  FromAddr:           0.0.0.0
  Prefix:             0
 Remote Address:
  FromAddr:           0.0.0.0
  Prefix:             0
 LocalPortFrom:       21                 LocalPortTo:       21
 LocalPortFrom:       21                 LocalPortTo:       21
 RemotePortFrom:      1024               RemotePortTo:      65535
 JobName:                                user ID:
 ServiceDirection:    Inbound

TTLS Action:                  enableGrpAct~LPARA29    c
  Version:                    3
  Status:                     Active
  Scope:                      Group c1
  TTLSEnabled:                On
  CtraceClearText:            Off
  Trace:                      2
  TTLSGroupAdvancedParms:
   SecondaryMap:              Off
   SyslogFacility:            Daemon
   EnvFile:                   //'TCPIP.SC33.STDENV'

TTLS Action:                  eAct1~FTP-Server d
  Version:                    3
  Status:                     Active
  Scope:                      Environment d1
  HandshakeRole:              Server
  TTLSKeyringParms:
   Keyring:                   TCPIP/SharedRing1
  TTLSEnvironmentAdvancedParms:
   SSLv2:                     Off
   SSLv3:                     On
   TLSv1:                     On
   ApplicationControlled:     Off
   HandshakeTimeout:          10
   ClientAuthType:            Required
   ResetCipherTimer:          0
  EnvironmentUserInstance:    0

TTLS Action:                  cAct1~FTP-Server e
  Version:                    3
  Status:                     Active
  Scope:                      Connection e1
  HandshakeRole:              Server
  TTLSConnectionAdvancedParms:
   SecondaryMap:              On
   ApplicationControlled:     On
   HandshakeTimeout:          60
  TTLSCipherParms:
   v3CipherSuites:
    09  TLS_RSA_WITH_DES_CBC_SHA
    0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
    2F  TLS_RSA_WITH_AES_128_CBC_SHA
```

Notice how the **pasearch** formatting of policies is much more understandable than a review of the actual policy file. The reusable objects from the actual policy file that was produced now

appear inline with the policy itself when the **pasearch** command is executed. It is thus much easier to view the logic of the policy with the **pasearch** output.

The rule **a** references three actions **b**. The names of the actions on the rule at **c**, **d**, and **e** point to the action definitions themselves, also indicated with **c**, **d**, and **e** in Example 15-49 on page 644.

However, note that there is only one Group Action **c1**, although the other actions (Connection **d1** and Environment **e1** are, in fact, part of the Group Action. You see the entire contents of the rule and the entire contents of the actions.

There is another way to see the results of **pasearch** without entering the UNIX shell. A REXX exec that can be invoked from TSO or NetView to execute the **pasearch** command or any other UNIX command for you.

Example 15-50 REXX exec: from TSO issue "unixcmd pasearch"

```
/* REXX */
parse arg input
parse var input number mycmd
if datatype(number,'N') then
    maxlines = number
else do
    maxlines = 9999
    mycmd = input
end
if syscalls('ON') >3 then do
    say 'Unable to establish the SYSCALL environment'
return
end
environment.0 = 2;
environment.1 = "PATH=/bin:/usr/sbin"
environment.2 = "LIBPATH=/lib:/usr/lib"
call bpxwunix mycmd,,stdout.,stderr.,environment.
outlines = 0
if stdout.0 > 0 then do i=1 to stdout.0
    if i > maxlines then do
       x=maxlines
       parm='MAX output lines ('||x||') reached'
       parm=parm||' - report truncated'
       say parm
       leave
    end
    say stdout.i
    outlines = outlines + 1
end
if stderr.0 > 0 then do i=1 to stderr.0
   if i > maxlines-outlines then do
       x=maxlines
       parm='MAX output lines ('||x||') reached'
       parm=parm||' - report truncated'
       say parm
       leave
    end
    say stderr.i
end
exit 0
```

If you do not want to create a REXX exec to execute from TSO or from NetView, you can still learn about AT-TLS policy with the **netstat** command. In Example 15-51 we have executed the **netstat** command with the **ttls** option.

Example 15-51 Sample NETSTAT TTLS command output

```
D TCPIP,TCPIPD,N,TTLS
 EZD0101I NETSTAT CS V1R9 TCPIPD 430
 TTLSGRPACTION                          GROUP ID          CONNS
 ENABLEGRPACT~LPARA29                   00000001              0
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

You see the name of a Group Action that is in our file. There are other variations of the **netstat ttls** command that we executed after we started FTP and connected some sessions.

Analyzing the FTPD daemon records in syslogd log

Now that we have some FTP connections in place, let us view what the syslog daemon log tells us about the FTP startup and the TLS client's login.

The syslog daemon isolation that we set up for the FTP procedure named FTPDD collects messages in /tmp/syslog/ftpdd.log. Browsing this log from the UNIX shell, as shown in Example 15-52, we see that the FTP server was successfully set up with security and that the AUTH TLS commands are allowed to flow with AT-TLS enabled.

Example 15-52 Syslogd output for the FTP server and the client login

```
DM1115 main: Initialization parameter values:
DM1118 ..localsite values ------------------------
.........................................
DM1284 ..security mechanism values ---------------- a
DM1286 ....SECURE_FTP................A
DM1288 ....SECURE_LOGIN..............N
DM1290 ....SECURE_PASSWORD...........R
DM1292 ....SECURE_PASSWORD_KERBEROS..R
DM1294 ....SECURE_CTRLCONN...........P
DM1296 ....SECURE_DATACONN...........C
DM1298 ....SECURE_PBSZ...............16384
DM1300 ....KEYRING...................TCPIP/SharedRing1
DM1302 ....CIPHERSUITE...............0A090102
DM1304 ....TLSPORT...................990
DM1306 ....TLSTIMEOUT................100
DM1310 ....TLSRFCLEVEL...............RFC4217
DM1313 ....TLSMECHANISM..............ATTLS
DM1316 ....securePort................FALSE
.........................................
DM1507 main: daemon's code page is: IBM-1047
DM1510 main: daemon's locale is: C
DM1522 main: environment variable: TZ=EST5EDT4
DM1522 main: environment variable: _BPXK_SETIBMOPT_TRANSPORT=TCPIPD b
DM1522 main: environment variable: _EDC_ADD_ERRNO2=1
EZY2700I Using port FTP control (21)
.........................................
PR0306 parse_cmd: entered
PR0485 parse_cmd: >>> USER cs02
SR3042 reply: --> 331 Send password please.
PR0490 parse_cmd: >>> PASS ******
RA0625 RACF_MIXED_CASE_PASSWORDS_ENABLED 0
```

```
EZYFS50I ID=FTPDD100001 CONN   starts Client IPaddr=::ffff:10.1.4.21c
SD0764 setup_new_pgm: entered
```

Note that a in Example 15-52 on page 647 shows the security parameters set in the FTP.DATA file. At b, this FTP server has established stack affinity to the TCPIPD stack. And at ct, the client IP address is that of the TCPIPB TSO client at 10.1.4.21.

Examine the output from the **netstat ttls** command in Example 15-53; b shows that we have two connections active to AT-TLS.

Example 15-53 Netstat TTLS connections

```
D TCPIP,TCPIPD,N,TTLS
 EZD0101I NETSTAT CS V1R9 TCPIPD 430
 TTLSGRPACTION                          GROUP ID          CONNS
 ENABLEGRPACT~LPARA29                   00000001              2 b
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

In Example 15-54, we show you the detail on the Action Group that is active.

Example 15-54 Netstat TTLS Action Group information

```
D TCPIP,TCPIPD,N,TTLS,GROUP,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPD 441
TTLSGRPACTION:   ENABLEGRPACT~LPARA29
  GROUPID:         00000001
  TASKS:           4                  GROUPCONNS:      2
  WORKQELEMENTS:   0                  SYSLOGQELEMENTS: 0
    ENV: EACT1~FTP-SERVER                       ENVCONNS: 2
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

Example 15-55 displays the connection ID of an FTP connection using a standard **netstat conn** command; it is followed by the **netstat ttls** display for that particular connection.

Example 15-55 Specific connection information for an ATTLS connection

```
D TCPIP,TCPIPD,N,CONN,PORT=21
EZD0101I NETSTAT CS V1R9 TCPIPD 466
USER ID  CONN     STATE
FTPDD1   000000CA LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00000038B FINWT2
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.4.21..1028
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT

D TCPIP,TCPIPD,N,TTLS,CONN=38B,DETAIL
EZD0101I NETSTAT CS V1R9 TCPIPD 458
CONNID: 0000038B
  JOBNAME:      FTPDD1
  LOCALSOCKET:  ::FFFF:10.1.1.40..21
  REMOTESOCKET: ::FFFF:10.1.4.21..1028
  SECLEVEL:     TLS VERSION 1
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
```

```
TTLSRULE: FTPATTLS_RULE~1
  PRIORITY:       255
  LOCALADDR:      0.0.0.0/0
  LOCALPORT:      21
 LOCALPORT:      21
 REMOTEADDR:     0.0.0.0/0
 REMOTEPORTFROM: 1024               REMOTEPORTTO: 65535
 DIRECTION:      INBOUND
 TTLSGRPACTION:  ENABLEGRPACT~LPARA29
   GROUPID:                    00000001
   TTLSENABLED:                ON
   ENVFILE:                    //'TCPIP.SC33.STDENV'
   CTRACECLEARTEXT:            OFF
   TRACE:                      2
   SYSLOGFACILITY:             DAEMON
   SECONDARYMAP:               OFF
 TTLSENVACTION:  EACT1~FTP-SERVER
   ENVIRONMENTUSERINSTANCE:    0
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~FTP-SERVER
    HANDSHAKEROLE:              SERVER
    V3CIPHERSUITES:             09 TLS_RSA_WITH_DES_CBC_SHA
                                0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
    APPLICATIONCONTROLLED:      ON
    HANDSHAKETIMEOUT:           60
    SECONDARYMAP:               ON
1 OF 1 RECORDS DISPLAYED
```

15.5 Backing up the backing store file and policies

This section discusses the processes for managing the backing store files.

It is worthwhile to preserve the Java backing store files in a server that is backed up nightly. The backing store file is also FTPed to the mainframe every time you send a policy file there if you have performed the following steps:

- On IBM Configuration Assistant, set up the Preferences panel to store the file on the mainframe in either an HFS or zFS, or in a partitioned data set.
- When you execute the Install option on Configuration Assistant to FTP the policy file to the mainframe target, select the check box that allows you to automatically save the backing store after the FTP transfer completes.

If you save the backing store file on the mainframe, you can also use MVS methods of backing up or archiving the file. The ASCII policy files can also be backed up using standard dump and restore procedures.

Note: With this version of the IBM Configuration Assistant, it is not possible to convert the ASCII file to a backing store file. At this level of the IBM Configuration Assistant code, after you decide to start editing the ASCII policy files manually, you must continue to do so unless the changes have been minor, like changing the trace levels of the processes.

If different personnel are responsible for different types of policies, there is no problem in maintaining multiple backing store files, each named differently. If the decision is made to merge them, there is an IMPORT utility on the IBM Configuration Assistant which can be used for this task.

Ultimately, the point is to maintain backups of critical files, like the backing store file and the various configuration and policy files.

There is a Centralized Policy Server which can simplify policy management. Instead of using local policy files for each of multiple TCP/IP images, you can reduce the number of individual files needing to be backed up by storing the policy files themselves directly on a Centralized Policy Server.

The files backed up at the Policy Server represent the bulk of the files required for a PAGENT implementation, thus limiting the scope of the backup services required. The subject of Centralized (or Distributed) Policy Services is treated in Chapter 6, “Central Policy Server” on page 257.

15.6 Migrating from native FTP TLS to FTP AT-TLS

This section discusses migrating from the native FTP TLS environment to the FTP ATTLS environment.

15.6.1 Migrating policies to a new release of z/OS Communications Server

Migration can be accomplished by moving from either a flat policy file or a policy file generated by the Configuration Assistant GUI.

Migrating from an existing flat policy file

If you already have an existing flat policy file from a previous release, you can continue to use it. Simply add the new AT-TLS policy to the suite of policies already defined.

Migrating from an existing IBM Configuration Assistant policy file

If you already have an existing backing store policy file from a previous release, you can open the current IBM Configuration Assistant for z/OS Communications Server and, using the file pull-down, open the existing file. It will be upgraded to the new structure within the IBM Configuration Assistant. Then simply add the new AT-TLS policy to the suite of policies already defined. Refer to Chapter 5, “Policy Agent” on page 221, for more information about upgrading existing files.

15.6.2 Details on migrating from TLS to AT-TLS

Migrating from native FTP TLS to FTP with AT-TLS is simple. Consider the following items:

- What to do with the existing key rings and certificates during migration
- What to do with the FTP.DATA files during migration
- How to define AT-TLS instead of native TLS

- What new procedures are necessary for migration
- What changes to security are required for migration
- What changes to the TCP/IP stack are required for migration

What to do with the existing key rings and certificates during migration

If you already have a functioning FTP with TLS, you can continue to use the same key ring and certificates to implement FTP with AT-TLS.

What to do with the FTP.DATA files during migration

You can also continue to use the same server FTP.DATA and client FTP.DATA by simply changing or adding one parameter: TLSMECHANISM ATTLS. Three existing types of parameters in the TLS FTP.DATA can be moved into an AT-TLS policy file; if you choose not to remove them, they will be superseded by what is coded in the AT-TLS policy file anyway. Table 15-1 shows which statements must migrate to the policy agent file.

Table 15-1 Migrating to AT-TLS: Statements that move to the AT-TLS policy file

FTP.DATA Statement	AT-TLS Statement	Location of AT-TLS Policy Statement
Keyring	Keyring	TTLSEnvironmentAction -> TTLSKeyRingParms
CipherSuite	V3CipherSuite	TTLSEnvironmentAction -> TTLSCipherParms
TlsTimeout	GSK_V3_Session_Timeout	TTLSEnvironmentAction -> TTLSGskAdvancedParms

CipherSuite migration

Table 15-2 shows how the cipher specifications are slightly different if you choose to migrate them from your existing TLS FTP.DATA file into a TTLS policy file. If you edit the policy files manually, you need to know the AT-TLS values; if you use the IBM Configuration Assistant, the values are easier to select using the GUI.

Table 15-2 Cipher Suite names and values

Cipher Suite cipher	V3CipherSuite value	Hex value
SSL_DES_SHA	TLS_RSA_WITH_DES_CBC_SHA	09
SSL_3DES_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	0A
SSL_NULL_MD5	TLS_RSA_WITH_NULL_MD5	01
SSL_NULL_SHA	TLS_RSA_WITH_NULL_SHA	02
SSL_RC2_MD5_EX	TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	06
SSL_RC4_MD5	TLS_RSA_WITH_RC4_128_MD5	04
SSL_RC4_MD5_EX	TLS_RSA_EXPORT_WITH_RC4_40_MD5	03
SSL_AES_128_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	2F
SSL_AES_256_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	35

How to define AT-TLS instead of native TLS

Define the AT-TLS policy using the IBM Configuration Assistant for z/OS Communications Server. Then, by using the **pasearch** command or scanning the ASCII policy file, compare the output values from AT-TLS with what you were operating with under native TLS to ensure

that you have truly implemented the AT-TLS environment to resemble as closely as possible the old tuning and operating values.

What new procedures are necessary for migration

For AT-TLS, you need a functioning Policy Agent. Refer to Chapter 5, “Policy Agent” on page 221 for information about how to accomplish this task.

What changes to security are required for migration

The authorizations that are already in place for key ring access and certificate and key processing are still valid for AT-TLS. Review “Create key ring and certificates and update RACF permissions” on page 586 if you need more information.

Policy Agent requires additional RACF authorizations, as do the users who need access to the PAGENT commands like **`pasearch`**. Refer to Chapter 5, “Policy Agent” on page 221 for information about how to accomplish this task.

Special authorizations in the SERVAUTH facility class are required in order to enable the stack and its associated servers and users to access the stack after AT-TLS is enabled, but prior to completed PAGENT initialization. Therefore you need to add these authorizations as described in Chapter 11, “Application Transparent Transport Layer Security” on page 443, or consult “TCP/IP stack initialization access control” on page 640 for more information about this topic.

What changes to the TCP/IP stack are required for migration

To activate AT-TLS, the TTLS parameter must be added to the TCPCONFIG profile config statement. For more information, consult “TCP/IP stack changes” on page 641.

Important: Do not insert this change into the TCP/IP profile before you have made the necessary RACF authorizations for INITSTACK processing.

Otherwise, any servers or users that attempt to open sockets on the TCP/IP stack prior to the completion of PAGENT initialization will be denied access unless their associated user IDs are authorized to the INITSTACK SERVAUTH profiles.

15.7 FTP TLS and AT-TLS problem determination

The most important tool for problem determination is a running SYSLOG daemon, preferably with syslogd message isolation in effect. Without a SYSLOG daemon, messages are buried in console logs or sometimes not recorded at all, thus severely reducing your effectiveness in solving problems.

The second most important tool for problem determination is access to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782. It contains detailed chapters that include common problem diagnosis, steps for taking traces, and so on.

In case of a server problem, first check the console and syslogd for error messages. Consult *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783; *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785; and *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786, for more details and for references to alternate publications within the z/OS family.

Look for the error codes and return codes and consult the IP messages volumes mentioned here. You may also have to consult UNIX System Services publications and z/OS Integrated

Security Services publications for specialized error codes, like those for Language Environment and SSL.

Scan the syslogd files for OBJERR messages if PAGENT problems are occurring. OBJERR errors usually point to logic or syntax errors in policy files.

If no problem can be identified from messages, enable a trace in any of the following manners:

- Specify the keyword TRACE as a PARMS parameter on the FTPD catalogued procedure.
- Set trace or debug options in the appropriate FTP.DATA file.
- Issue a MODIFY <ftpprocname> command to enable trace or debug.
- Request a trace or debug at client connect time.
- Request a server trace or debug from within the client connection by entering the appropriate **SITE** command.

z/OS Communications Server: IP System Administrator's Commands, SC31-8781, and *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, explain how to work with traces for FTP. *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, includes a list of the TLS/SSL error codes that are also documented in the SSL Programmer's Guide. These error codes give discrete explanations of violations of SSL protocols, and are useful in determining what types of errors are occurring on the TLS/SSL connection.

An invaluable tool is "Diagnosing File Transfer Protocol (FTP) Problems" in *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

15.8 Additional information

For additional information about FTP security, refer to:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
  - FTP is defined by RFC 959, RFC 2228, Internet Draft RFC, "On Securing FTP with TLS (draft 05)", and RFC 4217.

Tip: For descriptions of security considerations that affect specific servers or components, refer to "UNIX System Services Security Considerations" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Part 5

Appendixes

Given that security technologies are complex and can be confusing, this part includes tutorial information that discusses the following topics.

Appendix	Topic
Appendix A, “Basic cryptography” on page 657	Security issues, secret keys and their algorithms, public and private keys and digital certificates, cryptosystems and performance, and message integrity
Appendix B, “Telnet security: advanced settings” on page 675	Implementation and verification of the advanced TN3270 server configuration using the native TLS function, and using the AT-TLS policy
Appendix C, “Configuring IPSec between z/OS and Windows” on page 711	Steps used for our IPSec scenario between z/OS and Windows (setting up: the IKE daemon; certificates; the z/OS IPSec policy; a Windows IPSec policy; and verification that things are working)
Appendix D, “zIIP Assisted IPSec” on page 741	Configuring zIIP Assisted IPSec, and an example of zIIP Assisted IPSec implementation
Appendix E, “Our implementation environment” on page 757	The complete environment used for the four *Communications Server for z/OS TCP/IP Implementation* books, and the environment used for this IBM Redbooks publication

A

Basic cryptography

Cryptography has more uses than ensuring privacy through encrypting a message. Other uses for cryptography are to provide message integrity through the use of encrypted message hashes, and non-repudiation so that a sender cannot deny having sent a particular message. To ensure privacy, integrity, and non-repudiation in non-secure networks, cryptographic procedures need to be used.

Today, two distinct classes of cryptographic algorithms are in use:

- Secret key (or symmetric key)
- Public key (or asymmetric key)

They are fundamentally different in how they work, and thus in where they are used. These are the basic building blocks for securing transactions over the Internet or some other untrusted network.

This appendix discusses the following topics.

Section	Topic
"Potential problems with electronic message exchange" on page 658	An example to illustrate the security issues with electronic message exchanges.
"Secret key cryptography" on page 661	The basic concepts surrounding secret keys and the algorithms used for those keys.
"Public key cryptography" on page 662	The basic concepts surrounding public keys where public and private keys are used Digital certificates and their role in the secure use of public key cryptography.
"Performance issues of cryptosystems" on page 669	Performance concerns when using cryptography
"Message integrity" on page 669	How cryptography can aid in asserting message integrity (ensuring a message has not been altered in transit). How digital signatures can prove that the message sender actually sent the message

Cryptography background

The word *cryptography* has its roots in Greek, and it means “secret writing”. One of the earliest uses of cryptography was for protecting military communications. In ancient times, a human messenger would be dispatched with a military order. If that messenger was caught, the message could be read by the enemy. A method had to be used to hide the meaning of the message from an interceptor, but still allow the intended recipient to understand it.

The message (plain text) to be conveyed has to be encrypted by some mathematical formula (the *cipher*). The cipher normally has, as its inputs, the message to be encrypted and a *key*. By using a key, the cipher itself can be public knowledge but the key is kept private between the communicating parties. The text that is produced by the cipher is the *ciphertext*. The decryption process takes the ciphertext, runs it through the decryption cipher with the key, and produces the plain text again.

Potential problems with electronic message exchange

Let us take an example of an electronic message exchange for a stock broker. Clients log on to the system and send electronic buy or sell requests to the broker. Potential security problems involved with these message exchanges include:

- “The request is not really from your client”.
- “The order could have been intercepted and read”.
- “The order could have been intercepted and altered”.
- “An order is received from your client, but he denies sending it”.

In the following sections we discuss these problems and show what can be done to resolve each of them.

The request is not really from your client

Figure A-1 shows a hacker posing as a legitimate client (Garth).

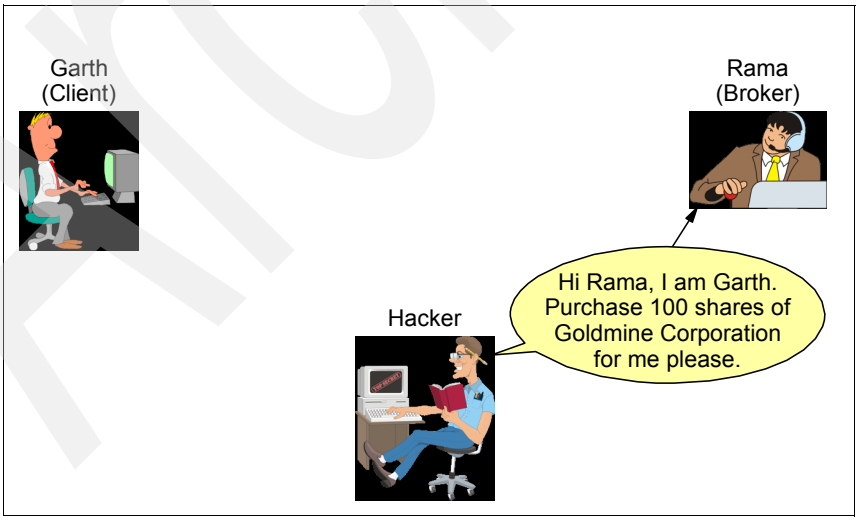


Figure A-1 Hacker posing as a genuine client

What is needed here is some way to ensure that the client is who it says it is. In some cases, this must involve some sort of shared secret, such as a password. This is called *user authentication*. “Authentication” on page 663 explains how this is done.

The order could have been intercepted and read

Figure A-2 shows a hacker intercepting and reading an order that the client has placed.

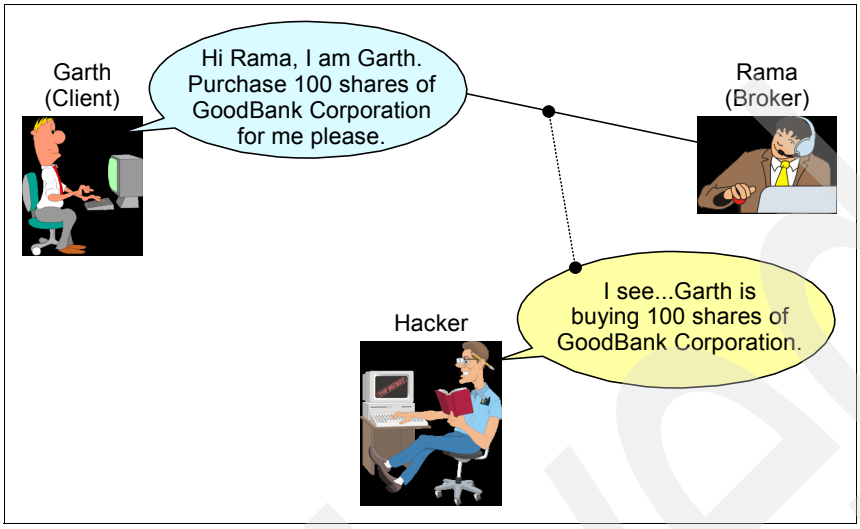


Figure A-2 Hacker intercepting the order

Assume that you have some way of knowing for sure that the order you have received originated from your client. How do you know that the order has not been read by anyone other than the two parties involved? You cannot be sure how many computers and links it has been across, and you do not know whether any intermediate link in the network has cached the message or logged it in any way, so what can you do?

The sender must alter the message so that its meaning is hidden to unauthorized parties using a process known as *encryption*. “Encryption” on page 663 explains this technology.

The order could have been intercepted and altered

Figure A-3 shows the hacker altering the original message.

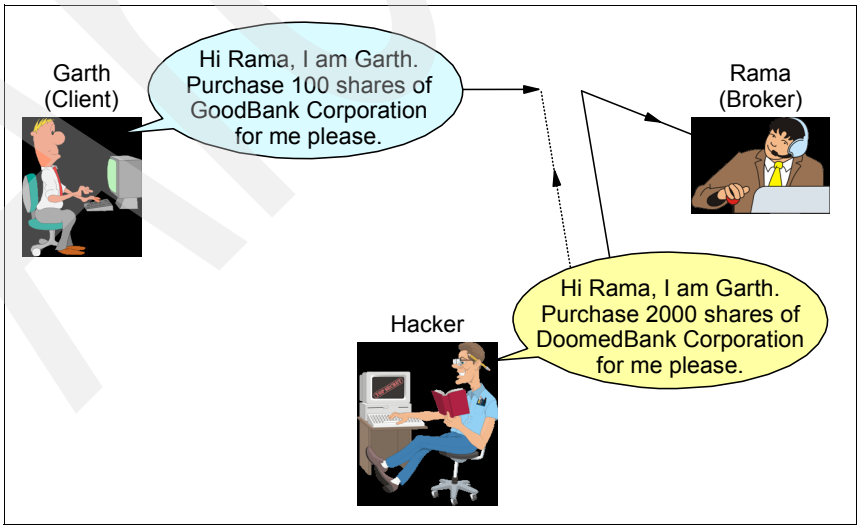


Figure A-3 Hacker intercepting and altering the order

How do you know, when you receive a message, that the contents of the message have not been modified? What is needed is some form of message authentication.

A message authentication process takes a message block, or stream, and mathematically summarizes the bits in the message to produce a fixed length message digest that represents that message. No two messages should produce the same message digest, or at least, it should be computationally unfeasible to find two that do. This message digest is normally appended to the original message, and transmitted along with it, to the destination. If the original message is altered, then when the receiver recalculates the message digest, it will differ from the one in the message. This does not guard against someone intercepting the message, altering it, recalculating the digest, and replacing the original digest and sending it along. That is why digests are often computed, using a secret shared key, which is then termed a message authentication code (MAC).

If the encryption process is by a private key (that is, nobody else knows the private key), then the MAC becomes a digital signature. A *digital signature* proves that one party, and one party alone, could have originated a particular message. If the message was intercepted and altered, the decryption process will yield rubbish and the receiver will know that the message should be retransmitted. These are explained in “Message authentication codes” on page 671, and “Digital signatures” on page 672.

An order is received from your client, but he denies sending it

Figure A-4 shows a hacker placing a false order and the client then denying that he placed the order.

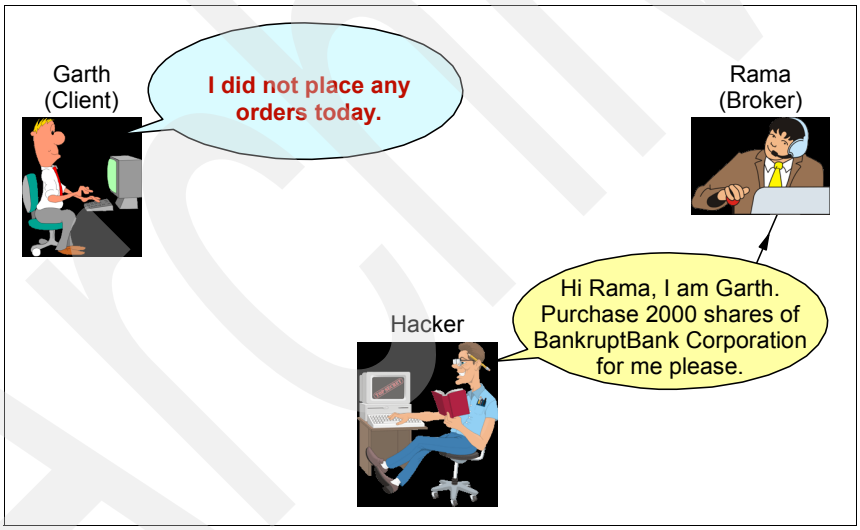


Figure A-4 Hacker placing an order

What is needed is some method where the sender of a message cannot deny having sent it. This requirement is called *non-repudiation*. This is done by making sure that the sender sends its request along with its unique digital signature. This is described in “Digital signatures” on page 672.

Secret key cryptography

Secret key cryptography is so called because the key used to encrypt the message must be kept secret from everyone but the two communicating parties. Ensuring a key is secret seems obvious but is not entirely necessary in public key systems; this is described in “Public key cryptography” on page 662. Another name for secret key encryption is *symmetric encryption*, so called because the same key that is used to encrypt the data is also used to decrypt the data and recover the clear text, as shown in Figure A-5.

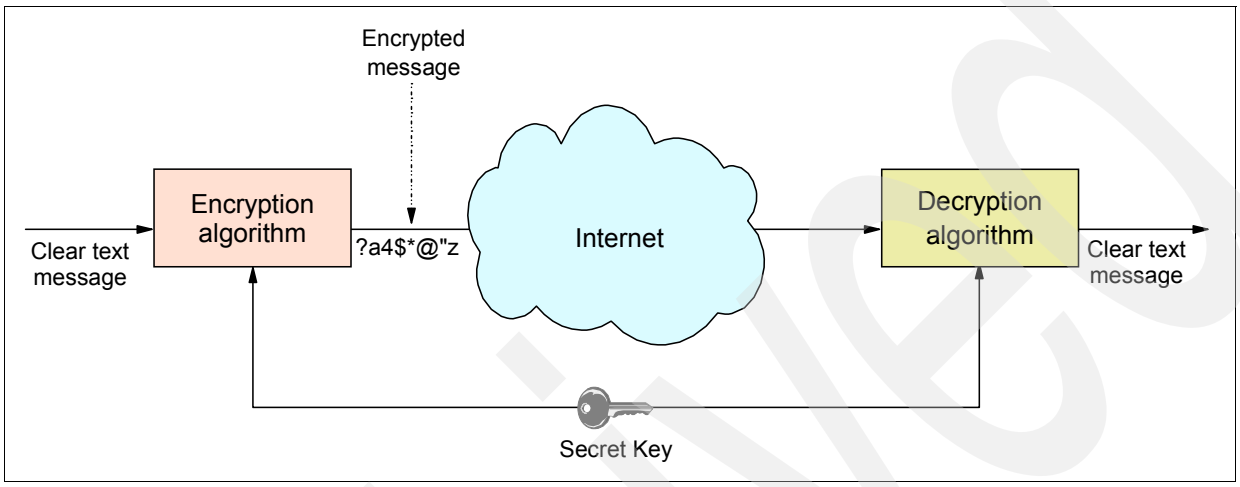


Figure A-5 Symmetric encryption and decryption: Using the same key

Symmetric algorithms are relatively efficient in terms of processing power. And, it is reasonable to offload popular symmetric algorithms to hardware features. Therefore, symmetric keys are the preferred method of encryption of bulk data.

However, they have one major drawback: 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. Public key encryption, described in “Public key cryptography” on page 662, can be used to exchange secret keys securely, and from then onward, the conversation can use the faster secret key encryption.

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, therefore the effective key length is 40 bits.

RC2 Developed by Ron Rivest for RSA Data Security, Inc., RC2 is a block cipher with variable key lengths operating on 8-byte blocks. Key lengths of 40 bits, 56 bits, 64 bits, and 128 bits are in use.

RC4 Developed by Ron Rivest for RSA Data Security, Inc., RC4 is 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 these ciphers, 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. And, the duration of the session is a factor too: if you change your symmetric key often enough, a potential hacker will not have time to crack it. 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, because the work factor rises exponentially with the size of the key.

Note: NIST has named Advanced Encryption Standard (AES) as the replacement for DES as the standard encryption algorithm. The Internet Engineering Task Force (IETF) IPSec Working Group intends for AES to eventually be adopted as the default IPSec Encapsulating Security Payload (ESP) cipher; therefore, AES must be included in compliant IPSec implementations.

Public key cryptography

Public key cryptography implements encryption and decryption using two different keys, which is why it is also termed *asymmetric encryption*. These two keys are known as a public key and a private key.

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, that person encrypts it using that key. Only you can understand the message, because only you have the private key.

Important: Public and private keys, if implemented in a reversible scheme such as RSA (described in the following section), yield extremely important properties:

- If the public key is used to encrypt the data, the private key must be used to recover the clear text. The public key cannot be used to decrypt (reverse) its own encryption—it only works in one direction.
- If the private key is used to encrypt the data, the public key must be used to recover the clear text.

Encryption

Figure A-6 shows an exchange where one party (left side) uses the second party's public key to encrypt a message.

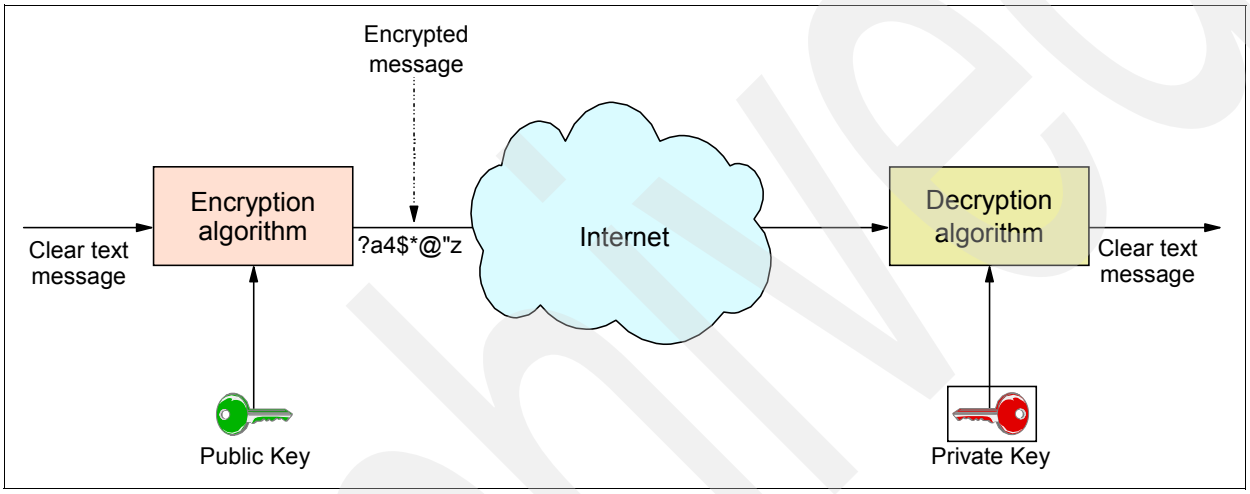


Figure A-6 Public-key cryptography: encryption using a public key

The ciphertext created by this encryption process is only decipherable by using the private key, which in turn is only known by the second party. There is no way for any other party to decipher this message. This type of encryption is used when you want the receiver to be the only person capable of understanding the message. This message flow can also be used to securely exchange a secret key between the conversation partners so that the faster secret (symmetric) key encryption can be used instead of public key.

Authentication

Asymmetric keys are also very useful for authentication. Look at Figure A-7 on page 664. What happens if you encrypt a message using your own private key?

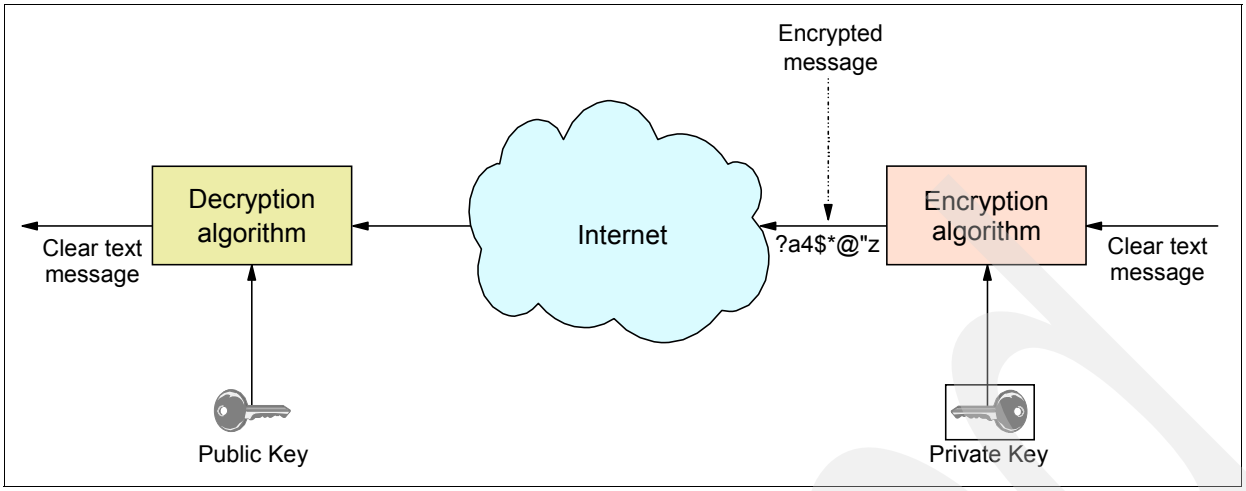


Figure A-7 Public-key cryptography: encryption using a private key results in authentication

As stated earlier, this indicates that anybody with access to your public key (and that should be anyone) would be able to decipher the message. This type of encryption, therefore, is obviously of no use to hide a message. By encrypting a message with your private key, a receiver *must* use your public key to decipher it, and that is the point: it proves that the message could *only* have come from you.

Public key algorithms

Asymmetric encryption algorithms, commonly called Public Key Cryptography 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 in 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 that the effort does not rise exponentially with the number of key bits. It is roughly $2^{(0.3*n)}$.

DSA z/OS (both gskkyman and RACF) also support the generation and storage of the Digital Signature Algorithm, or DSA key pairs. DSA asymmetric key generation is a standard published by the US Federal Government. DSA, along with RSA, is currently described in Federal Information Processing Standard, or FIPS 186-3.

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 for the DSA standard is also documented in FIPS PUB 186-3. Presently, there are no elliptic curve keys supported on the z/OS platform.

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*. Diffie-Hellman is not applicable to authentication of certificates during an TLS/SSL handshake like DSA and RSA. Instead, Diffie-Hellman is utilized in the protection of the symmetric key exchange used to establish IPSec VPNs.

Digital certificates

Digital certificates are used to publish a public key with a certainty that the public key is genuine, according to the Certificate Authority (CA) that digitally signs the certificate. First we discuss what can happen when a public key is used for communication, and that key is not genuine. We then cover what can be done about authenticating a public key by using digital certificates.

How to trust a published public key

If we want to communicate with ITSO Electronics Co., and we have found a public key published on the Internet, how can we use that public key? The two uses of another person's or entity's public key are:

- To decrypt a message originating from that person, who has encrypted with his private key
- To encrypt a message to be sent to that person so that only he can decrypt it with his private key

As mentioned, we have found ITSO Electronics Co.'s public key on the Internet. But, how do we know it is genuine? A malicious third party could have put its own public key on the Internet and now can intercept all communications from us to ITSO Electronics Co., acting as a sort of "relay" on the way; see Figure A-8.

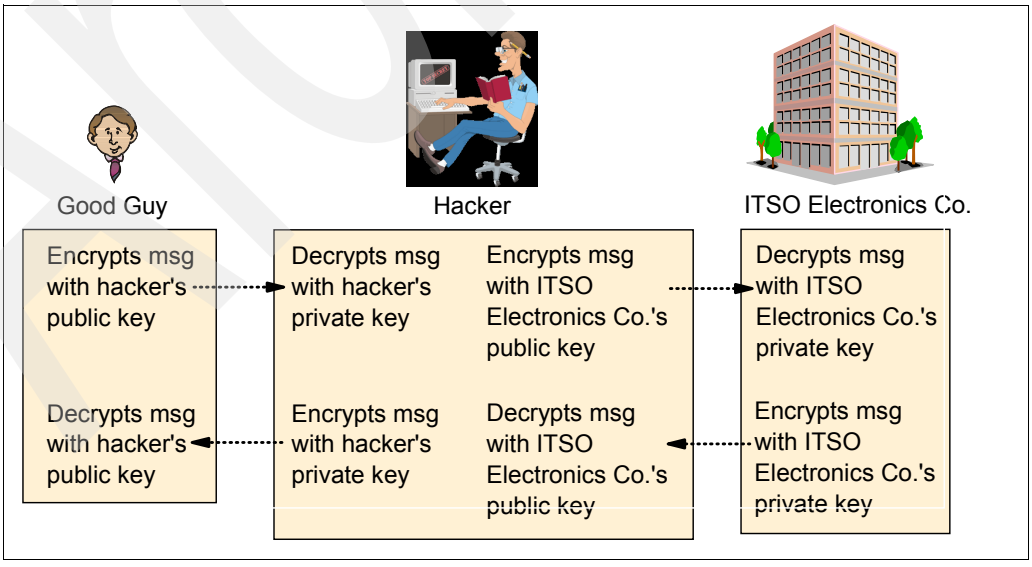


Figure A-8 Scenario where public key being used by good guy is really a hacker's key

In Figure A-8 on page 665, the “good guy” (representing us) assumes that he has obtained a public key for ITSO Electronics Co. from the Internet, but in reality, it was a hacker’s public key. We further assume that the hacker has some way of removing the our messages from the network, and injecting his own. The last assumption is that we are using ITSO Electronics Co.’s public key (or at least we think we are) to encrypt messages to ITSO Electronics Co. and the response will be encrypted by ITSO Electronics Co. with its private key.

You can see what can happen when a public key is used for communication when you do not know if it is genuine. Your messages to ITSO Electronics Co. will be encrypted using the hacker’s public key, because you thought it was ITSO Electronics Co.’s public key. The hacker then uses his private key to decrypt the message, make any changes he feels is necessary, and then encrypt the message with ITSO Electronics Co.’s real public key. When ITSO Electronics Co. receives the message, it will decrypt using its private key, process the message, and send a message back, encrypting with its own private key. The hacker then receives the response and decrypts using ITSO Electronics Co.’s public key, makes more changes, if necessary, and encrypts with his own (hacker’s) private key. Lastly, you receive the hacker’s message, decrypt with what you think is ITSO Electronics Co.’s public key, and now your communication to ITSO Electronics Co. has been totally compromised.

The problem of securely storing and retrieving public keys is dealt with by what is known as a Public Key Infrastructure (PKI), discussed in the following section. For more information about PKI, refer to IBM Redbook publication *Implementing PKI Services on z/OS*, SG24-6928, available at:

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

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.

When using a PKI, the user must be confident that the obtained 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 persons who request it have 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 authority 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. For instance, a passport issued by the governments of some countries, even if recognized to be authentic, will probably not be trusted by the government authorities of another country. Similarly, 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 CA. This can 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 4514 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 A-9 shows a simplified layout of a digital certificate.

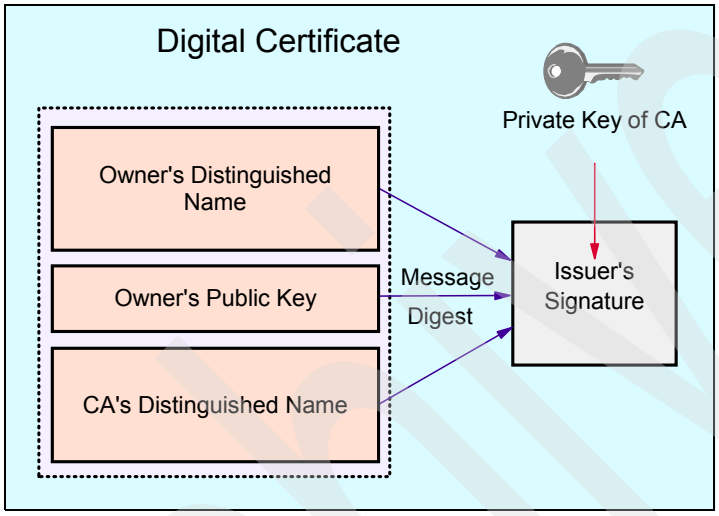


Figure A-9 Simplified layout of a digital certificate

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 you send your certificate with your public key in it to someone else, what keeps that person from misusing your certificate and posing as you? The answer is: your 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.

Certificate Authorities and trust hierarchies

Before we discuss what is termed a *certificate hierarchy*, let us look at an analogous example of trusted hierarchies. If you were selling a car, and a buyer asked you if it was acceptable to pay by personal check, then you have to decide whether you trust the buyer. If you do, the car is sold. If you do not trust the buyer, you may ask someone whom you both trust to countersign the check.

In digital certificate trust hierarchies, similar considerations to the car-buying example apply. Ultimately, the fact is that you have to trust somebody. You will have digital certificates in your database, and those certificates will either be set to *trusted* or *untrusted* status. A CA is a company that is considered trustworthy, and produces digital certificates for other individuals and companies (called *subjects*) bearing that subject's public key. This certificate is signed with a message hash that is encrypted using the CA's private 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 preconfigured 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.

Obtaining and storing certificates

As we have discussed, certificates are issued by a CA. If you do not want to use a CA, you can use utilities to issue your own certificates. These are called *self-signed* certificates and will only be accepted by people who trust you. Effectively, a self-signed certificate is its own CA, so you are still using a CA. However, the CA is not a separate certificate (and, the private key of this self-signed certificate is still the proof of identity).

Next, you can be your own local (internal) CA. For this, you create a single root CA and use it to sign one or more personal certificates.

If you use an external CA, you request certificates by visiting the CA vendor'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.

This is obviously an oversimplification. For more complete details on certificates, see Chapter 3, "Certificate management in z/OS" on page 31.

In the case of obtaining a certificate for a server, whether you use a self-signed or external CA signed certificate is dependent on the server environment. In an intranet environment, it is generally appropriate to use locally-signed certificates. In an environment where external users are accessing the server over the Internet, it is usually advisable to acquire a server certificate from a well-known CA, because the steps needed to import a locally signed certificate might be too labor-intensive. Many users will not have the ability to discern whether the action they are performing is of trivial consequence (for example, by importing a hacker's root CA inadvertently). 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.

Certificate management in z/OS

To manage certificates on a z/OS system, you can use either the UNIX program **gskkyman** to create and manage certificates, or you can use the RACF database and RACDCERT command. Again, this topic is covered in detail in Chapter 3, "Certificate management in z/OS" on page 31.

Performance issues of cryptosystems

Elliptic curve cryptosystems are said to have performance advantages over RSA and non-elliptical DSA 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) as it takes 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 be encrypted and decrypted, and the performance issues of public key systems are less important.

Message integrity

Message integrity is the ability to assert that a message received has not been altered in any way from the time that it was sent. In a networked environment, a message could have been altered by a third party intercepting it, or by some other means, such as electromagnetic interference (although in the latter case the transmission protocol normally handles a retransmission). To provide message integrity, you provide a message digest along with the text of your message. Note that the message being authenticated may or may not also be encrypted.

Message digest (or hash)

A message digest algorithm takes a message as input, and produces a small, fixed length *digest* string (usually 128 bits or 160 bits) often referred to as a *hash*. This hash can be thought of as a mathematical summary of a message. There are two important things to note about a message digest algorithm:

- The algorithm is a *one-way* function. This means that there is absolutely no way you can recover a message, given the hash of that message.
- It should be computationally unfeasible to produce another message that would produce the same message digest as another message.

Figure A-10 on page 670 is a graphical representation of appending a message digest to a message. When a message digest is appended to a message en route to its destination, the message cannot be tampered with, because a recalculation of the hash at the receiver's end will show the message digest received is invalid.

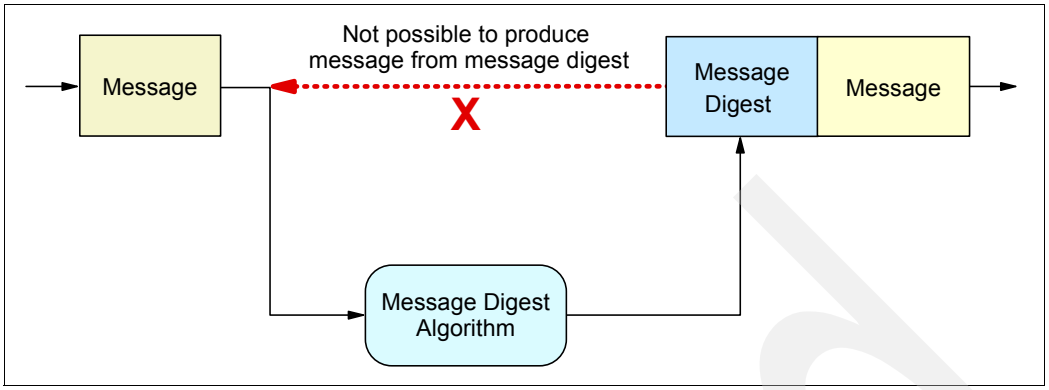


Figure A-10 Message digest

The message digest should not be sent in the clear. Because the digest algorithms are well known and no key is involved, a man-in-the-middle method could not only forge the message but also replace the message digest with that of the forged message. This will 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 as described in “Message authentication codes” on page 671.

Message digest algorithms

Common message digest algorithms are:

MD2 Developed by Ron Rivest of RSA Data Security, Inc., this algorithm is mostly used for Privacy Enhanced Mail (PEM) 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 MD5 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, as documented in *Cryptanalysis of MD5 Compress*, by Hans Dobbertin, which is 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, this 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 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. Because 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 SHS have been developed. SHA-256 provides a hash size of 256 bits, while SHA-512 provides a hash size of 512 bits.

Message authentication codes

Figure A-11 shows a message authentication code (MAC) being created for a message. A MAC is produced by feeding both the plaintext and a secret key into several iterations of the digest algorithm. The most common MAC procedure is HMAC, which simply iteratively applies the digest algorithm (such as SHA1).

That message digest can be encrypted with a key, and appended to the original message. Both the message and the associated MAC are then sent to the recipient. The assumption here is that the recipient shares the same key, so that the recipient may recompute the message digest and encrypt it with the shared key. This result should match the MAC sent on the message.

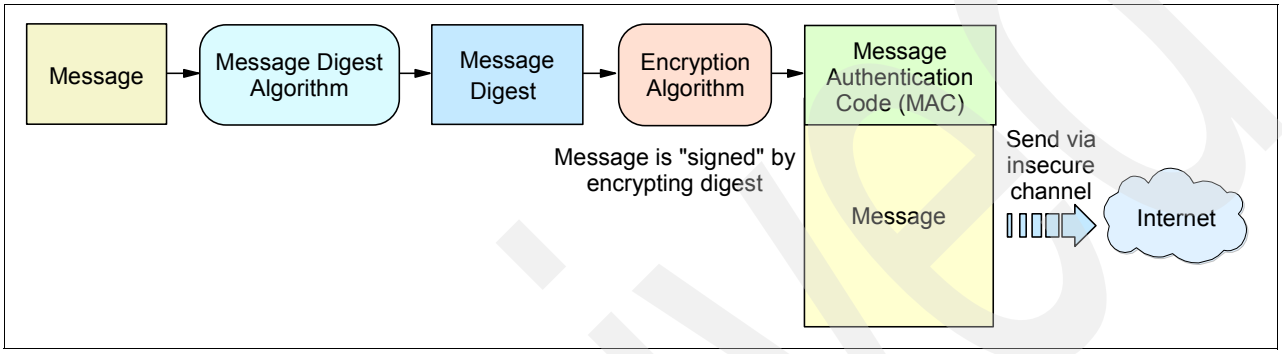


Figure A-11 Message digest for data integrity

Secret-key cryptographic algorithms, such as DES, can be used for encryption with message digests. A disadvantage of using a secret-key algorithm is that, because 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 key/private key encryption in order to provide for non-repudiation. When a MAC is encrypted with a sender's private key, rather than a secret (symmetric) key, that MAC becomes a *digital signature*. This is discussed further in "Digital signatures" on page 672.

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 is 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. In addition, HMAC with MD5 or SHA is used for IPSec VPNs.

Digital signatures

Digital signatures are 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 A-12, 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.

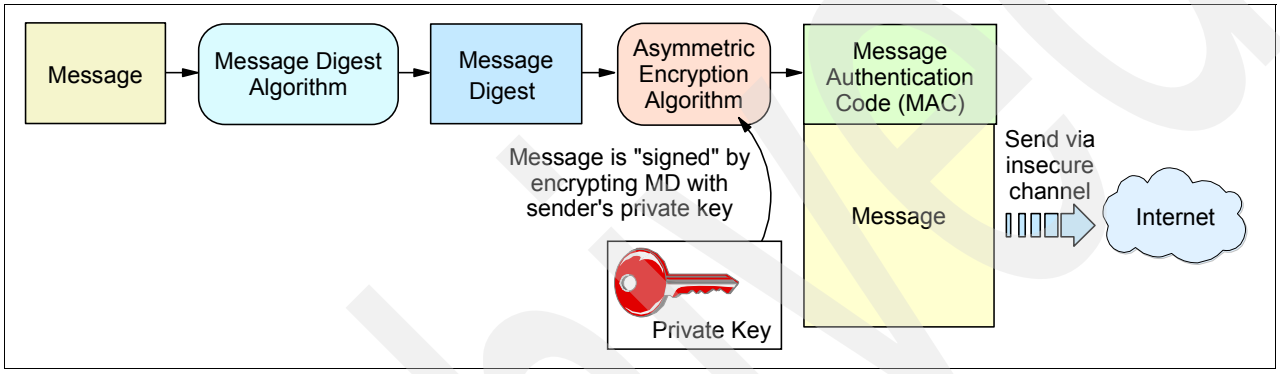


Figure A-12 Digital signature creation

The receiver, as shown in Figure A-13, uses the sender's public key to decrypt the message digest received. Then the receiver will use the message digesting algorithm to compute the message digest from the data received. If the computed 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.

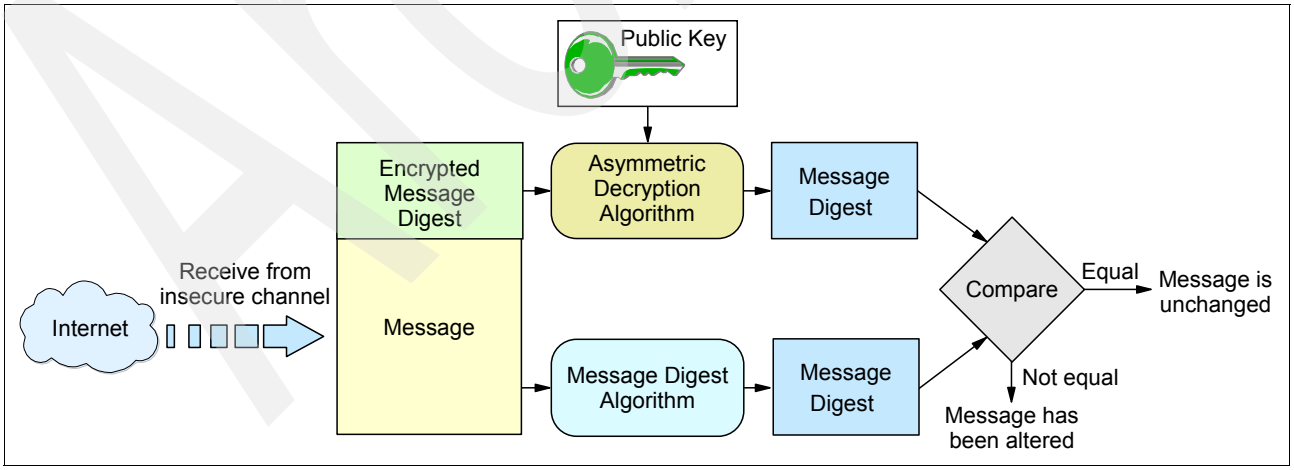


Figure A-13 Digital signature verification

With digital signatures, only public-key cryptosystems can be used. If secret-key cryptosystems are used to encrypt the signature, it will be very difficult to make sure that the receiver (having the key to decrypt the signature) cannot misuse this key to forge a signature of the sender. The private key of the sender is known to nobody else, therefore 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 the receiver's private key. This means everybody can send encrypted data to the receiver that only the receiver can decrypt. See Figure A-14 for a graphical representation.

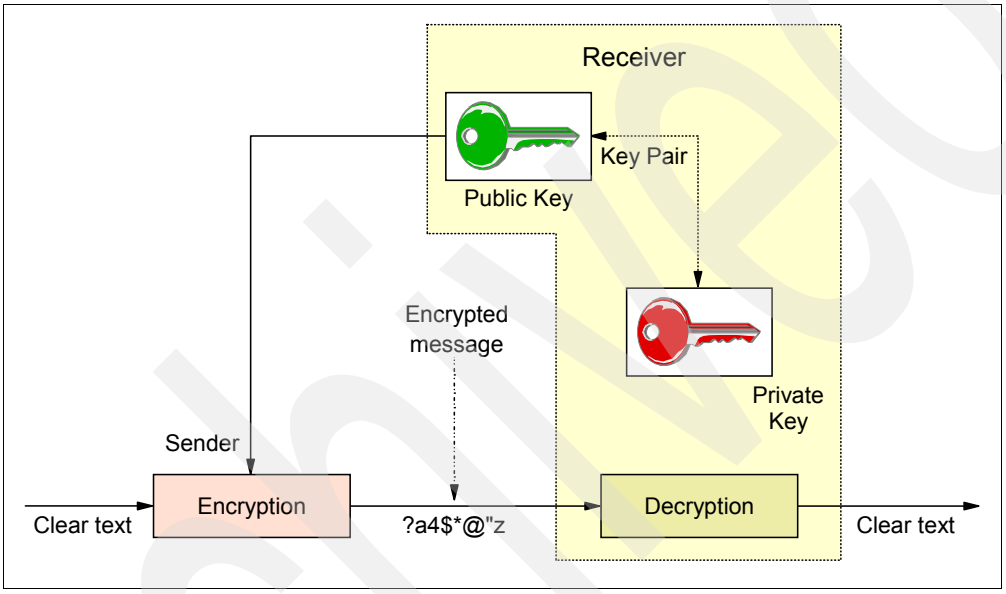


Figure A-14 Encrypting data with the receiver's public key

- With digital signatures, the sender uses the sender's private key to encrypt the sender's 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 issue with digital signatures is the trustworthy distribution of public keys, because a genuine copy of the sender's public key is required by the receiver. A solution to this problem is provided by digital certificates, as discussed in "Digital certificates" on page 665.

B

Telnet security: advanced settings

This appendix contains the implementation examples of advanced TN3270 server security using client ID groups. We show two scenarios: one configured with native Transport Layer Security (TLS) function, and the other configured with Application Transparent Transport Layer Security (AT-TLS) policy. As discussed in Chapter 14, “Telnet security” on page 539, we recommend the use of AT-TLS instead of the native TLS to implement the secure connections.

The following topics are discussed in this appendix.

Section	Topic
“Advanced native TLS configuration” on page 676	Implementation and verification of the advanced TN3270 server configuration using the native TLS function
“Advanced AT-TLS configuration using client ID groups” on page 688	Implementation and verification of the advanced TN3270 server configuration using the AT-TLS policy

Advanced native TLS configuration

In this scenario we define client ID object groups for granular control of the connection types the clients should use with a single port (992).

We define Dynamic VIPA (DVIPA) addresses that are to be associated with the client ID groups representing the different departments (or user groups) with various security requirements.

Following client ID groups are defined:

- General user - Accesses port 992 on destination 10.1.8.41, requiring no SSL
- Admin - Accesses port 992 on destination 10.1.8.42, requiring plain SSL
- Payroll - Accesses port 992 on destination 10.1.8.43, requiring client authentication
- Shipping - Accesses port 992 on destination 10.1.1.40, decides at connection time

The destination addresses that belong to subnet 10.1.8.* are Dynamic VIPA addresses defined by the TCP/IP stack. They are not defined as Distributed Dynamic VIPAs in the stack. The 10.1.1.40 address is the static VIPA address of the TCPIPD stack on SC33.

Figure B-1 depicts the environment for this scenario.

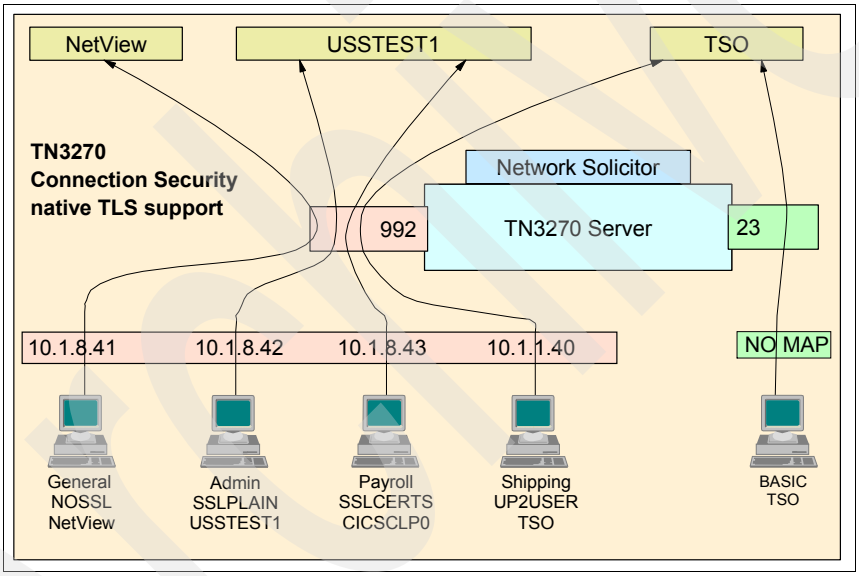


Figure B-1 Diagram of the advanced native TLS configuration

Implementation tasks

The following tasks are required to enable native TLS/SSL support for TN3270, with server authentication:

- Generate the key ring and certificates.
- Add a TLS-enabled port and security parameters to TN3270 profile.

Generate the key ring and certificates

Note: Chapter 3, “Certificate management in z/OS” on page 31,contains detailed examples and explanations for the steps involved in preparing the key ring and certificates used in this scenario. Refer to that chapter for a complete discussion about using a shared key ring and a SITE certificate.

Example B-1 shows the RACF statements necessary to create the shared key ring, the CA certificate, and the SITE certificate used by this scenario.

Example: B-1 Sample RACF for key ring and certificates

```
racdcert certauth gencert                          -
   subjectsdn( o('IBM Corporation')                 -
               ou('ITSO Certificate Authority')     -
                C('US'))                            -
    NOTBEFORE(DATE(2007-09-11))                     -
    NOTAFTER(DATE(2008-09-11))                      -
    keyusage(certsign)                              -
    withlabel('CS19 ITSO CA1')
  setropts raclist(facility) refresh
  racdcert certauth list

  racdcert site gencert subjectsdn(cn('ITSO.IBM.COM')   -
     o('IBM Corporation')                             -
     ou('ITSO CS19 Shared SITE')                      -
     C('US'))                                         -
     withlabel('CS19 ITSO SharedSite1')               -
     signwith(certauth label('CS19 ITSO CA1')
  racdcert site list

  racdcert ID(TCPIP) ADDRING(SharedRing1)

  racdcert ID(TCPIP) CONNECT(CERTAUTH                 -
                     LABEL('CS19 ITSO CA1')           -
                     RING(SharedRing1)                -
                     USAGE(CERTAUTH)

  racdcert ID(TCPIP) CONNECT(SITE                     -
                     LABEL('CS19 ITSO SharedSite1')   -
                     RING(SharedRing1)                -
                     DEFAULT                          -
                     USAGE(PERSONAL)

  setropts raclist(DIGTRING) refresh
  setropts raclist(DIGTCERT) refresh
  racdcert listring(*) id(TCPIP)
```

1 Creates a self-signed Certificate Authority certificate

2 Creates a SITE certificate

3 Creates a key ring

4 Connects a self-signed Certificate Authority certificate to a key ring

5 Connects a SITE certificate to a key ring

Add a TLS-enabled port and security parameters to TN3270 profile

Example B-2 on page 678 shows the example of TELNETPARMS definition and the use of PARMSGROUP parameter sets mapped to client ID groups.

The client ID groups are defined by identifying the clients that access the stack using the destination IP addresses. There are a number of alternative mapping methods to define client ID groups. This scenario uses the DESTIPGRP method.

Any user who accesses the stack over any IP address not defined explicitly in this profile will be presented with the Network Solicitor panel and be prompted for a user ID, password, and desired application. No SSL security will be implemented for that type of connection.

Example: B-2 Defining the TN3270 profile for native TLS connection security

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPD
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
ENDTELNETGLOBALS
;
TELNETPARMS
    SECUREPORT 992  ;Port 992 supports native TSL 1
    KEYRING SAF TCPIP/SharedRing1       ;keyring shared by servers 2
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 992
  DEFAULTLUS
     SC33DS01..SC33DS99
  ENDDEFAULTLUS
; ---------------------------------------------------------------------
; This NOSSL       group is mapped to use no SSL security.            -
; ---------------------------------------------------------------------
  PARMSGROUP NOSSL 3
     NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The SSLPLAIN group is mapped to use SSL security                     -
;                  with no Client Authentication required              -
; ---------------------------------------------------------------------
  PARMSGROUP SSLPLAIN 4
     CONNTYPE SECURE ; says plain SSL, no client auth specified
  ENDPARMSGROUP      ; and negotiate all available encryption algorithms
```

```
; ---------------------------------------------------------------------
; The SSLCERTS group is mapped to use SSL security                     -
;                 and to require Client Authentication (certificates) -
; ---------------------------------------------------------------------
  PARMSGROUP SSLCERTS 5
     CONNTYPE SECURE       ; Support SSL
     CLIENTAUTH SSLCERT    ; Client Certificate required
     ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
             SSL_3DES_SHA
     ENDENCRYPT
  ENDPARMSGROUP

; ---------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                 with no Client Authentication (no certificates)     -
; ---------------------------------------------------------------------
  PARMSGROUP UP2USER 6
     CONNTYPE ANY          ; Whatever User wants to do
  ENDPARMSGROUP

  DESTIPGROUP GENERALUSER 10.1.8.41   ENDDESTIPGROUP 7 ; D-VIPA
  DESTIPGROUP ADMIN       10.1.8.42   ENDDESTIPGROUP 8 ; D-VIPA
  DESTIPGROUP PAYROLL     10.1.8.43   ENDDESTIPGROUP 9 ; D-VIPA
  DESTIPGROUP SHIPPING    10.1.1.40    ENDDESTIPGROUP 10 ; Static VIPA

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER 11
  DEFAULTAPPL SC33N     DESTIPGRP,GENERALUSER

  PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN 12
  USSTCP   USSTEST1     DESTIPGRP,ADMIN

  PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL 13
  USSTCP USSTEST1 DESTIPGRP,PAYROLL

  PARMSMAP UP2USER      DESTIPGRP,SHIPPING 14
  DEFAULTAPPL TSO       DESTIPGRP,SHIPPING

  ALLOWAPPL SC33N*          ; Netview
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *       ; Allow all applications that have not been
                    ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    PORT 23         ; Port 23 supports basic (non-secure) connections
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23
```

```
  DEFAULTLUS
      SC33DB01..SC33DB99
  ENDDEFAULTLUS

  DEFAULTAPPL SC33TS        ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

1 The port 992 is used.

2 The name of the key ring in use.

3 PARMSGROUP NOSSL for basic (no TLS/SSL) connection.

4 PARMSGROUP SSLPLAIN for secure connection with no client authentication.

5 PARMSGROUP SSLCERTS for secure connection with client authentication.

6 PARMSGROUP UP2USER for basic or secure connection.

7 Client ID group GENERALUSER is defined for clients destined to 10.1.8.41.

8 Client ID group ADMIN is defined for clients destined to 10.1.8.42.

9 Client ID group PAYROLL is defined for clients destined to 10.1.8.43.

10 Client ID group SHIPPING is defined for clients destined to 10.1.1.40.

11 Client ID group GENERALUSER is mapped to NOSSL PARMSGROUP.

12 Client ID group ADMIN is mapped to SSLPLAIN PARMSGROUP.

13 Client ID group PAYROLL is mapped to SSLCERTS PARMSGROUP.

14 Client ID group SHIPPING is mapped to UP2USER PARMSGROUP.

Start the TN3270 server

To apply the new definition, start or restart the TN3270 server. The OBEYFILE command can be used also, but it is only effective for new connections. Example B-3 shows the messages given at the initialization of the TN3270 server.

Example: B-3 Starting the TN3270 server

```
S TN3270D
.....
EZZ6001I TELNET SERVER STARTED
.....
EZZ6003I TELNET LISTENING ON PORT   992 1
EZZ6003I TELNET LISTENING ON PORT    23 2
```

1 The native TLS port 992 is now active.

2 The basic connection port 23 is also active (the definition is not shown in this chapter).

Activation and verification

The following commands can be useful when validating secure port information in the Telnet server environment:

- Use TELNET CONN displays to show TN3270 connections.
- Display CONN to show connection SSL information
- Display CLIENTID to show client group SSL information.
- Display OBJECT to show object SSL information.

We used IBM Personal Communications (PComm) V5.7 to establish the connection. Because we used self-signed certificate, we downloaded and installed the certificate of Certificate Authority into the PComm.

Use TELNET CONN displays to show TN3270 connections

The display CONN command without the CONN= parameter specified gives you a high level view of what connections exist and what they are being used for. Look for EN/TY, Encryption Type. The connection command shows current connections and associated resources such as their LU name, Logmode, and application being used, as seen in Example B-4.

Example: B-4 Display Telnet CONN for connection overview information

```
D TCPIP,TN3270D,T,CONN,MAX=*
   EZZ6064I TELNET CONNECTION DISPLAY 719
            EN                                          TSP
   CONN     TY IPADDR..PORT           LUNAME   APPLID   PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   00002067    ::FFFF:10.1.1.30..1033 1
                                      SC33DS09 SC33TS08  TA3 SNX32704 1
   00002063    ::FFFF:10.1.1.20..1031 2
                                      SC33DS08 SC33N008  TA3 SNX32704 2
   00001FD1 4S ::FFFF:10.1.100.221..3235 3
                                      SC33DS07           TPE 3
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   8 OF 8 RECORDS DISPLAYED
```

1 TSO Telnet client on SC32 is connected to 10.1.1.40, port 992 (being mapped to TSO on SC33). A LU SC33DS09 is assigned, where S indicates the LU pool for port 992. The destination address 10.1.1.40 is associated with PARMSGROUP UP2USER (SSL optional), but the TSO Telnet client does not negotiate or request SSL. Therefore, connection 2067 did not perform any SSL handshake, and thus no encryption type is indicated.

2 TSO Telnet client on SC31 is connected to 10.1.8.41 port 922 (being mapped to NetView on SC33). The destination address 10.1.8.41 is associated with PARMSGROUP NOSSL, so no SSL handshake is performed and no encryption type is indicated.

3 Connection from a Personal Communications terminal is connected to destination IP address 10.1.8.42, port 992, to request SSL without client authentication. Notice the encryption type is 4S. The connection shows `pending` (T**P**E) with no APPLID or LOGMODE assigned while the USSTEST1 MSG10 is displayed. Then the user selects an application (NVAS, in our case) and TN3270 fills in the applid name and the associated logmode as shown in Example B-7 on page 683 and Example B-8 on page 683.

Display CONN to show connection SSL information

Example B-5 on page 682 and Example B-6 on page 682 show the display of CONN command before the user selects an application. Notice APPLID and LOGMODE are not

filled yet. TN3270 fills in the applid name and associated logmode, as shown in Example B-7 on page 683 and Example B-8 on page 683. Look for SSL information. An example is shown in Example B-5.

Example: B-5 Display Telnet CONN for SSL information

```
D TCPIP,TN3270D,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 382
            EN                                              TSP
   CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
   -------- -- --------------------- -------- --------  --- --------
   0000B23C 4S ::FFFF:10.1.100.224..2880
                                     SC33DS01           TPE
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

The CONNECTION display command with the CONN= parameter and DETail option specified gives you a complete look at one connection. It shows all the information available regarding a single connection. Look for TLS/SSL information. An example is shown in Example B-6.

Example: B-6 Display Telnet CONN DETAIL for SSL information, before UNIX System Services selection is made

```
D TCPIP,TN3270D,T,CONN,CONN=B23C,DET
   EZZ6065I TELNET CONNECTION DISPLAY 384
     CONNECTED: 11:36:00  09/26/2007  STATUS: SESSION PENDING 1
     CLIENT IDENTIFIER FOR CONN: 0000B23C   SECLABEL: **N/A**
       CLIENTAUTH USERID: **N/A**
       HOSTNAME: NO HOSTNAME
       CLNTIP..PORT: ::FFFF:10.1.100.224..2880
       DESTIP..PORT: ::FFFF:10.1.8.42..992
       LINKNAME: VIPL0A01082A
     PORT:   992 QUAL: NONE
       AFFINITY: TCPIPD
       STATUS: ACTIVE  SECURE        ACCESS: SECURE  4S TLSV1 2
     PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
       TYPE: TERMINAL GENERIC
    INPUT ===>                                          SCROLL ===> CSR
       OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                          NEWENV FUNCTIONS: --
     LUNAME: SC33DS01
     APPL: **N/A** 3
       USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
       LOGMODES  TN REQUESTED:           APPL SPECIFIED: 3
     MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
       LUMAP GEN:  NL (NULL)
                        >*DEFLUS*                 --------
       DEFLT APPL: **N/A**
       USS TABLE:  NL (NULL)
                        >USSTEST1                 P-------
       INT TABLE:  **N/A**
       PARMS:
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
     -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
```

```
    LM-----  ---S--------  --F  SSS------  *---ST-  S--- *TPARMS
    LM*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* TP-CURR
    LM*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* <-FINAL 3
  35 OF 35 RECORDS DISPLAYED
```

1 The connection is in pending state before the application is selected by user.

2 The connection is secure and the cipher used is 4S (SSL_RC4_SHA). Refer to Table 14-1 on page 543 for the complete list of supported ciphers.

3 The application name and the logmode is blank before the user logs on to a specific application.

4 The security parameter is defined in the Telnet profile. Each letter of PCKLECXN2 stands for one of the SECURITY parameters listed, in the same order, in Example B-9 on page 684.

After the user selects an application, TN3270 fills in the applid name and associated logmode, as shown in Example B-7 and Example B-8.

Example: B-7 Connection summary information for SSL port 992 after USSMSG10 appl is selected

```
D TCPIP,TN3270D,T,CONN
   EZZ6064I TELNET CONNECTION DISPLAY 421
            EN                                        TSP
   CONN     TY IPADDR..PORT           LUNAME   APPLID   PTR LOGMODE
   -------- -- ---------------------- -------- --------  --- --------
   0000B23C 4S ::FFFF:10.1.100.224..2880
                                    SC33DS01 SC33TS04  TAE SNX32702
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      1
   ------------------------------------------------------------
   4 OF 4 RECORDS DISPLAYED
```

Example B-8 shows further connection detail information.

Example: B-8 Connection detail information for SSL port 992 after USSMSG10 appl is selected

```
D TCPIP,TN3270D,T,CONN,CONN=B23C,DET
EZZ6065I TELNET CONNECTION DISPLAY 400
  CONNECTED: 11:36:00  09/26/2007  STATUS: SESSION ACTIVE 1
  CLIENT IDENTIFIER FOR CONN: 0000B23C   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..2880
    DESTIP..PORT: ::FFFF:10.1.8.42..992
    LINKNAME: VIPLOA01082A
  PORT:   992 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  SECURE        ACCESS: SECURE  4S TLSV1 2
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                       NEWENV FUNCTIONS: --
  LUNAME: SC33DS01
  APPL: SC33TS04 3
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702 3
  MAPPING TYPE:  CONN IDENTIFIER
                      OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                     >*DEFLUS*                  --------
```

```
   DEFLT APPL: **N/A**
   USS TABLE:  NL (NULL)
                  >USSTEST1                  P-------
   INT TABLE:  **N/A**
   PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  LM-----  ---S--------  --F  SSS------  *---ST-  S--- *TPARMS
  LM*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* TP-CURR
  LM*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* <-FINAL
35 OF 35 RECORDS DISPLAYED
```

1 The connection is in active state after the application is selected by user.

2 The connection is secure and the cipher used is 4S (SSL_RC4_SHA). Refer to Table 14-1 on page 543 for the complete list of supported ciphers.

3 The application name and the logmode is filled after the user logs on to a specific application.

4 The security parameter is defined in Telnet profile. Each letter of PCKLECXN2 stands for one of the SECURITY parameters listed, in the same order, in Example B-9.

Display PROF to show profile SSL information

The PROFILE display command enables you to determine what profile-wide options are in effect for each profile, including the security specifications, as shown in Example B-9.

Example: B-9 Display Telnet PROFILE for SSL information, detail

```
D TCPIP,TN3270D,T,PROF,PORT=992,DET
   EZZ6080I TELNET PROFILE DISPLAY 457
     PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
    (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
     -------  ------------  ---  ---------  -------  ----
     *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
     -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
     LM-----  ---S--------  --F  SSS------  *---ST-  S--- *TPARMS
     LM*****  **TSBTQ***RT  ECF  SSS*D****  *P**STS  SDD* CURR
   .....
   SECURITY
      SECUREPORT             992
      CONNTYPE            SECURE
      KEYRING             SAF TCPIP/SharedRing1
      CRLLDAPSERVER       NONE
      ENCRYPTION          4S,4M,A2,A1,3S,DS,4E,2E,NS,NM,NN (DEF)
      CLIENTAUTH          NONE
      NOEXPRESSLOGON
      NONACUSERID
      NOSSLV2
    TIMERS
   .....
      SSLTIMEOUT               5
   .....
      KEYRING             SAF TCPIP/SharedRing1
```

Display CLIENTID to show client group SSL information

The CLIENTID display can be used to see what client IDs are defined in the profile and details about the client ID, such as a DESTIPGRP. Look for SSL information, as highlighted in Example B-10.

Example: B-10 Display Telnet CLIENTID for SSL information, detail

```
D TCPIP,TN3270D,T,CLID,PORT=992,DET,MAX=*
   EZZ6081I TELNET CLIENTID DISPLAY 502
   CLIENT ID          CONNS  OBJECT    OBJECT   ITEM
   NAME               USING  TYPE      NAME     SPECIFIC   OPTIONS
   ------------------ ------ --------- -------- ---------- --------
   DESTIPGRP
     GENERALUSER
                          0 DEFAPPL   SC31TS               --------
     GENERALUSER
                          0 PARMSGRP  NOSSL                --------
     ADMIN
                          0 USS       USSTEST1             P-------
     ADMIN
                          0 PARMSGRP  SSLPLAIN             --------
     PAYROLL
                          0 DEFAPPL   CICSCLPO             --------
     PAYROLL
                          0 PARMSGRP  SSLCERTS             --------
     SHIPPING
                          0 USS       USSTABEE,USSSNAEE    PP------
     SHIPPING
                          0 PARMSGRP  UP2USER              --------
   NULL
     NULL
                          1 USS       USSTEST1             P-------
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   22 OF 22 RECORDS DISPLAYED
```

A client ID summary report can show a specific client group and the names that make it up, as seen in Example B-11.

Example: B-11 Display Telnet CLIENTID for SSL information, summary

```
D TCPIP,TN3270D,T,CLID,PORT=992,TYPE=DESTIPGRP,SUM,MAX=*
   EZZ6082I TELNET CLIENTID LIST 504
   DESTIPGRP
     GENERALUSER       ADMIN             PAYROLL
     SHIPPING
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   5 OF 5 RECORDS DISPLAYED
```

Display OBJECT to show object SSL information

The OBJECT display can be used to see what objects are defined in the profile and some details about the object. If you specify a TYPE (which you know is related to a secure group you defined, such as DEFAPPL, PARMSGRP, or USS), you can see SSL-related information. Some examples are shown in Example B-12.

Example: B-12 Display Telnet OBJECT for SSL information, summary

```
D TCPIP,TN3270D,T,OBJ,PORT=992,SUM,MAX=*
```

```
EZZ6084I TELNET OBJECT LIST 586
ARAPPL
  SC33N*    NVAS*     TSO*      *
DEFAPPL
  SC33N     CICSCLPO  TSO
PRTAPPL
  NO OBJECTS
LINEAPPL
  NO OBJECTS
MAPAPPL
  NO OBJECTS
USS
  USSTEST1
INT
  NO OBJECTS
LU
  NO OBJECTS
LUGRP
  *DEFLUS*
APPLLUG
  NO OBJECTS
PRT
  NO OBJECTS
PRTGRP
  NO OBJECTS
PARMSGRP
  NOSSL     SSLPLAIN  SSLCERTS  UP2USER   *DEFAULT  *TGLOBAL
  *TPARMS
MONGRP
  NO OBJECTS
----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
------------------------------------------------------------
31 OF 31 RECORDS DISPLAYED
```

The object detail report can be filtered to show a specific object type and the client IDs mapped to the type, as seen in Example B-13.

Example: B-13 Display Telnet OBJECT for SSL information, DEFAPPL

```
D TCPIP,TN3270D,T,OBJ,PORT=992,TYPE=DEFAPPL,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 613
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   DEFAPPL
    SC33N           1 DESTIPGRP GENERALUSER
                                                           --------
    CICSCLPO        0 DESTIPGRP PAYROLL
                                                           --------
    TSO             1 DESTIPGRP SHIPPING
                                                           --------
   ----- PORT:   992  ACTIVE              PROF: CURR CONNS:      3
   ------------------------------------------------------------
   9 OF 9 RECORDS DISPLAYED
```

The object report can be used to check which PARMSGROUPs are mapped to which client IDs. The report indicates how many connections are associated with which group, as seen in Example B-14 on page 687.

Example: B-14 Display Telnet OBJECT for SSL information, PARMSGRP

```
D TCPIP,TN3270D,T,OBJ,PORT=992,TYPE=PARMSGRP,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 511
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   PARMSGRP
    NOSSL           0 DESTIPGRP GENERALUSER
                                                            --------
    SSLPLAIN        0 DESTIPGRP ADMIN
                                                            --------
    SSLCERTS        0 DESTIPGRP PAYROLL
                                                            --------
    UP2USER         0 DESTIPGRP SHIPPING
                                                            --------
    *DEFAULT          -------NO MAPPING---------
                                                            --------
    *TGLOBAL          -------NO MAPPING---------
                                                            --------
    *TPARMS           -------NO MAPPING---------
                                                            --------
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   17 OF 17 RECORDS DISPLAYED
```

The object report can be used to check which USS tables are mapped to which client IDs. The report indicates how many connections are associated with which table, as seen in Example B-15.

Example: B-15 Display Telnet OBJECT for SSL information, USS

```
D TCPIP,TN3270D,T,OBJ,PORT=992,TYPE=USS,DET,MAX=*
   EZZ6083I TELNET OBJECT DISPLAY 513
   OBJECT      CONNS  CLIENT ID CLIENT ID        ITEM
   NAME        USING  TYPE      NAME             SPECIFIC   OPTIONS
   ----------  ------ --------- ---------------- ---------- --------
   USS
    USSTEST1        1 NULL      NULL
                                                            P-------
    USSTEST1        0 DESTIPGRP ADMIN
                                                            P-------
    USSTABEE,
     USSSNAEE       0 DESTIPGRP SHIPPING
                                                            PP------
   ----- PORT:   992  ACTIVE               PROF: CURR CONNS:      1
   ------------------------------------------------------------
   10 OF 10 RECORDS DISPLAYED
```

For more information, refer to the verification steps for a Telnet standalone task described in *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533.

Additional DISPLAY commands are used to show the security settings of a secure port. For a complete list of available TELNET-related commands and their syntax, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Advanced AT-TLS configuration using client ID groups

In this scenario we define client ID object groups for granular control of the connection types the clients should use with a single port (4992). We use the AT-TLS policy to implement the secure connection.

We define Dynamic VIPA (DVIPA) addresses that are to be associated with the client ID groups representing the different departments (or user groups) with various security requirements. Some of the client ID grouping statements (PARMSGROUP, PARMSMAP) can be included into the AT-TLS policy and omitted from the Telnet profile.

This scenario is similar to the scenario in “Advanced native TLS configuration” on page 676. The following client ID groups are defined:

- General user - Accesses port 992 on destination 10.1.8.41, requiring no SSL
- Admin - Accesses port 992 on destination 10.1.8.42, requiring plain SSL
- Payroll - Accesses port 992 on destination 10.1.8.43, requiring client authentication
- Shipping - Accesses port 992 on destination 10.1.1.40, decides at connection time

The destination addresses that belong to subnet 10.1.8.* are Dynamic VIPA addresses defined by the TCP/IP stack. They are not defined as Distributed Dynamic VIPAs in the stack. The 10.1.1.40 address is the static VIPA address of the TCPIPD stack on SC33.

Figure B-2 depicts the environment we use for this scenario.

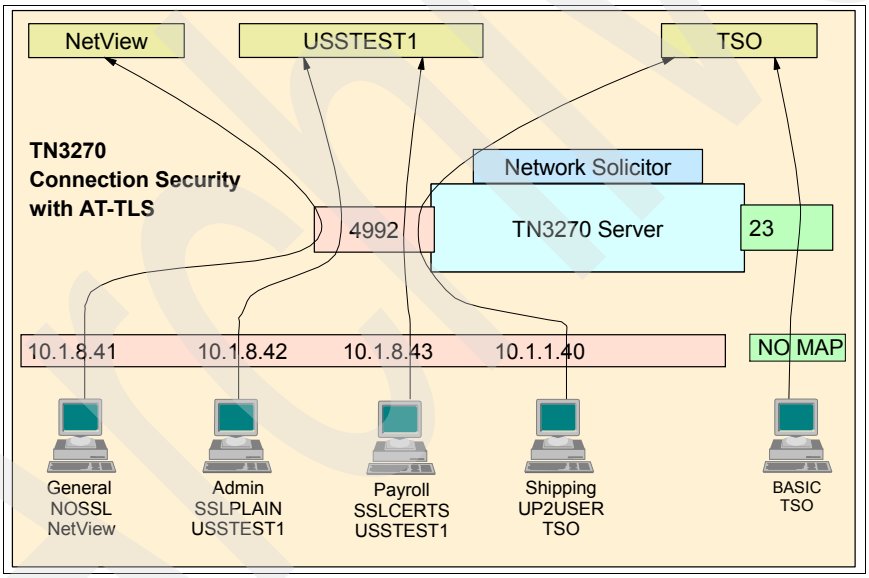


Figure B-2 TN3270 with advanced AT-TLS configuration diagram

Implementation tasks

Perform the following tasks to configure TN3270 AT-TLS support:

- Set up the policy agent.
- Create a certificate.
- Add authorization for the **`pasearch`** command in RACF.
- Modify the TCP/IP profile.
- Define AT-TLS policies.
- Upload the policy to z/OS.
- Modify the policy agent configuration file.

- Define the AT-TLS port in the TN3270 configuration file.

Set up the policy agent

We set up the policy agent as shown in 5.2, “Implementing PAGENT on z/OS” on page 229.

Create a certificate

We continued to use the certificate created for native TLS connection security as shown in “Advanced native TLS configuration” on page 676. If you have not created a certificate, follow the instructions provided in that section.

Add authorization for the pasearch command in RACF

If users other than superusers need to issue the **`pasearch`** command, add authorization for the **`pasearch`** command in RACF. This command is a sensitive command, so make sure you restrict the access to only administrators or operators. We set up the RACF authorization as shown in 5.2, “Implementing PAGENT on z/OS” on page 229.

Modify TCP/IP profile

To enable AT-TLS to the TCP/IP stack, add the TCPCONFIG TTLS statement in the TCP/IP Profile. To apply the change, either restart the TCP/IP stack or use the OBEYFILE command.

Example: B-16 Modify TCP/IP profile

```
TCPCONFIG TTLS
```

Define AT-TLS policies

We defined AT-TLS connectivity rules for groups that have or may have secure connections, as shown in Table B-1. We did not need to define the policy for the basic (non-secure) connections.

Note: The CONNTYPE statement must be defined in the TN3270 profile. There is no equivalent statement for CONNTYPE in the AT-TLS policy.

Table B-1 Connectivity rules in AT-TLS policy

Connectivity rule name	CONNTYPE in TN3270 profile	Destination IP address	Client authentication
ADMIN_SSLPLAIN	SECURE (default)	10.1.8.42	NONE
PAYROLL_SSLCERT	SECURE (default)	10.1.8.43	SSLCERT (Required)
SHIPPING_UP2USER	ANY	10.1.1.40	NONE

We used the IBM Configuration Assistant for z/OS Communication Server to define AT-TLS policies, as described here:

1. We started the IBM Configuration Assistant for z/OS Communication Server. In the Main Perspective panel, we clicked the **Add a New z/OS image** option.
2. We entered the name of the z/OS image (in our case, we entered `SC33`) and clicked **OK**.
3. In the Main Perspective panel, we clicked the **Add New TCP/IP stack** option, and entered the name of the TCP/IP stack (in our case, we entered `TCPIPD`), then clicked **OK**.
4. In the Main Perspective panel, we selected AT-TLS in the z/OS Communications Server technologies list. We clicked **Enable**, then clicked the **Configure** option.

5. This leads to the AT-TLS Perspective panel. Notice the stack in the left pane shows Incomplete Stack, because the AT-TLS policy is not yet configured.

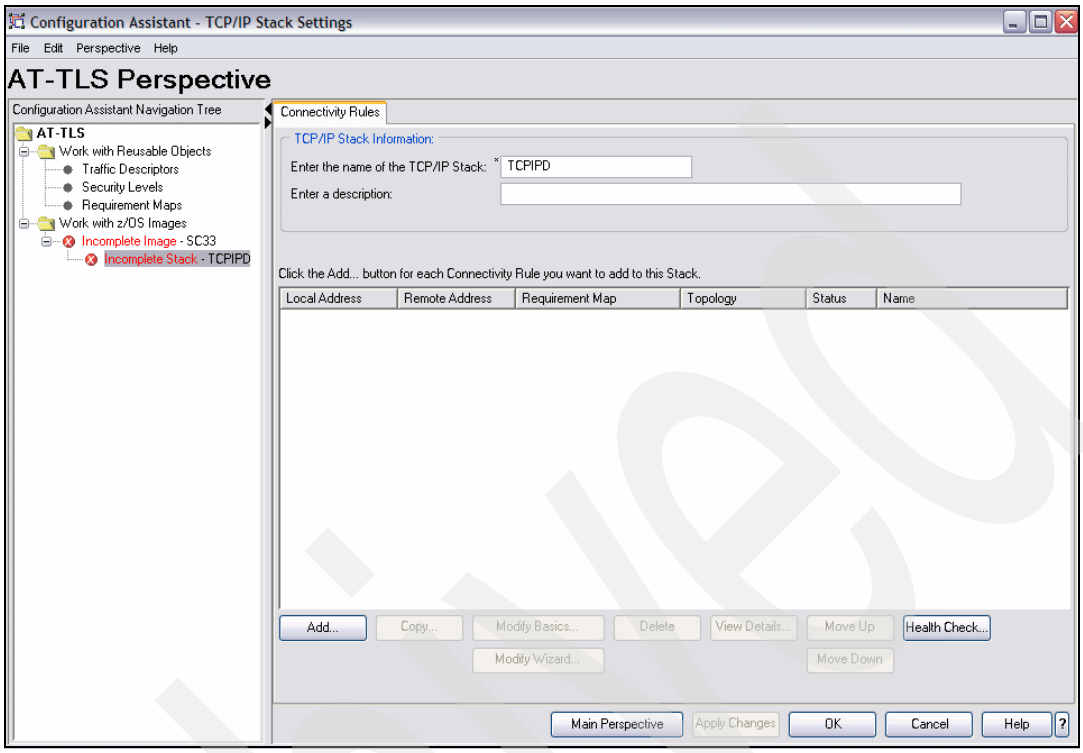


Figure B-3 AT-TLS Perspective panel

6. We selected **Work with Reusable Objects** -> **Traffic Descriptors** on the left pane of the AT-TLS Perspective panel. We created a new traffic descriptor to use port 4992 for the TN3270 server, instead of using the predefined traffic descriptors, then we clicked **Add**.
7. In the New Traffic Descriptor panel, we entered the name of the new traffic descriptor (we named it `ATTLS_TN3270D_4992`) and clicked **Add**. We defined the local port number and remote port number (in our example we applied the policy to the TN3270 server with TCP port number 4992, so we specified a local port number `4992` and the jobname `TN3270D`). We clicked the **AT-TLS Advanced** option.

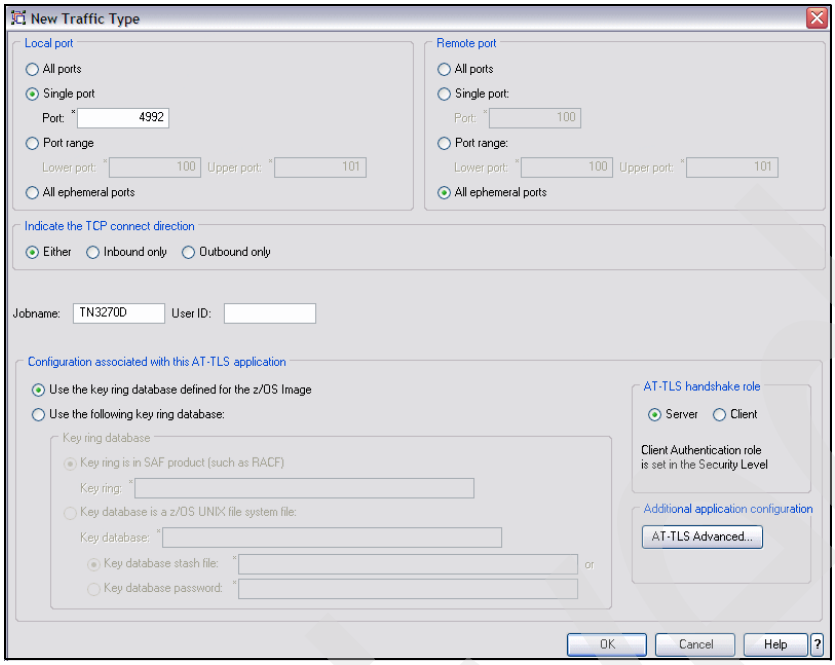


Figure B-4 New Traffic Type panel: Defining the traffic description

8. In the Key Ring and Advanced AT-TLS settings panel, we selected the **AT-TLS Tuning** tab, and toggled the **Application Controlled** option to On, then we clicked **OK**.

> **Note:** The TN3270 profile SSL handshake timeout (SSLTIMEOUT) is 5 seconds by default with native TLS support. The AT-TLS handshake timeout is 10 seconds by default. If you want to make them match, specify the AT-TLS handshake timeout parameter in the AT-TLS Tuning tab.

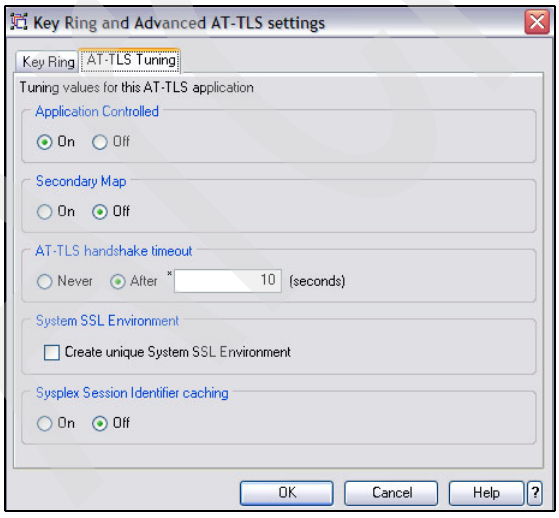


Figure B-5 Key Ring and Advanced AT-TLS settings panel

9. When back on the New Traffic Type panel, we clicked **OK**. The traffic descriptor just created is now shown in the list in the New Traffic Descriptor panel.
10. We selected **Work with Reusable Objects** -> **Security Levels** on the left pane of the AT-TLS Perspective panel. We created two security levels. One security level uses no

client authentication (applied for ADMIN_SSLPLAIN and SHIPPING_UP2USER rule). The other security level requires client authentication (applied to the PAYROLL_SSLCERT rule), which is equivalent to CLIENTAUTH SSLCERT in TN3270 profile for native TLS support.

11. We clicked **Add** to define a security level with no client authentication. In the New Security Level: Names and Type panel, we entered the name of the security level (we named the security level with no client authentication Gold_NoClientAuth). Click **Next**.

    In the New Security Level: Cipher Selections panel, we selected **Use Only Selected Ciphers** option and clicked the **Choose Ciphers** option. We selected the ciphers 0x0A and 0x2F, which are same selections as the IBM-supplied default AT-TLS_Gold shown in Table B-2. Click **OK**.

Table B-2 IBM-supplied security levels

Security level	Type	Entire TLS Version 1/ SSL Version 3 Cipher Suite in preferred order
Permit	No security	N/A
AT-TLS_Bronze (Low level of protection)	AT-TLS	0x02 - TLS_RSA_WITH_NULL_SHA
AT-TLS_Silver (Medium level of protection)	AT-TLS	0x09 - TLS_RSA_WITH_DES_CBC_SHA 0x0A - TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F - TLS_RSA_WITH_AES_128_CBC_SHA
AT-TLS_Gold (High level of protection)	AT-TLS	0x0A - TLS_RSA_WITH_3DES_EDE_CBC_SHA 0x2F - TLS_RSA_WITH_AES_128_CBC_SHA
AT-TLS_Platinum (Extremely high level of protection)	AT-TLS	0x35 - TLS_RSA_WITH_AES_256_CBC_SHA

Figure B-6 shows the menu for selecting ciphers.

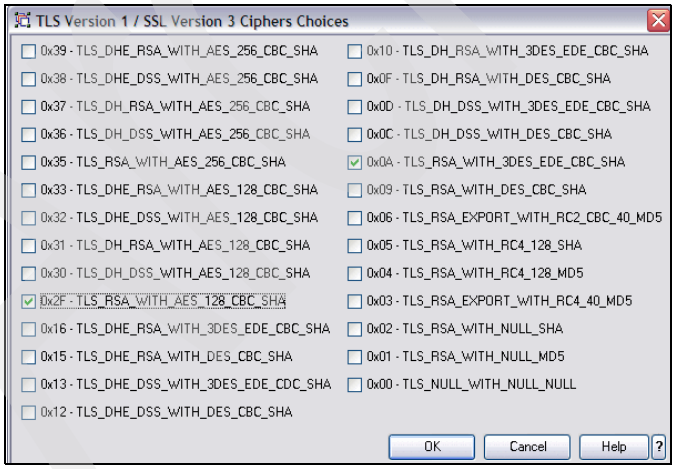


Figure B-6 Selecting ciphers

12. Back on the New Security Level panel, we clicked **Next**. In the next panel, we selected the **Advanced Settings** option and the **Client Authentication** tab. We made sure **No client authentication** is selected. Then we clicked **OK** to return to the New Security Level panel, and clicked **Finish**.

13. We defined another security level with client authentication required. by following the same instructions shown from step 11 to step 12. This time we named the security level Gold_ClientAuthSSLCert and selected the **Use client authentication** option with **Required** in the Advanced AT-TLS Settings panel.

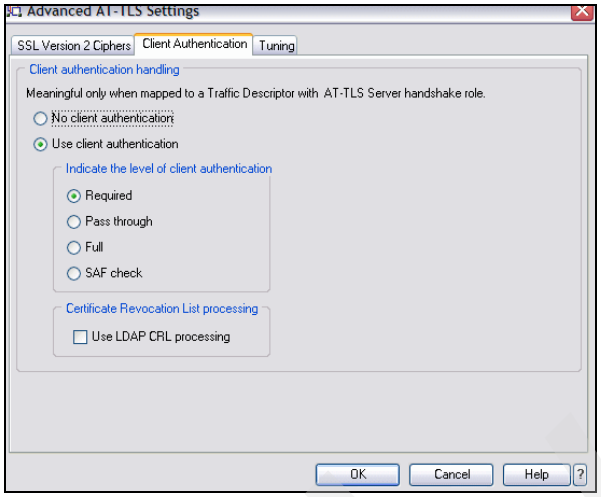


Figure B-7 Advanced AT-TLS Settings panel: defining client authentication

14. In the AT-TLS Perspective panel, we selected **Work with Reusable Objects** -> **Requirement Maps** on the left pane. We created two requirement maps that map the traffic descriptors and the security levels.

15. For the first requirement map, we mapped the ATTLS_TN3270D_4992 traffic descriptor and the Gold_NoClientAuth security level, and then clicked **Add**. We entered the name of the requirement map. We named it SSLPLAIN_ReqMap, and clicked **Next**.

    In New Requirement Map panel, we selected ATTLS_TN3270D_4992 from the Objects list, and clicked the **<--Add** option. From the AT-TLS - Security Level list, we selected Gold_NoClientAuth, and then clicked **OK**. When the Requirement Map Tip panel appeared, we clicked **Proceed**.

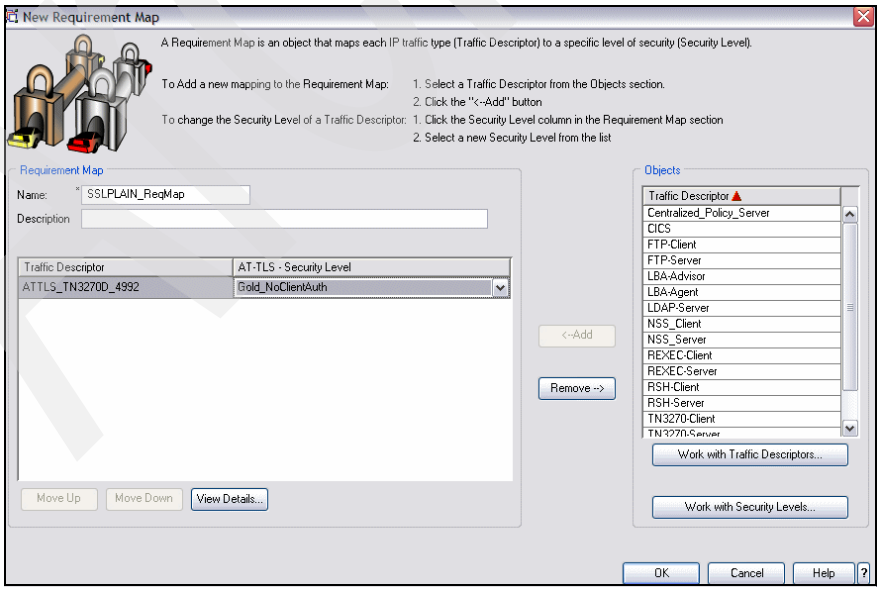


Figure B-8 New Requirement Map panel

16. For the second requirement map, we mapped the ATTLS_TN3270D_4992 traffic descriptor and the Gold_ClientAuthSSLCert security level, then clicked **Add**. We entered the name of the requirement map (we named it `SSLCERT_ReqMap`), and clicked **Next**.

    In the New Requirement Map panel, we selected ATTLS_TN3270D_4992 from the Objects list and clicked the **<--Add** option. From the AT-TLS - Security Level list, we selected Gold_ClientAuthSSLCert and clicked **OK**. When the Requirement Map Tip panel appeared, we clicked **Proceed**.

17. Next, we put all the defined objects together into the connectivity rules. In the AT-TLS Perspective panel, we selected the TCP/IP stack name (`TCPIPD`, in our case) in the left pane.

18. For the first connectivity rule, we made the definition for the ADMIN_SSLPLAIN rule. We clicked **Add**. In the New Connectivity Rule: Data Endpoints panel, we defined 10.1.8.42 for the destination IP address and selected **All IP V4 Addresses** for the source IP address. We entered the name of the rule (`ADMIN_SSLPLAIN`, in our case), and clicked **Next**. In the New Connectivity Rule: Select Requirement Map panel, we selected SSLPLAIN_ReqMap, and clicked **Next**. In the next panel, we clicked **Finish**.

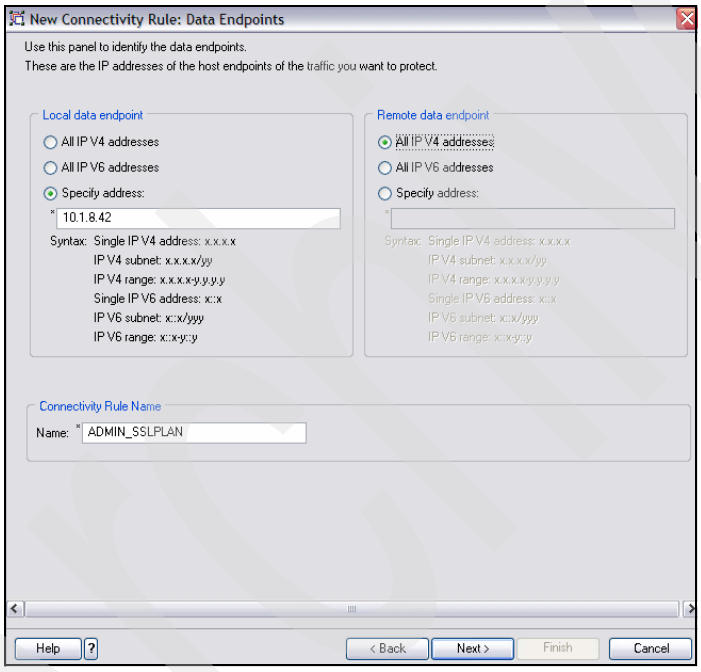


Figure B-9 New Connectivity Rule: Data Endpoints panel

19. For the second connectivity rule, we made the definition for the PAYROLL_SSLCERT rule. We clicked **Add**. In the New Connectivity Rule: Data Endpoints panel, we defined10.1.8.43 for the destination IP address and selected **All IP V4 Addresses** for source IP address. We entered the name of the rule (`PAYROLL_SSLCERT`, in our case), and clicked **Next**. In the New Connectivity Rule: Select Requirement Map panel, we selected SSLCERT_ReqMap and clicked **Next**. In the next panel, we clicked **Finish**.

20. For the third connectivity rule, we made the definition for the SHIPPING_UP2USER group. We clicked **Add**. In the New Connectivity Rule: Data Endpoints panel, we defined10.1.1.40 for the destination IP address and selected **All IP V4 Addresses** for source IP address. We entered the name of the rule (`SHIPPING_UP2USER`, in our case), and clicked **Next**. In the New Connectivity Rule: Select Requirement Map panel, we selected SSLPLAIN_ReqMap and clicked **Next**. In the next panel, we clicked **Finish**.

21. The AT-TLS Perspective panel displayed the defined connectivity rule, but still we still saw the Incomplete Image in the left pane because the Key Ring was not defined yet. We clicked the **Apply Changes** option to save the configuration.

22. In the left pane, we selected the image name (`SC33`, in our case). The required AT-TLS Image Level Settings should be displayed.

    If they are not displayed, select the **Image Level Settings** tab in the right pane. Specify the key ring name or the key database name that you created in “Create a certificate” on page 689.

    We specified `SharedRing1` in our example, and clicked **OK**.

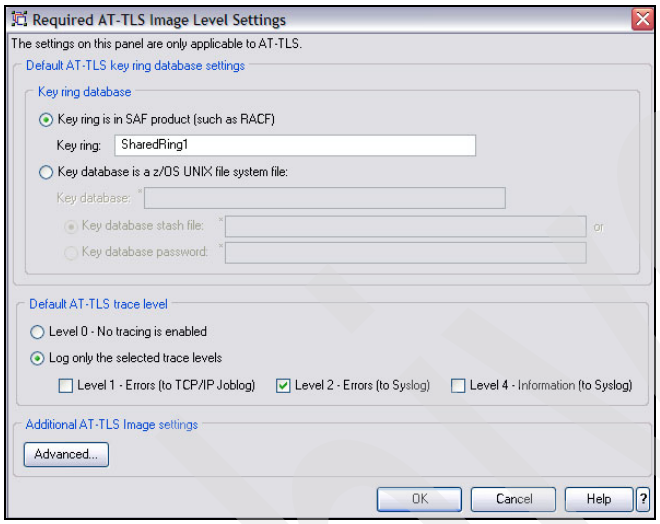


Figure B-10 AT-TLS Image Level Settings panel: specifying the key ring

23. The policy definition was now complete. The connectivity rules defined in the AT-TLS Perspective panel were displayed; Example 11-17 shows the configuration file generated by the IBM Configuration Assistant for z/OS Communication Server.

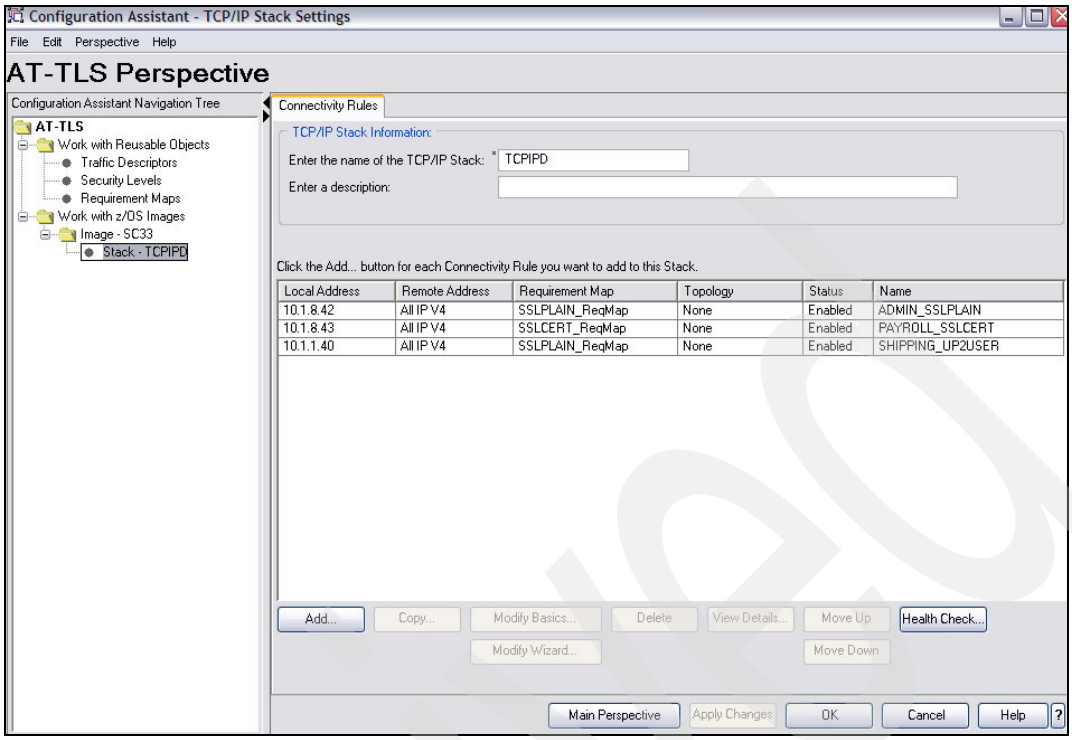


Figure B-11 AT-TLS Perspective panel with defined connectivity rules

Example B-17 shows the AT-TLS policy configuration file.

Example: B-17 AT-TLS policy configuration file

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R9\files\attls_scenario2
## FTP History:
## 2007-09-25 10:33:32  cs06 to 10.1.1.40
##
TTLSRule                          ADMIN_SSLPLAIN~1
{
  LocalAddrRef                    addr1
  RemoteAddrSetRef                addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Jobname                         TN3270D
  Direction                       Both
  Priority                        255
  TTLSGroupActionRef              gAct1~ATTLS_TN3270D_4992
  TTLSEnvironmentActionRef        eAct1~ATTLS_TN3270D_4992
  TTLSConnectionActionRef         cAct1~ATTLS_TN3270D_4992
}
TTLSRule                          PAYROLL_SSLCERT~2
{
  LocalAddrRef                    addr3
  RemoteAddrSetRef                addr2
```

```
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Jobname                         TN3270D
  Direction                       Both
  Priority                        254
  TTLSGroupActionRef              gAct1~ATTLS_TN3270D_4992
  TTLSEnvironmentActionRef        eAct2~ATTLS_TN3270D_4992
  TTLSConnectionActionRef         cAct2~ATTLS_TN3270D_4992
}
TTLSRule                          SHIPPING_UP2USER~3
{
  LocalAddrRef                    addr4
  RemoteAddrSetRef                addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Jobname                         TN3270D
  Direction                       Both
  Priority                        253
  TTLSGroupActionRef              gAct1~ATTLS_TN3270D_4992
  TTLSEnvironmentActionRef        eAct1~ATTLS_TN3270D_4992
  TTLSConnectionActionRef         cAct1~ATTLS_TN3270D_4992
}
TTLSGroupAction                   gAct1~ATTLS_TN3270D_4992
{
  TTLSEnabled                     On
}
TTLSEnvironmentAction             eAct1~ATTLS_TN3270D_4992
{
  HandshakeRole                   Server
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~SC33
}
TTLSEnvironmentAction             eAct2~ATTLS_TN3270D_4992
{
  HandshakeRole                   ServerWithClientAuth
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~SC33
}
TTLSConnectionAction              cAct1~ATTLS_TN3270D_4992
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1
  TTLSConnectionAdvancedParmsRef  cAdv1~ATTLS_TN3270D_4992
}
TTLSConnectionAction              cAct2~ATTLS_TN3270D_4992
{
  HandshakeRole                   ServerWithClientAuth
  TTLSCipherParmsRef              cipher1
  TTLSConnectionAdvancedParmsRef  cAdv1~ATTLS_TN3270D_4992
}
TTLSConnectionAdvancedParms       cAdv1~ATTLS_TN3270D_4992
{
  ApplicationControlled           On
}
TTLSKeyringParms                  keyR~SC33
{
  Keyring                         SharedRing1
}
TTLSCipherParms                   cipher1
{
```

```
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            10.1.8.42
}
IpAddrSet                         addr2
{
  Prefix                          0.0.0.0/0
}
IpAddr                            addr3
{
  Addr                            10.1.8.43
}
IpAddr                            addr4
{
  Addr                            10.1.1.40
}
PortRange                         portR1
{
  Port                            4992
}
PortRange                         portR2
{
  Port                            1024-65535
}
```

Upload the policy to z/OS

To upload the policy to z/OS, follow these steps:

1. In the Configuration Assistant Navigation Tree pane, right-click the TCP/IP stack name (TCPIPD, in our example), and select the **Install Configuration Files** option.
2. Select the *stackname* - AT-TLS: Policy Agent Stack Configuration in the list and click the **FTP** option. If you do not want to use the FTP option to upload the configuration file directly to z/OS, select the **Show Configuration File** option instead and then select the **Save As** option to save it locally for later transfer.
3. Enter the FTP server IP address, port number, FTP user ID, and password. Also specify the filename to be saved as, and then click **Send**. When the FTP transfer is completed, the list in the Installation panel informs the transferred date and time in the **Sent** column.

Modify the policy agent configuration file

Example B-18 shows the main configuration file of the policy agent. It defines the stack-specific configuration file name for the TCPIPD stack.

Example: B-18 Main configuration file of policy agent

```
#  ******************************************************************
#  /SC33/etc/pagent33.conf
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent33_TCPIPD.conf  FLUSH PURGE 600
```

Example B-19 shows the stack-specific configuration file for the TCPIPD stack. Add the TTLSConfig statement that points to the policy configuration file we created.

Example: B-19 Stack-specific configuration file of policy agent

```
#  ******************************************************************
```

```
#  /SC33/etc/pagent33_TCPIPD.conf
#  ****************************************************************
TTLSConfig     /etc/pagent33_TTLS.conf FLUSH PURGE
```

Define AT-TLS port in TN3270 configuration file

To enable AT-TLS security, specify the TTLSPORT statement. In Example B-20, TCP port 4992 is used for accepting the secure connections with AT-TLS.

Example: B-20 TELNETPARMS definition for AT-TLS port in TN3270 configuration file

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPD
    TELNETDEVICE IBM-3277     SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3278-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2-E SNX32702,SNX32702
    TELNETDEVICE IBM-3279-2   SNX32702,SNX32702
    TELNETDEVICE IBM-3278-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3278-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3-E SNX32703,SNX32703
    TELNETDEVICE IBM-3279-3   SNX32703,SNX32703
    TELNETDEVICE IBM-3278-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3278-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4-E SNX32704,SNX32704
    TELNETDEVICE IBM-3279-4   SNX32704,SNX32704
    TELNETDEVICE IBM-3278-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3278-5   SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5-E SNX32705,SNX32705
    TELNETDEVICE IBM-3279-5   SNX32705,SNX32705
ENDTELNETGLOBALS
;
TELNETPARMS
    TTLSPORT 4992 1 ;Port 4992 supports AT-TLS secure connections
;   KEYRING SAF TCPIP/SharedRing1 2 ; omit - defined in AT-TLS policy
    CONNTYPE SECURE 3 ; Default conntype
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 4992 1
  DEFAULTLUS
     SC33DT01..SC33DT99
  ENDDEFAULTLUS

; ------------------------------------------------------------------------
; This NOSSL        group is mapped to use no SSL security.            -
; ------------------------------------------------------------------------
  PARMSGROUP NOSSL
     NOLUSESSIONPEND
     CONNTYPE BASIC  ; support non-secure, overrides telnetparms
  ENDPARMSGROUP
```

```
; ------------------------------------------------------------------
; The SSLPLAIN     group is mapped to use SSL security               -
;                  with no Client Authentication required           -
; ------------------------------------------------------------------
; PARMSGROUP SSLPLAIN 4 ; omit - defined in AT-TLS policy
;    CONNTYPE SECURE
; ENDPARMSGROUP

; ------------------------------------------------------------------
; The SSLCERTS     group is mapped to use SSL security               -
;                  and to require Client Authentication (certificates) -
; ------------------------------------------------------------------
; PARMSGROUP SSLCERTS 5 ; omit - defined in AT-TLS policy
;    CONNTYPE SECURE
;    CLIENTAUTH SSLCERT 2
;    ENCRYPT SSL_DES_SHA 2
;            SSL_3DES_SHA
;    ENDENCRYPT
; ENDPARMSGROUP

; ------------------------------------------------------------------
; The UP2USER      group is mapped to use ANY security (user's choice) -
;                  with no Client Authentication (no certificates)     -
; ------------------------------------------------------------------
  PARMSGROUP UP2USER 6
    CONNTYPE ANY         ; User choose secure or non-secure connection
 ENDPARMSGROUP

  DESTIPGROUP GENERALUSER 10.1.8.41     ENDDESTIPGROUP
; DESTIPGROUP ADMIN       10.1.8.42     ENDDESTIPGROUP ; omit - defined in AT-TLS policy
; DESTIPGROUP PAYROLL     10.1.8.43     ENDDESTIPGROUP ; omit - defined in AT-TLS policy
  DESTIPGROUP SHIPPING    10.1.1.40     ENDDESTIPGROUP

  PARMSMAP NOSSL        DESTIPGRP,GENERALUSER 7
  DEFAULTAPPL SC33TS    DESTIPGRP,GENERALUSER

; PARMSMAP SSLPLAIN     DESTIPGRP,ADMIN    ; omit - defined in AT-TLS policy 4
; USSTCP  USSTEST1   DESTIPGRP,ADMIN   ; omit - defined in AT-TLS policy

; PARMSMAP SSLCERTS     DESTIPGRP,PAYROLL  ; omit - defined in AT-TLS policy 5
; DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL  ; omit - defined in AT-TLS policy

  PARMSMAP UP2USER         DESTIPGRP,SHIPPING 6
  USSTCP USSTABEE,USSSNAEE DESTIPGRP,SHIPPING

  USSTCP   USSTEST1      ; Default USSTAB

  ALLOWAPPL SC3*           ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
;
TELNETPARMS
    PORT 23        ; Port 23 supports basic (non-secure) connections
    INACTIVE 0
    TIMEMARK 600
    SCANINTERVAL 120
    FULLDATATRACE
```

```
    SMFINIT 0  SMFINIT NOTYPE119
    SMFTERM 0  SMFTERM TYPE119
    SNAEXT
    MSG07
    LUSESSIONPEND
ENDTELNETPARMS
;
BEGINVTAM
  PORT 23
  DEFAULTLUS
      SC33DB01..SC33DB99
  ENDDEFAULTLUS

  DEFAULTAPPL SC33TS        ; All users go to TSO
  ALLOWAPPL SC*             ; Netview and TSO
  ALLOWAPPL NVAS* QSESSION  ; session mngr queues back upon CLSDST
  ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
  ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
ENDVTAM
```

1 Port 4992 is used for the AT-TLS secure connection.

2 These security parameters are omitted from the TN3270 profile because they are defined in the policy agent.

3 Default connection type is secure.

4 The PARMSGROUP and PARMSMAP statements for ADMIN are omitted because they are defined in the policy agent.

5 The PARMSGROUP and PARMSMAP statements for PAYROLL are omitted because they are defined in the policy agent.

6 The PARMSGROUP and PARMSMAP statements for SHIPPING are specified to override the CONNTYPE to ANY.

Activation and verification

Perform the following tasks to activate and verify TN3270 AT-TLS support:

- Start the policy agent.
- Start the TN3270 server.
- Display PROF to show profile AT-TLS information.
- Display the AT-TLS profile using the pasearch command.
- Display CONN to show connection AT-TLS information.

Start the policy agent

Start the policy agent to enable the policy-based routing as shown in Example B-21. The message **1** shows the AT-TLS policy is processed and now in effect.

Example: B-21 Starting the policy agent

```
S PAGENT
$HASP100 PAGENT   ON STCINRDR
IEF695I START PAGENT   WITH JOBNAME PAGENT   IS ASSIGNED TO USER
PAGENT  , GROUP TCPGRP
$HASP373 PAGENT   STARTED
EZZ8431I PAGENT STARTING
```

```
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS 1
```

If you have the policy agent started already, use the F *jobname*,REFRESH command as shown in Example B-22. The message 2 informs you that the AT-TLS policy is loaded (or reloaded).

Example: B-22 Refreshing the policy agent

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS 2
```

Start the TN3270 server

Start the TN3270 server. Ensure that TN3270 server starts listening on the TTLS port 1.

Example: B-23 Starting the TN3270 Server

```
S TN3270D
$HASP100 TN3270D  ON STCINRDR
.....
EZZ6001I TELNET SERVER STARTED
.....
EZZ6003I TELNET LISTENING ON PORT  4992 1
EZZ6003I TELNET LISTENING ON PORT   992
EZZ6003I TELNET LISTENING ON PORT    23
```

Display PROF to show profile AT-TLS information

Verify the profile is set up correctly. Use the Display Telnet,Profile command to display the profile.

Example: B-24 Telnet Profile Display

```
D TCPIP,TN3270D,T,PROF,PORT=4992,DET
EZZ6080I TELNET PROFILE DISPLAY 042
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* CURR
 SECURITY
   TTLSPORT             4992
   CONNTYPE           SECURE
   KEYRING            TTLS 1
   CRLLDAPSERVER      TTLS 1
   ENCRYPTION         TTLS 1
   CLIENTAUTH         TTLS 1
   NOEXPRESSLOGON
   NONACUSERID
   SSLV2              TTLS 1
 TIMERS
.....
   SSLTIMEOUT           TTLS 1
.....
   KEYRING            SAF TCPIP/SharedRing1 2
```

1 The AT-TLS policy is used for security parameters.

2 The key ring name specified in the AT-TLS policy configuration file is applied.

Display the AT-TLS profile using the pasearch command

The **pasearch** command can be used to display the AT-TLS policy configuration.

Example: B-25 pasearch -t display

```
CS06 @ SC33:/u/cs06>CS06 @ SC33:/u/cs06>pasearch -t

TCP/IP pasearch CS V1R9                  Image Name: TCPIPD
  Date:                 09/25/2007       Time:  11:00:18
  TTLS Instance Id:     1190732223

policyRule:             ADMIN_SSLPLAIN~1
  Rule Type:            TTLS
  Version:              3                Status:           Active
  Weight:               255              ForLoadDist:      False
  Priority:             255              Sequence Actions: Don't Care
  No. Policy Action:    3
  policyAction:         gAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Group
   Action Sequence:     0
  policyAction:         eAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Environment
   Action Sequence:     0
  policyAction:         cAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Connection
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00            To TimeOfDay:     24:00
   Fr TimeOfDay UTC:    04:00            To TimeOfDay UTC: 04:00
   TimeZone:            Local
  TTLS Condition Summary:                NegativeIndicator: Off
   Local Address:
    FromAddr:           10.1.8.42
    ToAddr:             10.1.8.42
   Remote Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   LocalPortFrom:       4992             LocalPortTo:      4992
   RemotePortFrom:      1024             RemotePortTo:     65535
   JobName:             TN3270D          UserId:
   ServiceDirection:    Both

  TTLS Action:                  gAct1~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Group
    TTLSEnabled:                On
    CtraceClearText:            Off
    Trace:                      2
```

```
    TTLSGroupAdvancedParms:
     SecondaryMap:              Off
     SyslogFacility:            Daemon

  TTLS Action:                  eAct1~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Environment
    HandshakeRole:              Server
    TTLSKeyringParms:
     Keyring:                   SharedRing1
    TTLSEnvironmentAdvancedParms:
     SSLv2:                     Off
     SSLv3:                     On
     TLSv1:                     On
     ApplicationControlled:     Off
     HandshakeTimeout:          10
     ClientAuthType:            Required
     ResetCipherTimer:          0
    EnvironmentUserInstance:    0

  TTLS Action:                  cAct1~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Connection
    HandshakeRole:              Server
    TTLSConnectionAdvancedParms:
     ApplicationControlled:     On
    TTLSCipherParms:
     v3CipherSuites:
      0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
      2F  TLS_RSA_WITH_AES_128_CBC_SHA

policyRule:             PAYROLL_SSLCERT~2
  Rule Type:            TTLS
  Version:              3                 Status:            Active
  Weight:               254               ForLoadDist:       False
  Priority:             254               Sequence Actions:  Don't Care
  No. Policy Action:    3
  policyAction:         gAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Group
   Action Sequence:     0
  policyAction:         eAct2~ATTLS_TN3270D_4992
   ActionType:          TTLS Environment
   Action Sequence:     0
  policyAction:         cAct2~ATTLS_TN3270D_4992
   ActionType:          TTLS Connection
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
   Month of Yr Mask:    111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
   Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC:  04:00
   TimeZone:            Local
  TTLS Condition Summary:                 NegativeIndicator: Off
```

```
   Local Address:
    FromAddr:           10.1.8.43
    ToAddr:             10.1.8.43
   Remote Address:
    FromAddr:           0.0.0.0
    Prefix:             0
   LocalPortFrom:       4992              LocalPortTo:       4992
   RemotePortFrom:      1024              RemotePortTo:      65535
   JobName:             TN3270D           UserId:
   ServiceDirection:    Both

  TTLS Action:                  gAct1~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Group
    TTLSEnabled:                On
    CtraceClearText:            Off
    Trace:                      2
    TTLSGroupAdvancedParms:
     SecondaryMap:              Off
     SyslogFacility:            Daemon

  TTLS Action:                  eAct2~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Environment
    HandshakeRole:              ServerWithClientAuth
    TTLSKeyringParms:
     Keyring:                   SharedRing1
    TTLSEnvironmentAdvancedParms:
     SSLv2:                     Off
     SSLv3:                     On
     TLSv1:                     On
     ApplicationControlled:     Off
     HandshakeTimeout:          10
     ClientAuthType:            Required
     ResetCipherTimer:          0
    EnvironmentUserInstance:    0

  TTLS Action:                  cAct2~ATTLS_TN3270D_4992
    Version:                    3
    Status:                     Active
    Scope:                      Connection
    HandshakeRole:              ServerWithClientAuth
    TTLSConnectionAdvancedParms:
     ApplicationControlled:     On
    TTLSCipherParms:
     v3CipherSuites:
      0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
      2F  TLS_RSA_WITH_AES_128_CBC_SHA

policyRule:             SHIPPING_UP2USER~3
  Rule Type:            TTLS
  Version:              3                 Status:            Active
  Weight:               253               ForLoadDist:       False
  Priority:             253               Sequence Actions:  Don't Care
  No. Policy Action:    3
  policyAction:         gAct1~ATTLS_TN3270D_4992
   ActionType:          TTLS Group
   Action Sequence:     0
```

```
policyAction:         eAct1~ATTLS_TN3270D_4992
 ActionType:          TTLS Environment
 Action Sequence:     0
policyAction:         cAct1~ATTLS_TN3270D_4992
 ActionType:          TTLS Connection
 Action Sequence:     0
Time Periods:
 Day of Month Mask:
 First to Last:       11111111111111111111111111111111
 Last to First:       11111111111111111111111111111111
 Month of Yr Mask:    111111111111
 Day of Week Mask:    1111111  (Sunday - Saturday)
 Start Date Time:     None
 End Date Time:       None
 Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
 Fr TimeOfDay UTC:    04:00             To TimeOfDay UTC:  04:00
 TimeZone:            Local
TTLS Condition Summary:                 NegativeIndicator: Off
 Local Address:
  FromAddr:           10.1.1.40
  ToAddr:             10.1.1.40
 Remote Address:
  FromAddr:           0.0.0.0
  Prefix:             0
 LocalPortFrom:       4992              LocalPortTo:       4992
 RemotePortFrom:      1024              RemotePortTo:      65535
 JobName:             TN3270D           UserId:
 ServiceDirection:    Both

TTLS Action:                  gAct1~ATTLS_TN3270D_4992
  Version:                    3
  Status:                     Active
  Scope:                      Group
  TTLSEnabled:                On
  CtraceClearText:            Off
  Trace:                      2
  TTLSGroupAdvancedParms:
   SecondaryMap:              Off
   SyslogFacility:            Daemon

TTLS Action:                  eAct1~ATTLS_TN3270D_4992
  Version:                    3
  Status:                     Active
  Scope:                      Environment
  HandshakeRole:              Server
  TTLSKeyringParms:
   Keyring:                   SharedRing1
  TTLSEnvironmentAdvancedParms:
   SSLv2:                     Off
   SSLv3:                     On
   TLSv1:                     On
   ApplicationControlled:     Off
   HandshakeTimeout:          10
   ClientAuthType:            Required
   ResetCipherTimer:          0
  EnvironmentUserInstance:    0

TTLS Action:                  cAct1~ATTLS_TN3270D_4992
  Version:                    3
  Status:                     Active
```

```
   Scope:                       Connection
   HandshakeRole:               Server
   TTLSConnectionAdvancedParms:
    ApplicationControlled:      On
   TTLSCipherParms:
    v3CipherSuites:
     0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
     2F  TLS_RSA_WITH_AES_128_CBC_SHA
```

Display CONN to show connection AT-TLS information

We used IBM Personal Communications (PComm) V5.7 to establish a secure session. Because we used self-signed certificate, we downloaded and installed the certificate of the Certificate Authority into the PComm. We defined a PComm connection profile for the AT-TLS connection with exactly same security parameters that we defined for the TN3270 native TLS secure connection.

The client group with destination IP address 10.1.8.42 was defined as CONNTYPE SECURE (default) and supported only secure connections. Example B-26 shows a secure connection using destination IP address 10.1.8.42.

Example: B-26 TLS secure connection using port 4992 and destination IP address 10.1.8.42

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 002
         EN                                        TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
000099CB 0A ::FFFF:10.1.100.224..1555
                                  SC33DT02 SC33TS06  TAE SNX32702
----- PORT:  4992 1 ACTIVE              PROF: CURR CONNS:     1
------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED

D TCPIP,TN3270D,T,CONN,CONN=99CB,DET
EZZ6065I TELNET CONNECTION DISPLAY 004
  CONNECTED: 12:03:34  09/25/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 000099CB   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..1555
    DESTIP..PORT: ::FFFF:10.1.8.42..4992 1
    LINKNAME: VIPL0A01082A
  PORT:  4992 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  TTLSSECURE    ACCESS: SECURE  0A TLSV1 2
     TTLSRULE: ADMIN_SSLPLAIN~1 3
     TTLSGRPACTION: gAct1~ATTLS_TN3270D_4992
     TTLSENVACTION: eAct1~ATTLS_TN3270D_4992
     TTLSCONNACTION: cAct1~ATTLS_TN3270D_4992
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                        NEWENV FUNCTIONS: --
  LUNAME: SC33DT02
  APPL: SC33TS06
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                      OBJECT   ITEM SPECIFIC     OPTIONS
```

```
   LUMAP GEN:  NL (NULL)
                    >*DEFLUS*                    --------
   DEFLT APPL: **N/A**
   USS TABLE:  NL (NULL)
                    >USSTEST1                    P-------
   INT TABLE:  **N/A**
   PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* TP-CURR
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* <-FINAL 4
39 OF 39 RECORDS DISPLAYED
```

1 AT-TLS port 4992 is used.

2 The connection is secure and uses cipher 0A (TLS_RSA_WITH_3DES_EDE_CBC_SHA).

3 The connection used the ADMIN_SSLPLAIN connectivity rule.

4 The security parameters were specified in the AT-TLS policy.

The client ID group with destination IP address 10.1.1.40 was defined CONNTYPE ANY and can support both secure and non-secure connections. Example B-27 shows a secure connection with a client requesting TLS which is destined to IP address 10.1.1.40.

Example: B-27 TLS secure connection using port 4992 and destination IP address 10.1.1.40

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 017
         EN                                        TSP
CONN     TY IPADDR..PORT           LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
000099D8 0A ::FFFF:10.1.100.224..1672
                                SC33DT03 SC33TS06  TAE SNX32702
----- PORT:  4992 1 ACTIVE              PROF: CURR CONNS:      1
------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED

D TCPIP,TN3270D,T,CONN,CONN=99D8,DET
EZZ6065I TELNET CONNECTION DISPLAY 025
  CONNECTED: 12:06:23  09/25/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 000099D8   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..1672
    DESTIP..PORT: ::FFFF:10.1.1.40..4992 1
    LINKNAME: VIPA1L
  PORT:  4992 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  TTLSSECURE    ACCESS: SECURE  0A TLSV1 2
     TTLSRULE: SHIPPING_UP2USER~3 3
     TTLSGRPACTION: gAct1~ATTLS_TN3270D_4992
     TTLSENVACTION: eAct1~ATTLS_TN3270D_4992
     TTLSCONNACTION: cAct1~ATTLS_TN3270D_4992
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
```

```
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                        NEWENV FUNCTIONS: --
  LUNAME: SC33DT03
  APPL: SC33TS06
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                      >*DEFLUS*                   --------
    DEFLT APPL: **N/A**
    USS TABLE:  DG SHIPPING
                       USSTABEE,>USSSNAEE         PP------
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* TP-CURR
    PARMSGROUP: DG SHIPPING
  L------  ------------  ---  -A-------  -------  ---- UP2USER
  LM*****  **TSBTQ***RT  ECF  TATTTT**T  *P**STT  SDD* <-FINAL 4
41 OF 41 RECORDS DISPLAYED
```

1 AT-TLS port 4992 is used.

2 The connection is secure and uses cipher 0A (TLS_RSA_WITH_3DES_EDE_CBC_SHA).

3 The connection used the SHIPPING_UP2USER connectivity rule.

4 The security parameters were specified in the AT-TLS policy.

Example B-28 shows a non-secure connection with a client requesting no security which is destined to IP address 10.1.1.40.

Example: B-28 TLS non-secure connection using port 4992 and destination IP address 10.1.1.40

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 031
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
000099E6    ::FFFF:10.1.100.224..1777
                                   SC33DT04 SC33TS06  TAE SNX32702
----- PORT:  4992 1 ACTIVE               PROF: CURR CONNS:      1
------------------------------------------------------------
4 OF 4 RECORDS DISPLAYED

D TCPIP,TN3270D,T,CONN,CONN=99E6,DET
EZZ6065I TELNET CONNECTION DISPLAY 033
  CONNECTED: 12:08:58  09/25/2007  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 000099E6   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.224..1777
    DESTIP..PORT: ::FFFF:10.1.1.40..4992 1
    LINKNAME: VIPA1L
```

```
  PORT:  4992 QUAL: NONE
    AFFINITY: TCPIPD
    STATUS: ACTIVE  TTLSSECURE    ACCESS: NON-SECURE 2
     TTLSRULE: SHIPPING_UP2USER~3 3
     TTLSGRPACTION: gAct1~ATTLS_TN3270D_4992
     TTLSENVACTION: eAct1~ATTLS_TN3270D_4992
     TTLSCONNACTION: cAct1~ATTLS_TN3270D_4992
  PROTOCOL: TN3270E           DEVICETYPE: IBM-3278-2-E
    TYPE: TERMINAL GENERIC
    OPTIONS: ETET----   3270E FUNCTIONS: BSR----
                         NEWENV FUNCTIONS: --
  LUNAME: SC33DT04
  APPL: SC33TS06
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED: SNX32702  APPL SPECIFIED: SNX32702
  MAPPING TYPE:  CONN IDENTIFIER
                       OBJECT   ITEM SPECIFIC     OPTIONS
    LUMAP GEN:  NL (NULL)
                      >*DEFLUS*                   --------
    DEFLT APPL: **N/A**
    USS TABLE:  DG SHIPPING
                       USSTABEE,>USSSNAEE         PP------
    INT TABLE:  **N/A**
    PARMS:
  PERSIS   FUNCTION      DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OATSKTQSWHRT)(DRF)(PCKLECXN2)(IPKPSTS)(SMLT)
  -------  ------------  ---  ---------  -------  ----
  *******  **TSBTQ***RT  EC*  BB**D****  *P**STS  *DD* *DEFAULT
  -------  -----------T  ---  ---------  -------  ---- *TGLOBAL
  -M-----  ---S--------  --F  TSTTTT--T  *---STT  S--- *TPARMS
  *M*****  **TSBTQ***RT  ECF  TSTTTT**T  *P**STT  SDD* TP-CURR
    PARMSGROUP: DG SHIPPING
  L------  ------------  ---  -A-------  -------  ---- UP2USER
  LM*****  **TSBTQ***RT  ECF  TATTTT**T  *P**STT  SDD* <-FINAL 4
41 OF 41 RECORDS DISPLAYED
```

1 AT-TLS port 4992 was used.

2 The connection was non-secure.

3 The connection used the SHIPPING_UP2USER connectivity rule.

4 The security parameters were specified in the AT-TLS policy (non-applicable, in this case).

C

Configuring IPSec between z/OS and Windows

This appendix contains the steps used for our IPSec scenario between z/OS and Windows. The following topics are discussed in this appendix.

Section	Topic
“IPSec between z/OS and Windows” on page 712	Setting up the IKE daemon; the certificates; the z/OS IPSec policy; and a Windows IPSec policy. Verifying that things are working.

For more details about our IPSec scenario between z/OS systems, refer to 9.5, “Configuring IPSec between two z/OS systems” on page 394.

IPSec between z/OS and Windows

Figure C-1 shows the setup we implemented in this scenario.

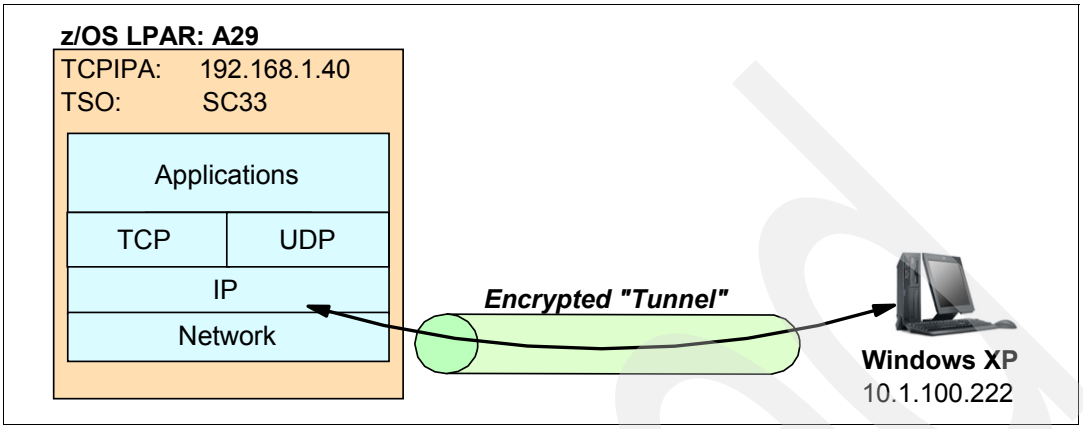


Figure C-1 VPN between z/OS and Windows

The steps to configure a VPN between z/OS image and Windows XP are:

1. Set up the IKE daemon.
2. Set up the certificates.
3. Set up the z/OS IPSec policy.
4. Set up a Windows IPSec policy.
5. Verify that things are working.

We explain these steps in more detail in the following sections.

Set up the IKE daemon

In terms of procedure, user ID, and other configuration choices, the IKE daemon in our case was set up using the same principles as outlined in 9.3.7, “Setting up the IKE daemon” on page 385.

The _CEE_ENVFILE environment variable was used to set up a STDENV DD card for controlling the environment variables. MVS data sets were used for all files. A sample of the cataloged procedure is shown in Example C-1.

Example: C-1 IKE daemon cataloged procedure

```
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD DUMMY
//STDENV   DD DSN=TCPIP.IKED.ENV(IKED),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

There is no need to specify a resolver configuration file here because, as is the case with the Policy Agent, one server should be used for all stacks within the image.

A sample config is supplied in the z/OS Communications Server installation file /usr/lpp/tcpip/samples/IBM/EZAIKCFG. The config we used is shown in Example C-2.

Example: C-2 IKE daemon config file

```
IkeConfig
```

```
{
 IkeSyslogLevel    127
 PagentSyslogLevel 127
 KeyRing           IKED/IKED_keyring
 KeyRetries        10
 KeyWait           30
 DataRetries       10
 DataWait          15
 Echo              no
 PagentWait        0
}
```

- IkeSyslogLevel and PagentSyslogLevel: These levels were set to 127 during testing to obtain helpful messages. After successful configuration, a much lower value should be used to avoid over-filling of the syslogd file.
- KeyRing: This is the RACF key ring name used to hold the certificate authority certificate required for ISAKMP/IKE authentication. This key ring was created using the user ID for the IKED started task, IKED. If the key ring had been created under any other user ID, then this user ID would need to be prepended to the key ring name using the syntax USERID/keyring.

  For example, in the IKE configuration file, we could have specified KeyRing IKED/IKED_keyring to complete the exact same effect. Note, however, that accessing a key ring that is owned by another user ID requires UPDATE access to IRR.DIGTCERT.LISTRING.

Set up the certificates

The next step is to establish a certificate environment for the key exchange. Both the z/OS host and the Windows XP host must have valid certificates configured in order for the IKE exchange to successfully establish a security association.

Because the key exchange mechanism of IKE involves a direct request of the certificate authority required, self-signed certificates were not an option. Instead, a certificate authority (CERTAUTH) certificate was created using RACF, and then this certificate was used to sign a personal certificate. Then a key ring was created and both certificates were connected. The personal certificate was connected as the default. The JCL we used is shown in Example C-3.

Example: C-3 JCL for defining our key ring and digital certificates

```
//CERTAUTH  JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTAUTH EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Step 1:                                                       *
//*     Create Certificate Authority Certificate for ITSO             *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)
  RACDCERT ID(IKED) addring(IKED_keyring)
  RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('SC33 RACF CA') O('IBM') -
           OU('ITSO') C('us')) WITHLABEL('SC33 RACF CA')
  RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('WINXP Client2') O('IBM') -
           OU('ITSO') C('us')) WITHLABEL('WINXP Client2') -
           KEYUSAGE(HANDSHAKE) ALTNAME(IP(10.1.100.222))
  RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('SC33 Server') O('IBM') -
           OU('ITSO') C('us')) WITHLABEL('SC33 Server') -
```

```
          SIGNWITH(CERTAUTH LABEL('SC33 RACF CA')) -
          KEYUSAGE(HANDSHAKE) ALTNAME(IP(192.168.1.40))
     SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

  RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('WINXP Client2') -
          RING(IKED_keyring) USAGE(personal))
  RACDCERT ID(IKED) CONNECT(CERTAUTH LABEL('SC33 RACF CA') -
          RING(IKED_keyring) USAGE(CERTAUTH))
  RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('SC33 Server') -
     RING(IKED_keyring) USAGE(PERSONAL))
/*
```

In order to keep the configuration scenario simple, the same certificate authority and personal certificate were used on the XP workstation. To do this, the certificates have to be exported from the RACF database into MVS data sets using the commands shown in Example C-4.

Example: C-4 Export job JCL

```
//EXPORT    JOB MSGCLASS=X,NOTIFY=&SYSUID
//EXPORT   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//*     Export the Self-signed Certificate Authority certificate      *
//*     from the RACF database in base-64 encoded format.  This is    *
//*     then FTP'd to the clients so that they can verify the server  *
//*     certificiates when passed in the SSL exchange.                *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  RACDCERT CERTAUTH EXPORT(LABEL('SC33 RACF CA')) -
     DSN('CS05.CERT.CACERT') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
  RACDCERT ID(IKED) EXPORT(LABEL('WINXP Client2')) -
     DSN('CS05.CERT.XPCERT') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
/*
```

Note that the certificate authority (signing) certificate does not require a private key. However, the personal certificate should include the private key, and hence the PKCS12DER format, along with a password, was used.

Set up the z/OS IPSec policy

We used the IBM Configuration Assistant for z/OS Communication Server to create our IPSec policy file for SC33 z/OS image.

We defined two connectivity rules for the TCP/IP stack of SC33:

- **Basic Services**: A rule that permits Resolver, DNS, IKE, and OMPROUTE-IP_V4 traffic between all IPv4 addresses to all IPv4 addresses.
- **All_Traffic_PFS**: A rule that creates a dynamic IPSEC tunnel for all other traffic types between the z/OS image IP address (192.168.1.40) and the Windows XP IP address (10.1.100.222).

For the **All_Traffic_PFS** connectivity rule, we choose the security level that we created in “Creating a security level for PFS” on page 381.

To obtain the same configuration we created, follow these steps:

1. Create a requirement map object called **All_Traffic_PFS** and **Basic_Services**.
2. For the TCPIPE stack of the z/OS image SC33, we created two connectivity rules, **All_Traffic_PFS_Rule** and **Basic_Services_Rule,** assigning to them the requirement maps **All_Traffic_PFS** and **Basic_Services**.
3. Verify that the IKE daemon settings are correct.
4. Update the IPSec policy of the z/OS image.

Creating requirement map objects

To create requirement map objects using z/OS Configuration Assistant, follow these steps:

1. Start the Configuration Assistant. In the Main Perspective panel, select **IPsec Technology** and click **Configure**. In the IPSec Perspective panel, click **Requirement Maps**.
2. In the Requirement Map panel, enter a name and description for the object (we entered `All_Traffic_PFS`). In addition, select **PFS** to be the IPSEC - Security Level of the All_other_traffic traffic descriptor, which you can do by selecting the list box of IPSec - Security Level, as shown in Figure C-2. Click **OK**.

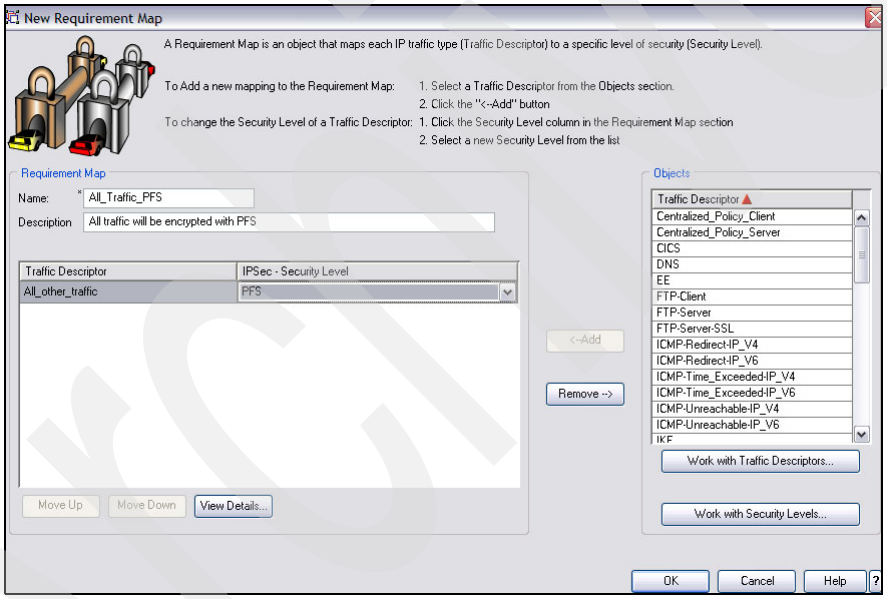


Figure C-2 Requirement Map panel: Adding All_Traffic_PFS requirement map object

3. In the Requirement Map Tip panel, click **Proceed** (we flagged this panel not to appear anymore).
4. In the Requirement Map panel, click **Add**.

5. In the Requirement Map panel:
   a. Enter a name and description for the object.
   b. Select the **All_other_traffic** traffic descriptor and click **Remove**.
   c. In the Objects traffic descriptor list, select **DNS** and then click **Add**.

      Repeat the same procedure for the following traffic descriptor objects: OMPROUTE-IP-4 and Resolver.
   d. Change the Security Level of all the objects to **Permit** by selecting the list box of IPSec - Security Level.

      The final configuration of this panel should like the configuration shown in Figure C-3.
   e. Click **OK**.

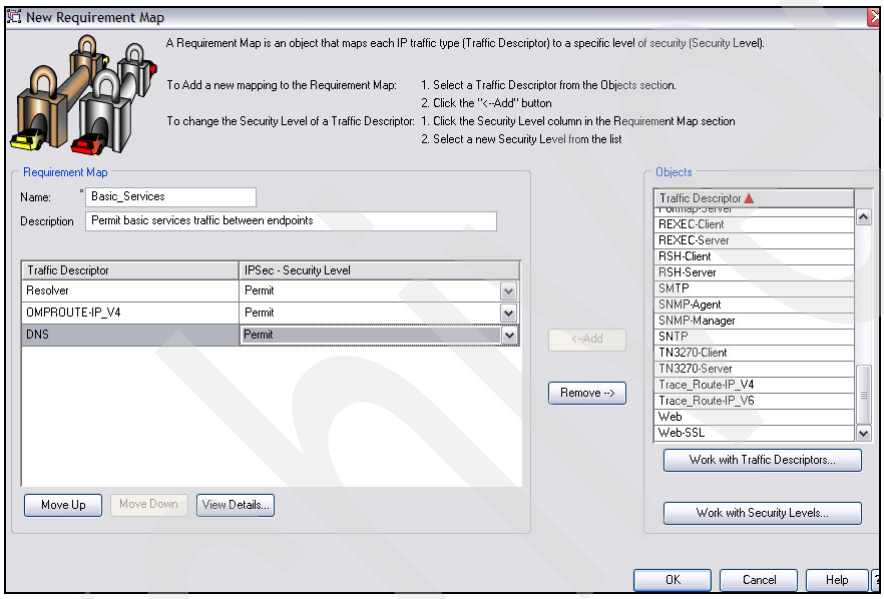


Figure C-3 Requirement Map panel: Adding Basic_Services object

The Requirement Maps panel should now include the new objects, as shown in Figure C-4.

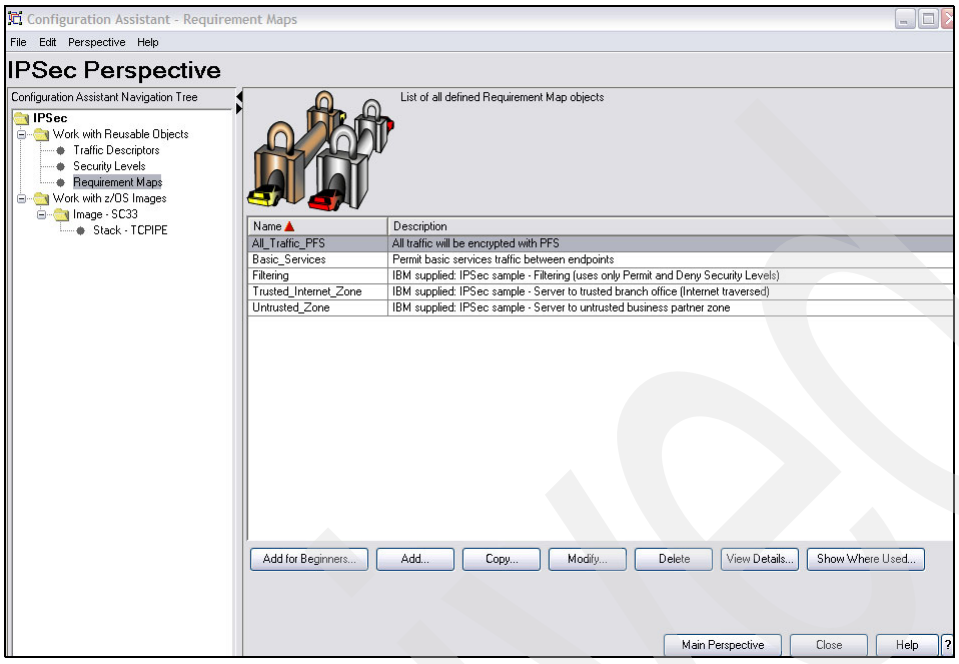


Figure C-4 Requirement Map panel: including All_Traffic_PFS and Basic_Services objects

Creating the connectivity rules

To create the connectivity rules, **Basic_Services_Rule** and **All_Traffic_PFS_Rule**:

1. Select **IPSec** → **Work with z/OS Images** → **Image - *image_name*** → **Stack - *stack_name*** from the navigation tree.
2. To add the Basic_Services_Rule, click **Add** from the TCP/IP Stack Settings panel, and then click **Next**.
3. In the Connectivity Rule: Network Topology panel, select the **This connectivity Rule will contain only Permit and Deny Security Levels** button and click **Next**.
4. In the Connectivity Rule: Data Endpoints panel, select **All IP V4 addresses** as the source data endpoint and **All IP V4 addresses** as the destination data endpoints. In addition, enter a name to the rule. (In our configuration, we entered `Basic_Services`, as shown in Figure C-4.) Click **Next**.
5. In Connectivity Rule: Select Requirement Map panel, select **Basic_Services** from the list and click **Next**. Then click **Finish**.
6. To add the All_Traffic_PFS rule, click **Add** from the TCP/IP Stack Settings panel and click **Next**.
7. In the Connectivity Rule: Network Topology panel, select the **This connectivity Rule will contain a Security Level using IPSec tunnels** button, and then select **Host to Host** button. Click **Next**.
8. In the Connectivity Rule: Data Endpoints panel, enter the z/OS image IP address as the local data endpoint (in our configuration, `192.168.1.40`) and the Windows XP IP address as the remote data endpoint (in our configuration, `10.1.100.222`). In addition, enter a name to the rule. In our configuration, we entered `All_Traffic_PFS`, as shown in Figure C-5 on page 718. Click **Next**.

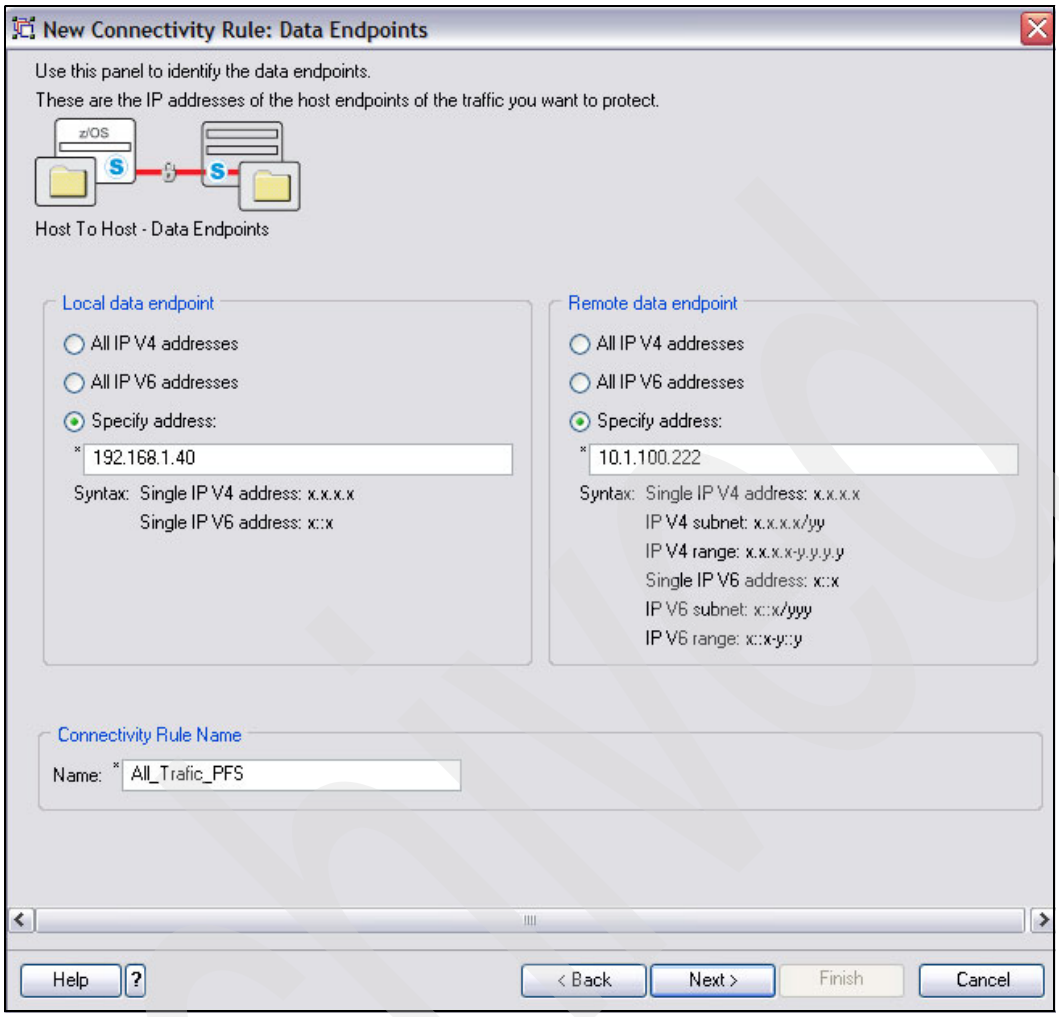


Figure C-5 Connectivity Rule: Data Endpoints panel - Defining All_Traffic_PFS

9. In the Connectivity Rule: Select Requirement Map panel, select **All_Traffic_PFS** from the list and click **Next**.

10. In the Connectivity Rule: Remote Security Endpoint Information panel, enter the remote IKE identity. We chose to use IP address as the remote identity type and therefore entered the Windows XP IP address. Select **Shared key** as the authentication method for remote IKE peers. Select **ASCII** and enter your shared key (in our configuration, `abcde`), as shown in Figure C-6 on page 719.

    Click **Next** and then click **Finish**.

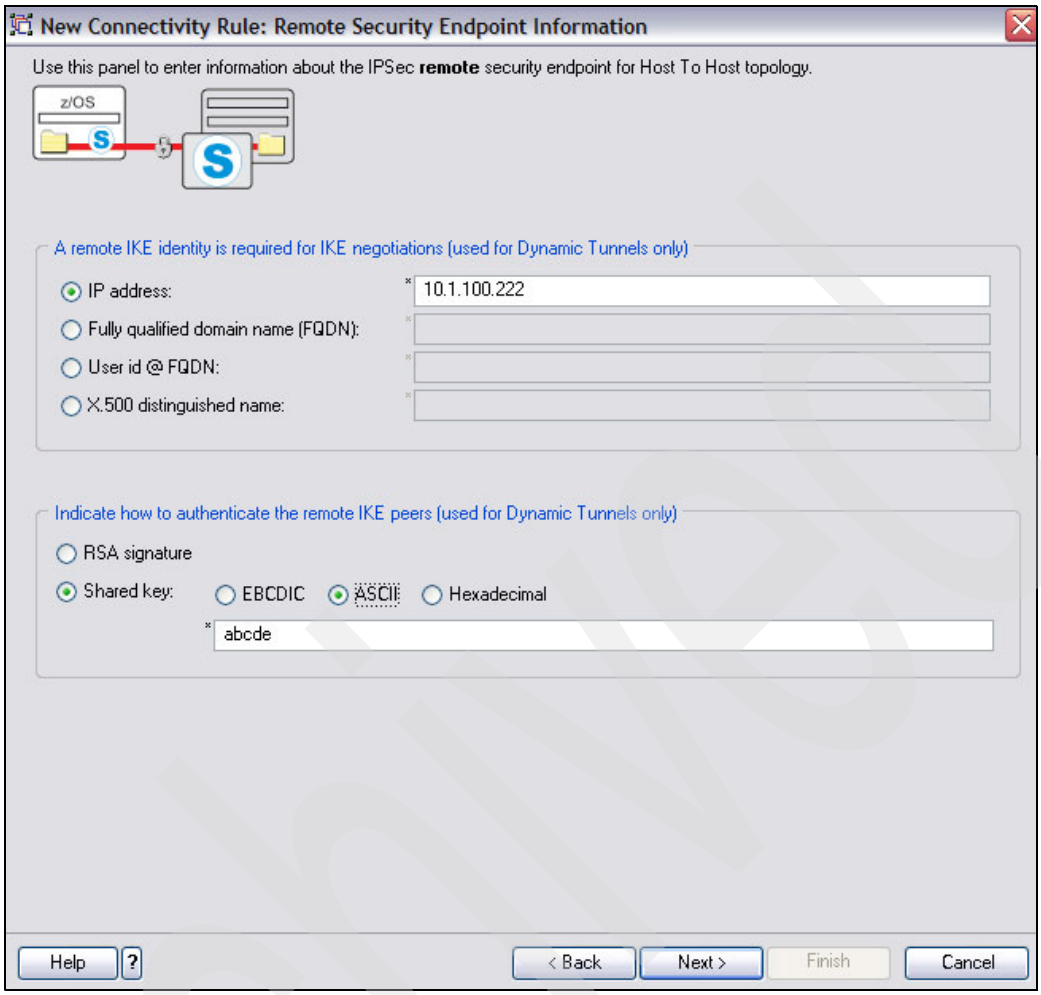


Figure C-6 Connectivity Rule: Remote Security Endpoint Information panel

The TCP/IP Stack Settings panel should now include the new rules, as shown in Figure C-7.

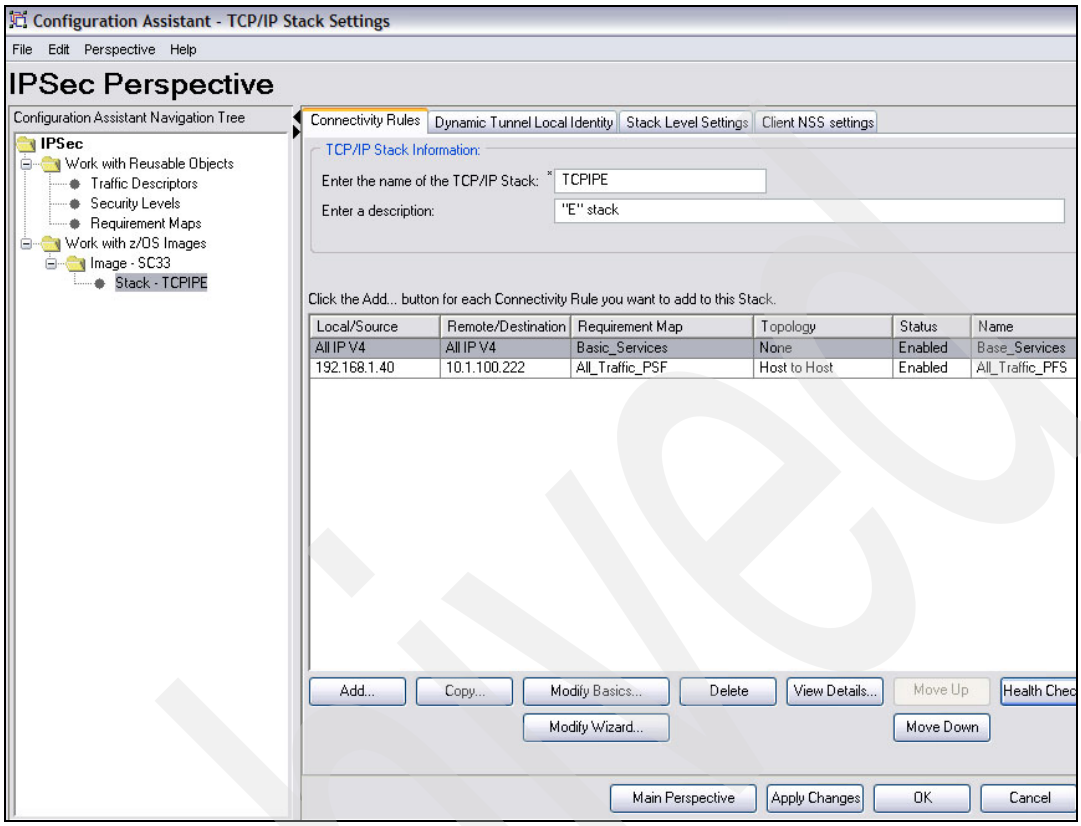


Figure C-7 TCP/IP Stack Settings panel with the All_Traffic_PFS rule

Note: There is an importance to the order of the rules in the connectivity rules list. Verify that the rules are in the right order and use the **Move UP** button if necessary to change the order of the rules.

Inherent in the rules discussed next is the default rule that always exists for the Policy Agent, which is to deny everything. If no matching rule is found, the packet will be denied by this default rule.

Verifying that the IKE daemon settings are correct

To verify that the IKE daemon settings are correct:

1. Select **IPSec** -> **Work with z/OS Images** -> **Image - *image_name*** from the navigation tree.
2. In the z/OS Image Settings panel, select the **IPSec: IKE daemon Settings** tab and verify that SAF key ring database value is IKED/IKED_keyring and that the list of supported certificate authorities includes SC33 RACF CA.

Updating the IPSec policy of the z/OS image

Install the new policy to the z/OS image, as explained in 9.5, “Configuring IPSec between two z/OS systems” on page 394.

Run the MODIFY PAGENT,REFRESH console command to update the Policy Agent and IKED with the new configuration.

Checkpoint

To summarize the situation at this point in the scenario, the Policy Agent (PAGENT) and IKE daemons should both be running. The GUI has been used to configure a set of policies that will direct the z/OS host to send all traffic through a dynamic IPSec VPN. A set of certificates has been created and IKE is pointing to the key ring that contains those certificates in the RACF database. The next step is to ensure that an equivalent setup has been handled on the Windows XP workstation.

Set up a Windows IPSec policy

In this section, we describe how to set up the Microsoft Management Console (MMC). The MMC will have two snap-ins added: Certificates and IP Security Management (see Figure C-8 on page 722). The Certificates Snap-in is used to import the certificates that are created by z/OS RACF. The IP Security Management Snap-in is used to configure the VPN connection between the Windows XP client and the z/OS server.

1. Click **Start** -> **Run** from the Windows XP task bar.
2. Enter `mmc` in the Open field. Click **OK** to start the Microsoft Management Console.
3. In the Console 1 window, click **Console** from the menu bar. From the pull-down menu, click **Add/Remove Snap-in**.
4. In the Add/Remove Snap-in window, click **Add**.
5. In the Add Standalone Snap-in window, select **Certificates** and click **Add**.
6. In the Certificates Snap-in window, select **Computer account** and click **Next**.
7. Select **Local computer** and click **Finish**.
8. In the Add Standalone Snap-in window, select **IP Security Policy Management** and click **Add**.
9. In the Select Computer window, select **Local computer** and click **Finish**.
10. In the Add Standalone Snap-in window, click **Close**.
11. In the Add/Remove Snap-in window, verify that two snap-ins have been added: Certificates (Local computer) and IP Security Policies on Local Machine. Click **OK**.

This process completes the required settings for the MMC.

Importing the z/OS certificates into Windows XP

In this section, we explain how to import two certificates that are created by z/OS RACF: the Trusted Root Certification Authority (CA) certificate, and the client certificate.

Before installing the client certificate on the Windows XP client, Windows XP needs to entrust the Trusted Root CA (in this case, z/OS RACF acts as a Trusted Root CA to provide certificates to the clients). After Windows XP entrusts the CA, the client certificate can be installed on the Windows XP client. This client certificate is used for identity authentication in the IKE phase 1 negotiation.

z/OS RACF creates an individual client certificate for each client, because the client certificate includes the client IP address information. This IP address information is used to verify the required authority on each client to connect to z/OS.

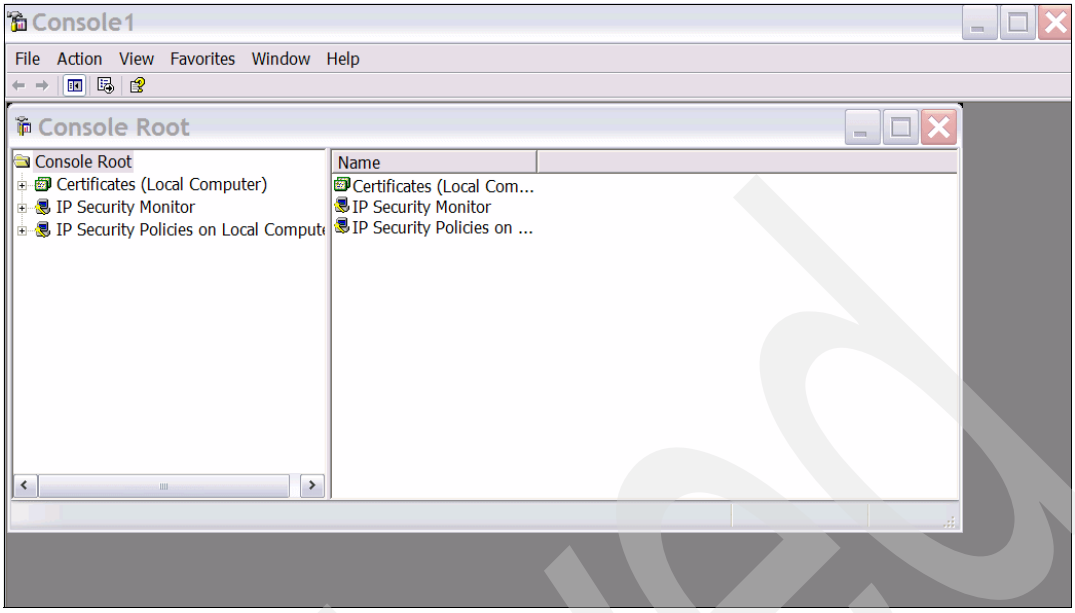


Figure C-8 Microsoft Management Console (MMC): Certificates

Transfer the certificate files from z/OS image using FTP, as shown in Example C-5.

Example: C-5 Transfer the certificates to the Windows XP client

```
C:\>ftp wtsc33.itso.ibm.com
Connected to wtsc33.itso.ibm.com.
220-FTPMVS1 IBM FTP CS V1R9 at wtsc33.ITSO.IBM.COM, 17:00:35 on 2007-09-26.
220 Connection will close if idle for more than 5 minutes.
User (wtsc33.itso.ibm.com:(none)): cs05
331 Send password please.
Password:
230 CS05 is logged on.  Working directory is "CS05.".
ftp> bin
200 Representation type is Image
ftp> get cert.xpcert c:\xpclient.p12
200 Port request OK.
125 Sending data set CS05.CERT.XPCERT
250 Transfer completed successfully.
ftp: 1722 bytes received in 0.00Seconds 1722000.00Kbytes/sec.
```

Note: You must change the representation type of the FTP to image before running the **get** commands.

To import the z/OS certificates into Windows XP, perform the following steps:

1. In the MMC window shown in Figure C-9 on page 723, click the plus sign (**+**) next to Certificates (Local Computer) to show the list of available tasks.
2. Right-click **Trusted Root Certification Authorities** and choose **All Tasks** from the pull-down menu. Choose **Import** from the next pull-down menu and click it, as shown in Figure C-9 on page 723.

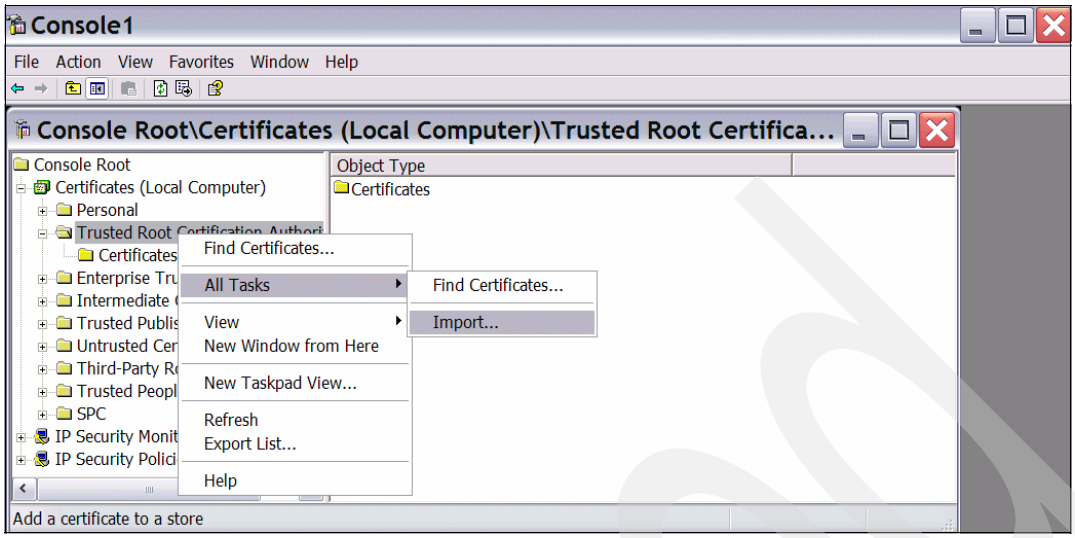


Figure C-9 Trusted Root Certification Authorities

3. In the Certificate Import Wizard window, click **Next**.
4. In the Certificate Import Wizard - File to Import window, click **Browse** and specify the Trusted Root CA file name (in this example, we choose `C:\racfca.p12` for the Trusted Root CA file, as shown in Figure C-10). Click **Next**.

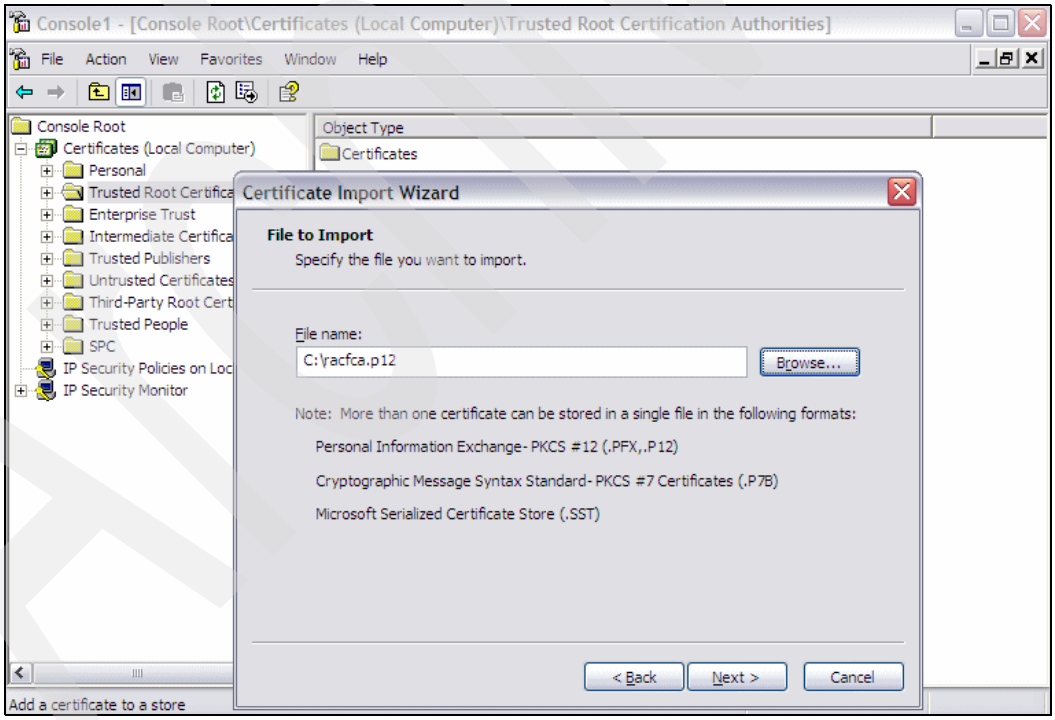


Figure C-10 Certificate Import Wizard

5. Work through the Certificate Import Wizard, indicating the following:
   a. Enter the password you gave to the CA certificate. (We entered `security` as the password.)
   b. Do *not* select `Mark this key as exportable.` (shown in Figure C-11 on page 724).

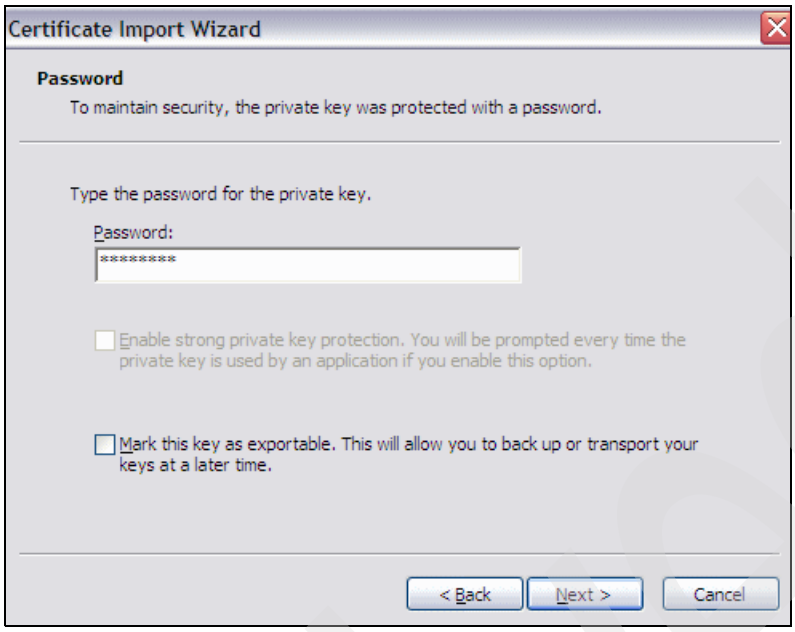


Figure C-11 Certificate Import Wizard: Entering the password of the certificate

c. Select **Place all certificates in the following store**.

d. Indicate that `Trusted Root Certification Authorities` is shown in the certificate store column.

e. Click **Finish** in the Completing the Certificate Import Wizard window.

You will receive a message indicating that the import was successful.

6. In the MMC window, click the plus (**+**) sign next to Trusted Root Certification Authorities, and then click **Certificates**. Scroll down and verify that your Trusted Root CA is installed in the list. In this example, SC33 RACF CA is installed as a Trusted Root CA, as shown in Figure C-12.

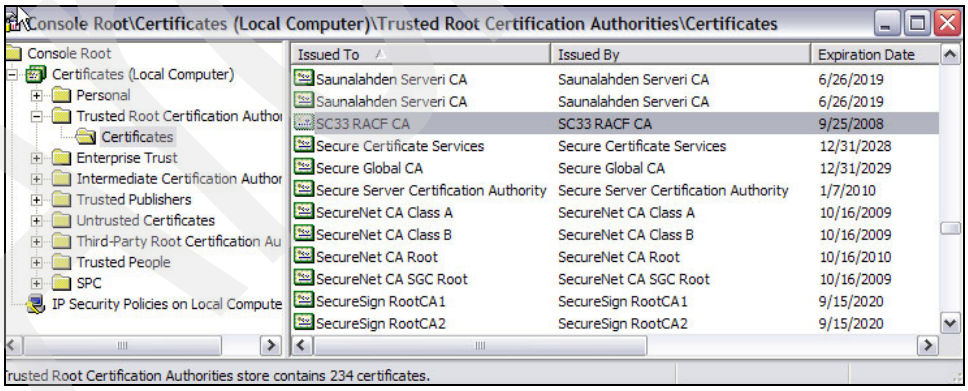


Figure C-12 The Certificates list with the new SC33 RACF CA certificate

7. Right-click **Personal** and choose **All Tasks** from the pull-down menu. Choose **Import** from the next pull-down menu and select it, as shown in Figure C-13.

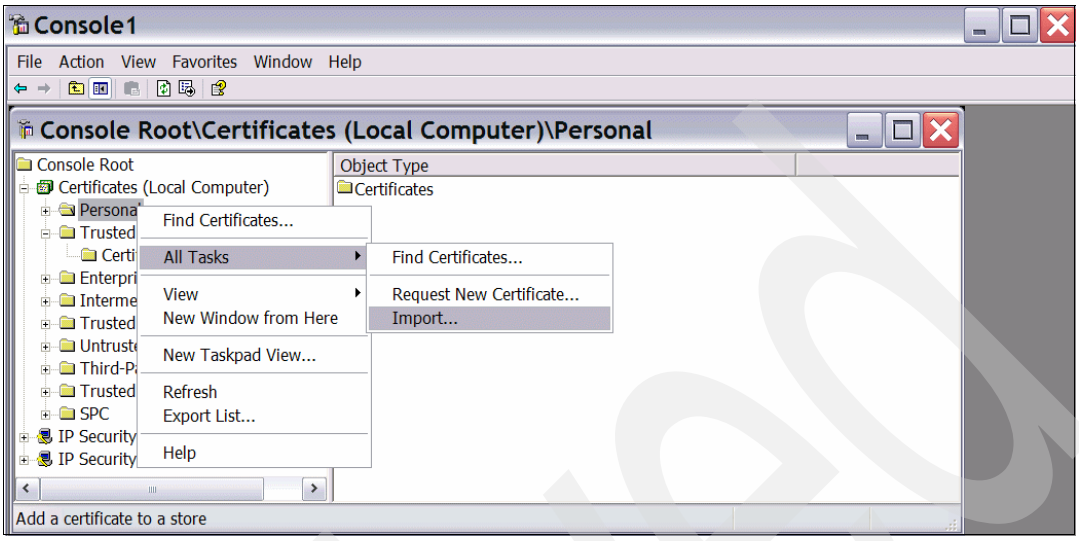


Figure C-13 Personal Certification Authorities

8. In the Certificate Import Wizard, click **Next**.
9. In the Certificate Import Wizard - File to Import window, click **Browse** and specify the client certificate file name (in this example, we choose `C:\xpclient.p12` for the client certificate file shown in Figure C-14). Click **Next**.

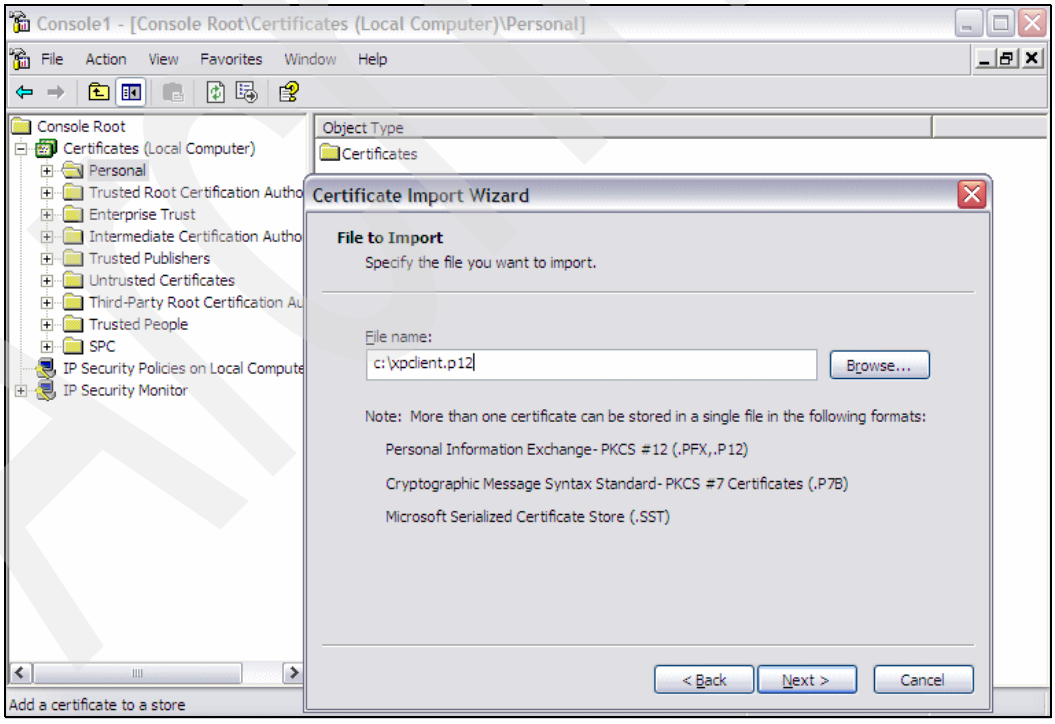


Figure C-14 Importing the XP client certificate

10. Continue to work through the Certificate Import Wizard, indicating the following:

    a. Enter the password you gave to the certificate. (We entered `security` as the password.)

    b. Do *not* select “Mark the private key as exportable.”, as shown in Figure C-15.

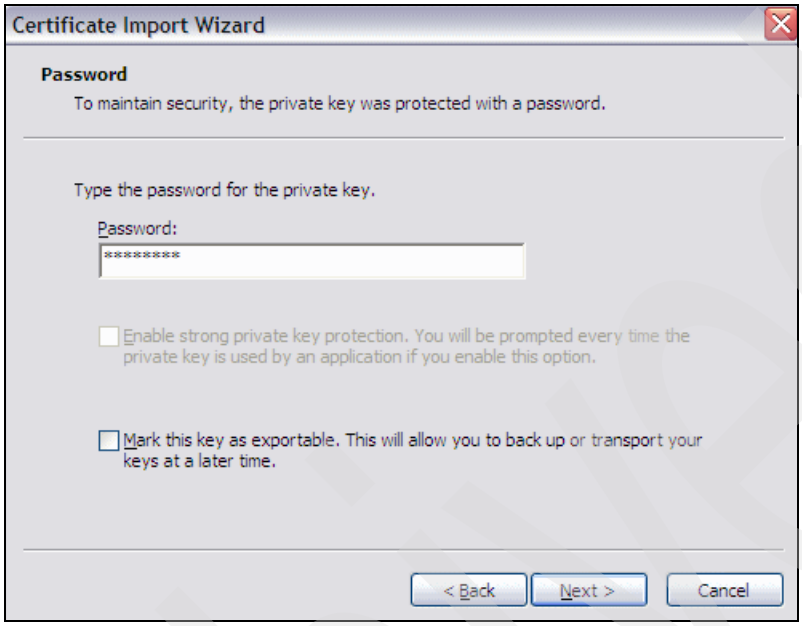


Figure C-15 Certificate Import Wizard: Entering the password of the certificate

    c. Make sure **Place all certificates in the following store** is selected and `Personal` is shown in the certificate store column.

    d. Click **Finish** in the Completing the Certificate Import Wizard window.

    You will receive a message indicating that the import was successful.

11. In the MMC window, click the plus (**+**) sign next to Personal, and then click **Certificates**. Scroll down and verify that your client certificate is installed in the list. In this example, WinXP Client is installed as a client certificate, as shown in Figure C-16.

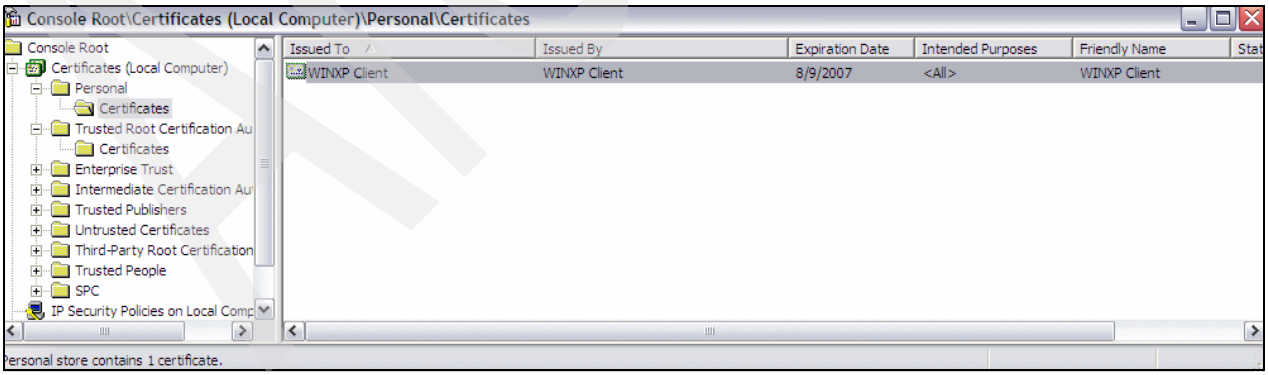


Figure C-16 Personal certificates list with WINXP Client certificate

Creating the IP security policy

In the following steps, you create the IP Security policy on your Windows XP workstation for the VPN connection between z/OS and the Windows XP client.

1. In the MMC - IP Security Policies on Local Computer window, right-click **IP Security Policies on Local Computer**. In the pull-down menu, click **Create IP Security Policy**, as shown in Figure C-17.

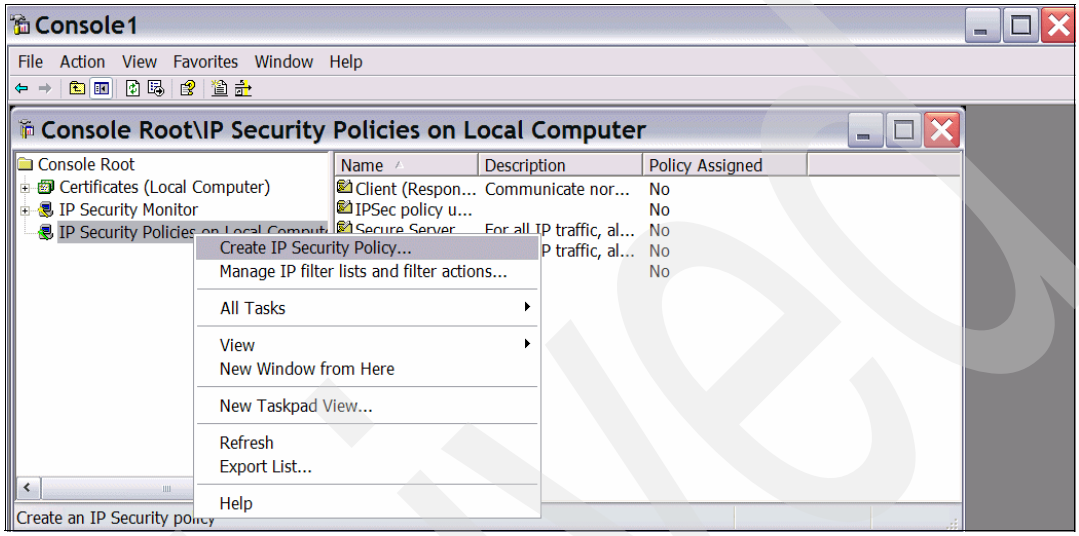


Figure C-17 MMC: IP Security Policies on Local Computer

2. In the IP Security Policy Wizard - Welcome to the IP Security Policy Wizard, click **Next**.
3. In the IP Security Policy Wizard - IP Security Policy Name window, type the name for the z/OS VPN connection. (In this example, we typed `zOSVPN` for the VPN connection name.) Type the description, if required. Click **Next**.
4. In the IP Security Policy Wizard - Requests for Secure Communication window, clear the Activate the default response rule check box. Click **Next**.
5. In the IP Security Policy Wizard - Completing the IP Security Policy Wizard, make sure that the **Edit properties** check box is selected. Click **Finish**.

6. In the IP Policy Properties window (in this example, the zOSVPN Properties window), select the **Use Add Wizard** check box, as shown in Figure C-18. Click **Add**.

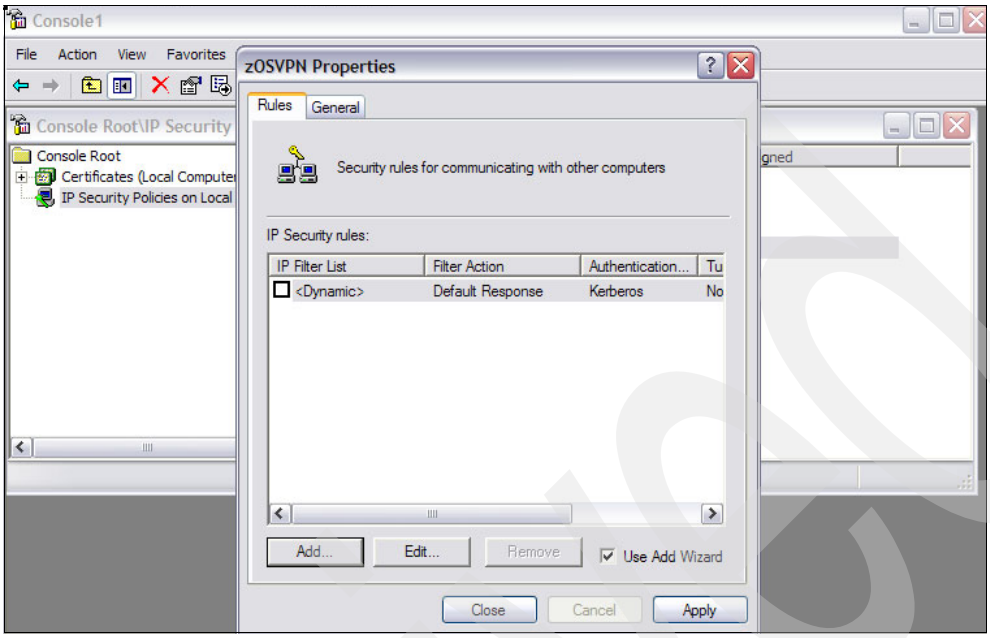


Figure C-18 IP policy properties window

7. In the IP Security Wizard - Welcome to the IP Security Policy Wizard window, click **Next**.
8. In the Security Rule Wizard - Tunnel Endpoint window, select **This rule does not specify a tunnel** and click **Next**. This selection means that the z/OS image is the endpoint of the VPN tunnel with Windows XP, and the VPN tunnel is defined as transport mode.
9. In the Security Rule Wizard - Network Type window, select **All network connections**. Click **Next**. In this example, z/OS image and Windows XP are connected with the Ethernet LAN.

> **Note:** If you want to limit the remote access connection, select **Remote Access**.

10. In the IP Security Policy Wizard - Authentication Method window, select **Use this string to protect the key exchange (preshared key)**, enter your shared key, as shown in Figure C-19 on page 729. (In this example, our shared key is `abcde`.) Click **Next**.

> **Note:** We recommend that you do *not* use a shared key authentication method in a production environment. In a production environment, you should configure RSA signature as the authentication method of the IKE peers. For more information about this topic, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

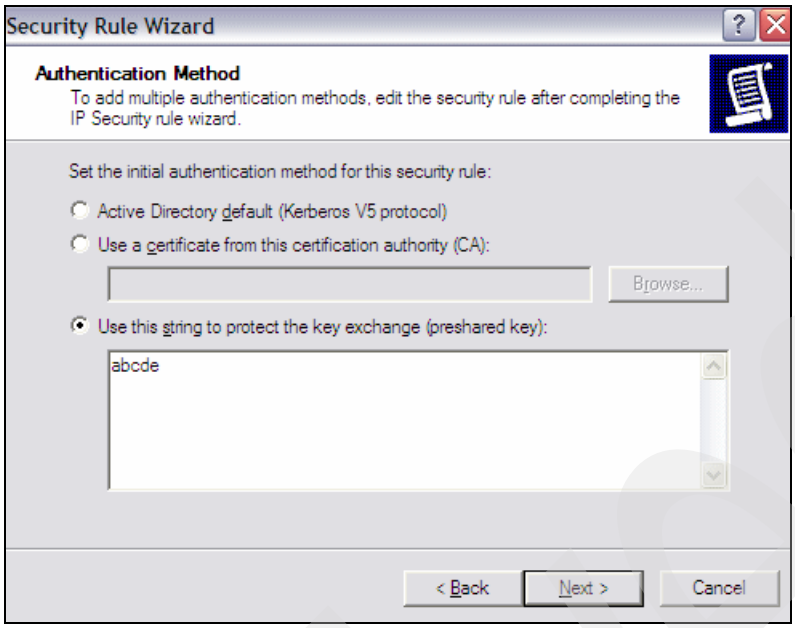


Figure C-19 Authentication Method panel: Using a shared key

11. In the Security Rule Wizard - IP Filter List window, click the circle for **All IP Traffic**, as shown in Figure C-20. Click **Edit**.

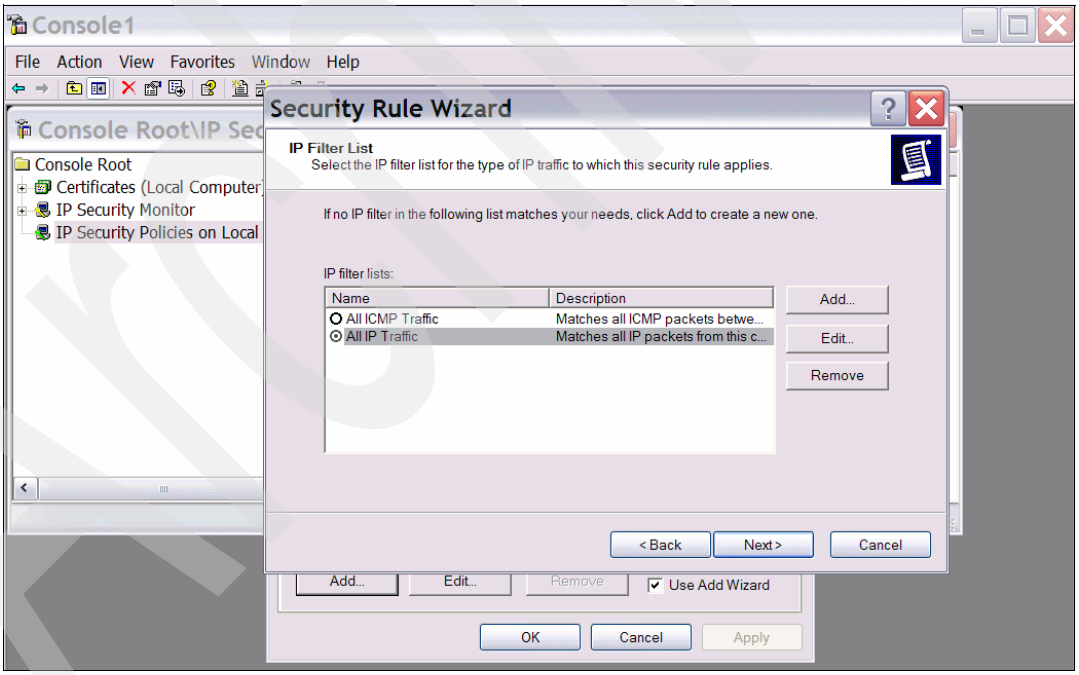


Figure C-20 Security Rule Wizard: IP filter list

Important: Notice that this IP filter works as a trigger event to establish the VPN tunnel. In this example, we choose `All IP Traffic` for the protocol and `192.168.1.40` for the Destination IP address. This means that if any IP datagram is about to issue from the Windows XP to 10.1.100.222, this IP filter detects the event and pulls a trigger to create a VPN tunnel between the Windows XP and 192.168.1.40.

12. In the IP filter List window, click **Edit**, as shown in Figure C-21.

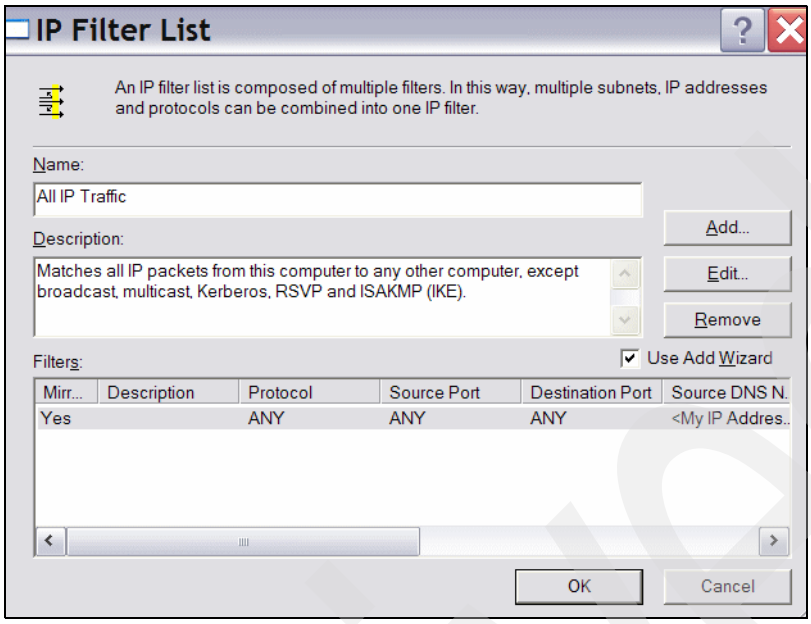


Figure C-21 IP filter list

13. In the Filter Properties window (shown in Figure C-22), select **A specific IP Address** in the Destination address column. Type the IP address of the z/OS image. (This IP address also means the VPN endpoint.) Select the **Mirrored. Also match packets with the exact opposite source and destination addresses** check box. (In this example, we typed `192.168.1.40` for the Destination IP address.) Click **OK**.

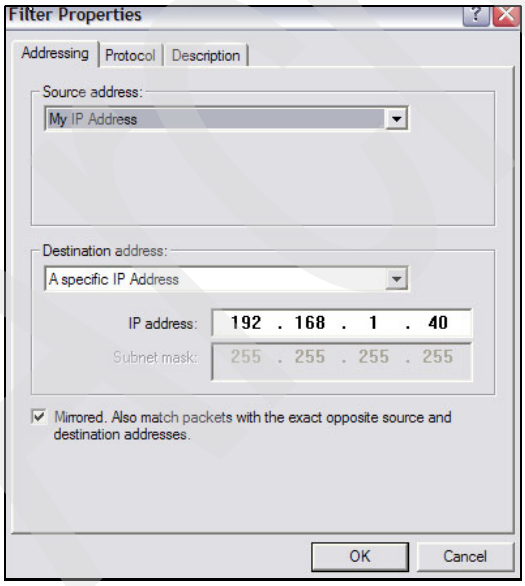


Figure C-22 Filter properties

14. In the IP Filter List window, click **OK**.
15. In the Security Rule Wizard - IP Filter list window, click **Next**.
16. In the Security Rule Wizard - Filter Action window, click the circle for **Require Security**, as shown in Figure C-23 on page 731. Click **Edit**.

Figure C-23 Security Rule Wizard: Filter Action

17. In the Require Security Properties window, choose **Negotiate security**; **Accept unsecured communication, but always respond using IPSec**; and **Session key Perfect Forward Security**.

    Use of Session key Perfect Forward Security is optional. In this example, we have to select it because the matching sample configuration in z/OS specifies to use the Session key Perfect Forward Security. Choose the topmost security method and click **Edit**, as shown in Figure C-24.

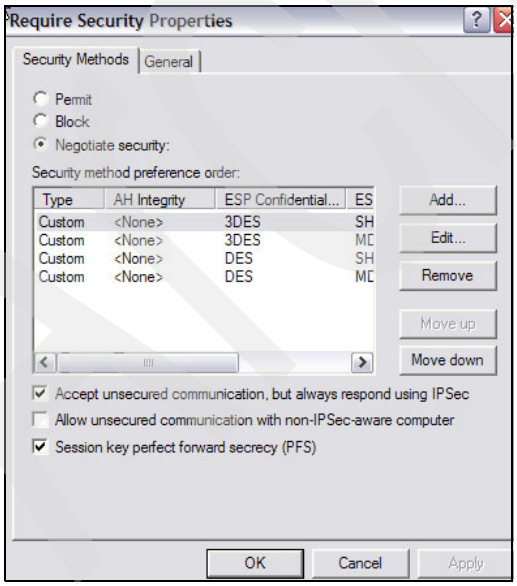


Figure C-24 Require Security Properties

18. In the Modify Security Method window, choose **Custom** (for expert users). Click **Settings**, as shown in Figure C-25 on page 732.

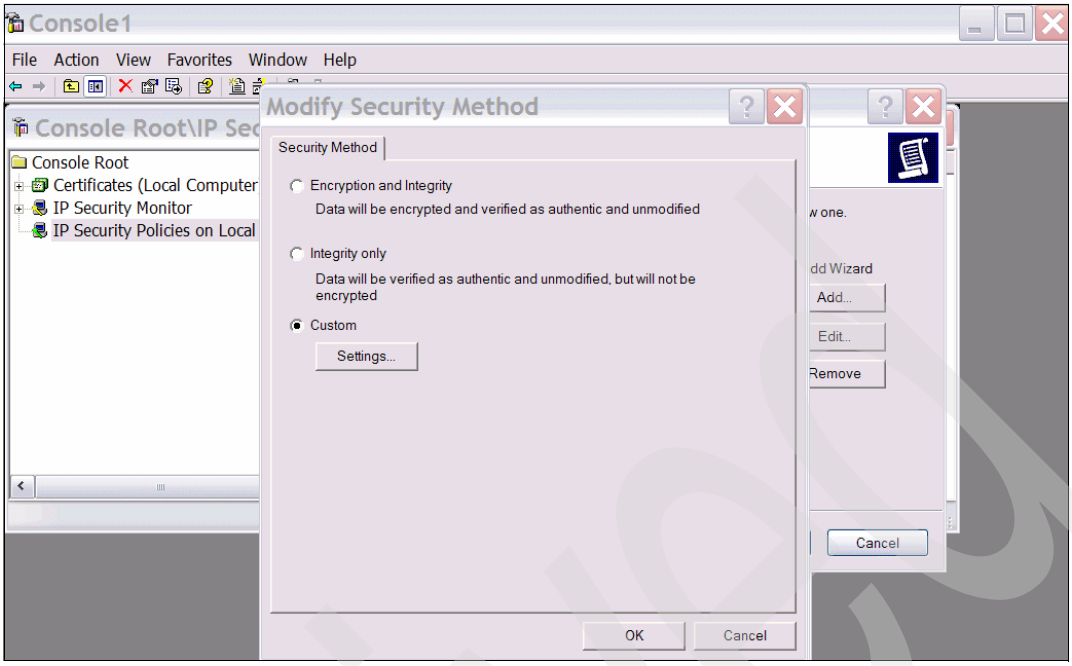


Figure C-25 Modify security method

19. In the Custom Security Method Settings window, make sure that the Data and address integrity without encryption (AH) check box is *not* selected.

    Select **Data integrity and encryption (ESP)**, and select **SHA1** for integrity algorithm. Select **3DES** for encryption algorithm. Clear the Generate a new key every Kbytes check box. Select the **Generate a new key every seconds** check box and type 7200 in the seconds column, as shown in Figure C-26. Click **OK**.

Figure C-26 Custom Security Method Settings

20. In the Modify Security Method window, click **OK**.
21. In the Require Security Properties window, click **OK**.
22. In the Security Rule Wizard - Filter Action window, click **Next**.
23. In the Security Rule Wizard - Completing the New Rule Wizard window, clear the Edit properties check box and click **Finish**.

24. In the zOSVPN Properties window, make sure that the **All IP Traffic** check box is checked, as shown in Figure C-27. Click **OK**.

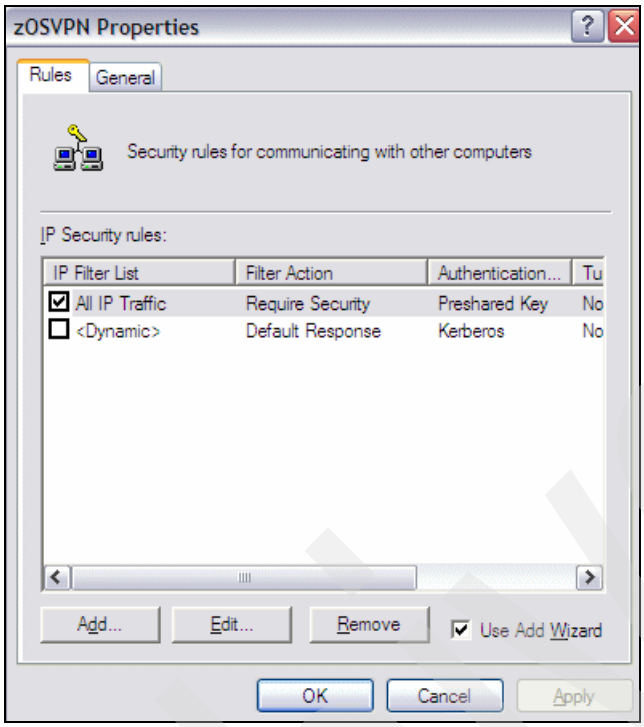


Figure C-27 zOSVPN properties

25. Verify that the IP Security Policies on Local Computer list now includes the zOSVPN policy.
26. Right-click the zOSVPN policy and select **Assign** from the menu.
27. Verify that the Policy Assigned status has changed to `Yes`, as shown in Figure C-28.

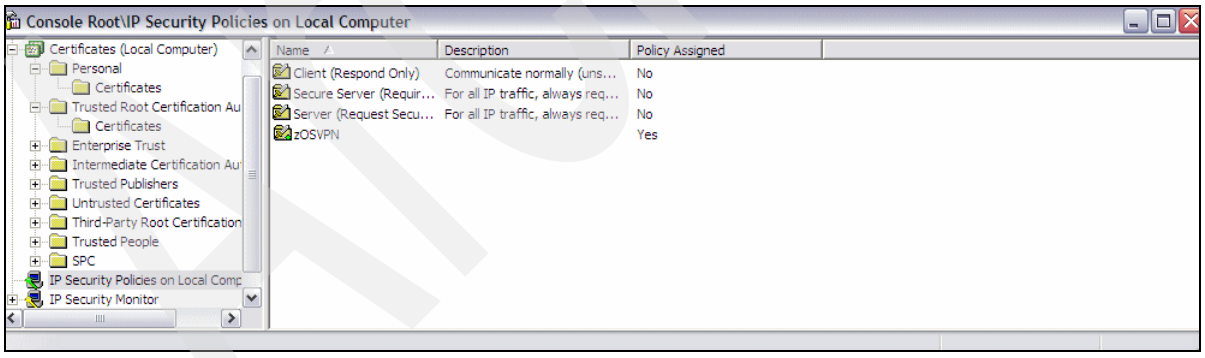


Figure C-28 The IP Security Policies on Local Computer panel: With zOSVPN policy assigned

Changing the Diffie-Hellman group (option)

We also tested multiple PFS groups supported by z/OS. For the PFS security level on z/OS IPSec definitions, we specified **none**, **Diffie-Hellman groups 1**, **2** and **5** as **Acceptable Perfect Forward Secrecy Levels** with IBM Configuration Assistant for z/OS Communication Server. Then we changed the setting of Diffie-Hellman group from **Group2,** which is the

default of Windows XP to **Group1** in **IP Security Policies on Local Computer** definitions in **Local Security Settings** window as follows:

1. In the zOSVPN Properties window, Select **General**.
2. In the Key Exchange Settings, click **Methods...**.
3. In Key Exchange Security Methods, choose the topmost security method and click **Edit**.
4. In the IKE Security Algorithms, choose **Low (1)** in Diffie-Hellman group, as shown in Figure C-29. Click **OK**.

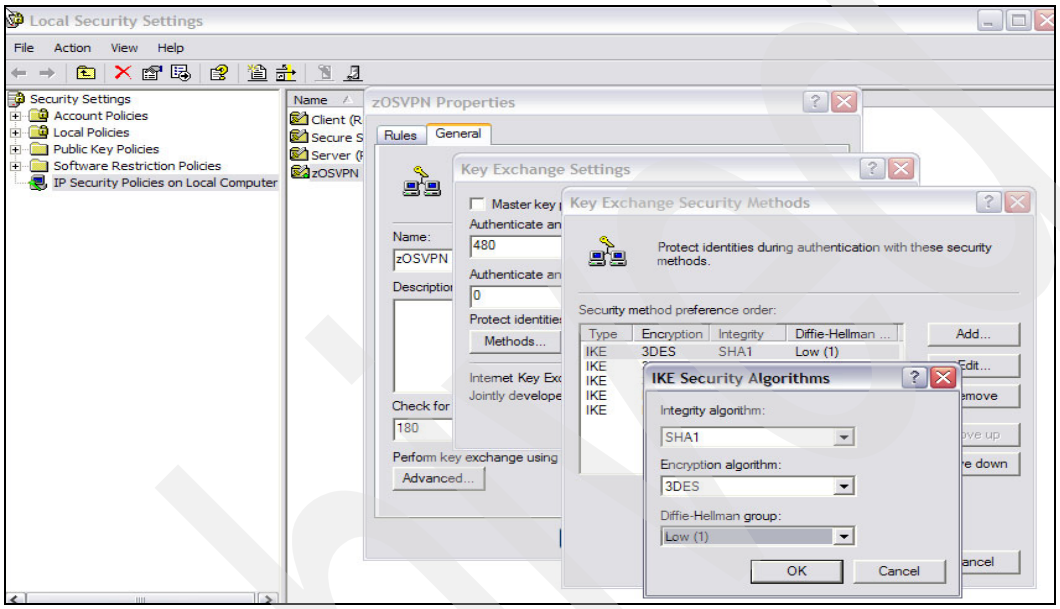


Figure C-29 IKE Security Algorithms: Diffie-Hellman group setting

Verify that things are working

Follow these steps to verify that things are working:

1. Check that the Policy Agent has read the new policy.
2. Check that you succeed in running the **ping** command from the Windows XP to the z/OS image.
3. Verify that phase 1 of the security association has started.
4. Verify that phase 2 of the security association has started.
5. If you have problems, check the syslogd messages.

Check that the Policy Agent has read the new policy

The first thing you need to check is that the Policy Agent has read in the current policies.

After you FTP the configuration file to the z/OS image, use the following command to refresh the PAGENT (where pagent is the started task name for the Policy Agent):

```
MODIFY PAGENT,REFRESH
```

If no changes have been made since the last time the polices were read, you should receive a message like this:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : NONE
```

If changes have been made, then you should receive a message like this:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

Check that the ping command is working

To verify that IP security is working, try the following **ping** command from the Windows XP to the z/OS image.

```
ping 192.168.1.40
```

You should receive output similar to that shown in Example C-6.

Example: C-6 The ping command

```
C:\Documents and Settings\RESIDENT>ping 192.168.1.40

Pinging 192.168.1.40 with 32 bytes of data:

Negotiating IP Security.
Reply from 192.168.1.40: bytes=32 time<1ms TTL=63
Reply from 192.168.1.40: bytes=32 time<1ms TTL=63
Reply from 192.168.1.40: bytes=32 time<1ms TTL=63

Ping statistics for 192.168.1.40:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average =
0ms
```

Verify that security association phase 1 has started

To verify that an IKED tunnel has been created, run the **ipsec -k** command.

```
ipsec -p tcpipA -k display
```

You should receive output similar to that shown in Example C-7.

Example: C-7 The ipsec -k command after the ping command from XP to z/OS

```
Primary:  IKE tunnel      Function: Display          Format:   Detail
Source:   IKED            Scope:    Current          TotAvail: n/a

TunnelID:                     K1
KeyExchangeRuleName:          0~5
KeyExchangeActionName:        0
LocalEndPoint:                192.168.1.40
LocalIDType:                  IPV4
LocalID:                      192.168.1.40
RemoteEndPoint:               10.1.100.222
RemoteIDType:                 IPV4
RemoteID:                     10.1.100.222
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      Hmac_Sha
EncryptionAlgorithm:          DES
DiffieHellmanGroup            1
AuthenticationMethod:         PresharedKey
InitiatorCookie:              0X9065110C1FE7D109
ResponderCookie:              0XB2D1C745A9A588C0
Lifesize:                     0K
CurrentByteCount:             1824b
Lifetime:                     480m
LifetimeRefresh:              2006/08/15 17:02:34
LifetimeExpires:              2006/08/15 18:26:17
```

```
Role:                          Responder
AssociatedDynamicTunnels:      1
NATTSupportLevel:              None
NATInFrntLclScEndPnt:          No
NATInFrntRmtScEndPnt:          No
zOSCanInitiateP1SA:            Yes
AllowNat:                      No
RmtNAPTDetected:               No
RmtUdpEncapPort:               n/a
*******************************************************************

1 entries selected
```

Verify that security association phase 2 has started

To verify that a dynamic tunnel has been created, run the **ipsec -y** command.

```
ipsec -p tcpipA -y display
```

You should receive output similar to that shown in Example C-8.

Example: C-8 The ipsec -y command after the ping command from XP to z/OS

```
Primary:  Dynamic tunnel  Function: Display          Format:   Detail
Source:   Stack           Scope:    Current          TotAvail: 1

TunnelID:                      Y4
ParentIKETunnelID:             K1
VpnActionName:                 DES
LocalDynVpnRule:               n/a
State:                         Active
HowToEncap:                    Transport
LocalEndPoint:                 192.168.1.40
RemoteEndPoint:                10.1.100.222
LocalAddressBase:              192.168.1.40
LocalAddressPrefix:            n/a
LocalAddressRange:             n/a
RemoteAddressBase:             10.1.100.222
RemoteAddressPrefix:           n/a
RemoteAddressRange:            n/a
HowToAuth:                     ESP
 AuthAlgorithm:                Hmac_Sha
 AuthInboundSpi:               1328063198
 AuthOutboundSpi:              3664910140
HowToEncrypt:                  DES
 EncryptInboundSpi:            1328063198
 EncryptOutboundSpi:           3664910140
Protocol:                      ALL(0)
LocalPort:                     0
RemotePort:                    0
OutboundPackets:               2
OutboundBytes:                 80
InboundPackets:                2
InboundBytes:                  80
Lifesize:                      OK
LifesizeRefresh:               OK
CurrentByteCount:              0b
LifetimeRefresh:               2006/08/15 12:26:42
LifetimeExpires:               2006/08/15 12:55:23
CurrentTime:                   2006/08/15 10:57:15
VPNLifeExpires:                2006/08/16 10:55:23
```

```
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtNAPTDetected:            No
  RmtIsGw:                    n/a
  RmtIsZOS:                   n/a
  zOSCanInitP2SA:             n/a
  RmtUdpEncapPort:            n/a
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
*********************************************************************

1 entries selected
```

Check the syslogd for messages

With TRMD running, syslogd is the repository for all Policy Agent and IKE daemon messages.

Enter the UNIX System Services environment by running the command:

```
tso omvs
```

From the UNIX System Services environment, browse the log file you defined for IKED in the IKED configuration file.

```
obrowse /tmp/iked-SC33.log
```

Look for messages that can help you to solve the problem. If the log is empty, verify that TRMD is running and that syslogd is running with the right configuration file.

Note: The IKED log file has an important role in problem determination. When implementing the IP security for the first time, make sure that:

- IKED is configured to write log messages.
- TRMD is running.
- syslogd is running with the updated configuration file.

Verify the security association on Windows

To verify that a dynamic tunnel has been created, in the Event Viewer panel, select **Security**.

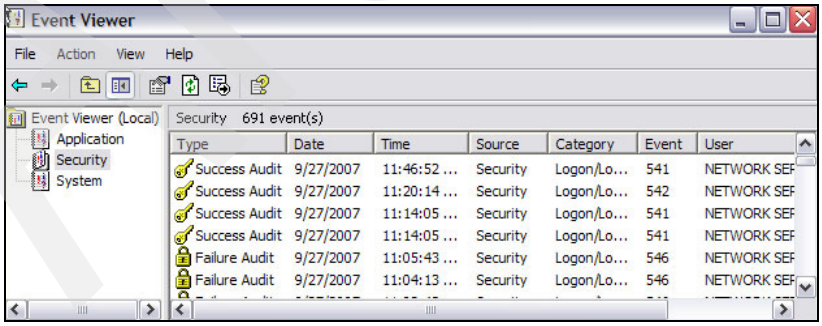


Figure C-30 Event Viewer -Security option

Double-click an event. A window pops up as shown in Figure C-31.

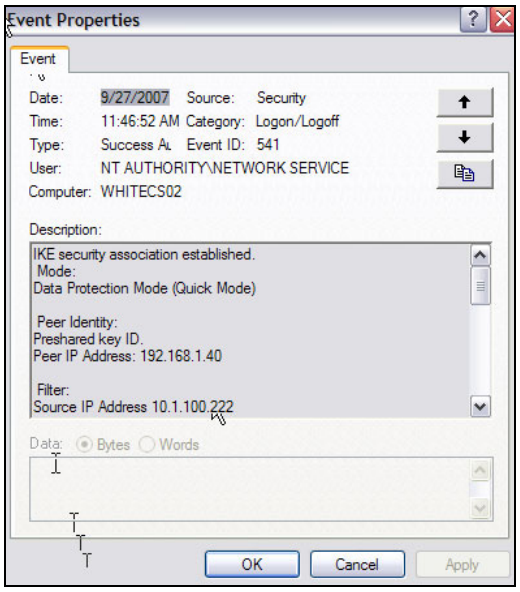


Figure C-31 Detailed Security event

Example C-9 shows the description of the Success Audit in the Event Properties window.

Example: C-9 Description of Success Audit

```
IKE security association established.
 Mode:
Data Protection Mode (Quick Mode)

 Peer Identity:
Preshared key ID.
Peer IP Address: 192.168.1.40

 Filter:
Source IP Address 10.1.100.222
Source IP Address Mask 255.255.255.255
Destination IP Address 192.168.1.40
Destination IP Address Mask 255.255.255.255
Protocol 0
Source Port 0
Destination Port 0
IKE Local Addr 10.1.100.222
IKE Peer Addr 192.168.1.40

 Parameters:
ESP Algorithm Triple DES CBC
HMAC Algorithm SHA
AH Algorithm None
Encapsulation Transport Mode
InboundSpi 3003844234 (0xb30b068a)
OutBoundSpi 2223428829 (0x8486d4dd)
Lifetime (sec) 7200
Lifetime (kb) 100000
```

Example C-10 shows the sample description in the Failure Audit case.

Example: C-10 Description of Failure Audit

```
IKE security association establishment failed because peer sent invalid proposal.
 Mode:
Key Exchange Mode (Main Mode)

 Filter:
Source IP Address 10.1.100.222
Source IP Address Mask 255.255.255.255
Destination IP Address 192.168.1.40
Destination IP Address Mask 255.255.255.255
Protocol 0
Source Port 0
Destination Port 0
IKE Local Addr 10.1.100.222
IKE Peer Addr 192.168.1.40

 Attribute:
Phase I Diffie-Hellman Group
 Expected value:
2
 Received value:
1
```

D

zIIP Assisted IPSec

The IBM System z Integrated Information Processor (zIIP) is a specialty engine designed to free up general purpose CPs and lower the software costs for selected workloads. The zIIP Assisted IPSec function allows Communications Server to interact with z/OS Workload Manager to have its work directed to zIIPs.

This appendix discusses the following topics.

Section	Topic
“Configuring zIIP Assisted IPSEC” on page 742	Configuration options
“Example of zIIP Assisted IPSec implementation” on page 743	

Background

The IP Security function of z/OS Communications Server uses the System z crypto hardware, called Crypto Express2 (CE2) or CP Assist for Cryptographic Function (CPACF), which is standard on every PU. CE2 and CPACF are used for data encryption and decryption, as well as for authentication.

Both CPACF (in conjunction with Integrated Cryptographic Service Facility (ICSF)) and CE2 create an increase in CPU utilization when IPSec is added, especially CPACF because it runs on the general purpose CPs. Even using CE2, some bulk workloads (such as FTP) can utilize a lot of CPU. This is because the cost of CPU is relative to the amount of data being moved.

In some cases, when running an LPAR at a high utilization level, the usage of IP Security (IPSec) could be a problem. In most cases the TCP/IP address space has a high priority and depending on the workload priority, this problem can be magnified. To minimize the problem the IPSec workload could run on a different TCP/IP address space with a lower priority, but the CPU consumption problem will remain.

As mentioned, the IBM System z Integrated Information Processor (zIIP) specialty engine frees up general purpose CPs and lowers the software costs for selected workloads. And the zIIP Assisted IPSec function allows Communications Server to interact with z/OS Workload Manager to have its work directed to zIIPs.

If you are running IPSec, you may be able to achieve significant reduction in general purpose CP consumption by using the zIIP-Assisted IPSec function.

In this appendix, IP Security (IPSec) is used in two different ways with different meanings:

IPSec Virtual Private Network (VPN) IP Security (IPSec), a peer-to-peer IP tunnel

IPSEC IP Security (IPSEC) feature in z/OS Communications Server, which provides TCP/IP filtering (firewall) and VPN IPSec support

For more information about the zIIP, go to the following Web page and perform a search on zIIP:

`http://www.ibm.com/support/techdocs`

Configuring zIIP Assisted IPSEC

An option in the GLOBALCONFIG statement enables the SRB-mode IPSec Authentication header (AH) and Encapsulating Security Payload (ESP) protocols to be processed on the zIIP. Figure D-1 shows the GOBALCONFIG statement to configure the zIIP Assisted IPSec function.

```
GLOBALCONFIG
  ZIIP IPSECURITY
```

Figure D-1 zIIP IPSec configuration

The other option is **`ZIIP NOIPSECURITY`**, which leaves the IPSec processing running on general CPs. This is the default.

Configuring GLOBALCONFIG ZIIP IPSECURITY causes inbound ESP and AH Protocol traffic to be processed in enclave SRBs, and targeted to available zIIPs. Outbound ESP and AH protocol traffic may also be processed on available zIIPs when:

- The application invoking the *send*() function is already running on a zIIP.
- The data to be transmitted is in response to normal TCP flow control (for example, data transmitted in response to a received TCP acknowledgement or window update).

If the machine does not have the zIIP installed, there is an option to project the CPU in the IEAOPTxx parmlib member called PROJECTCPU. Figure D-2 shows how to configure this option.

```
PROJECTCPU=YES
```

Figure D-2 IEAOPTxx PROJECTCPU configuration example

Example of zIIP Assisted IPSec implementation

In our test environment we implemented the zIIP Assisted IPSec function by defining a VPN IPSec tunnel between two z/OS systems, SC32 and SC33. Figure D-3 shows our environment.

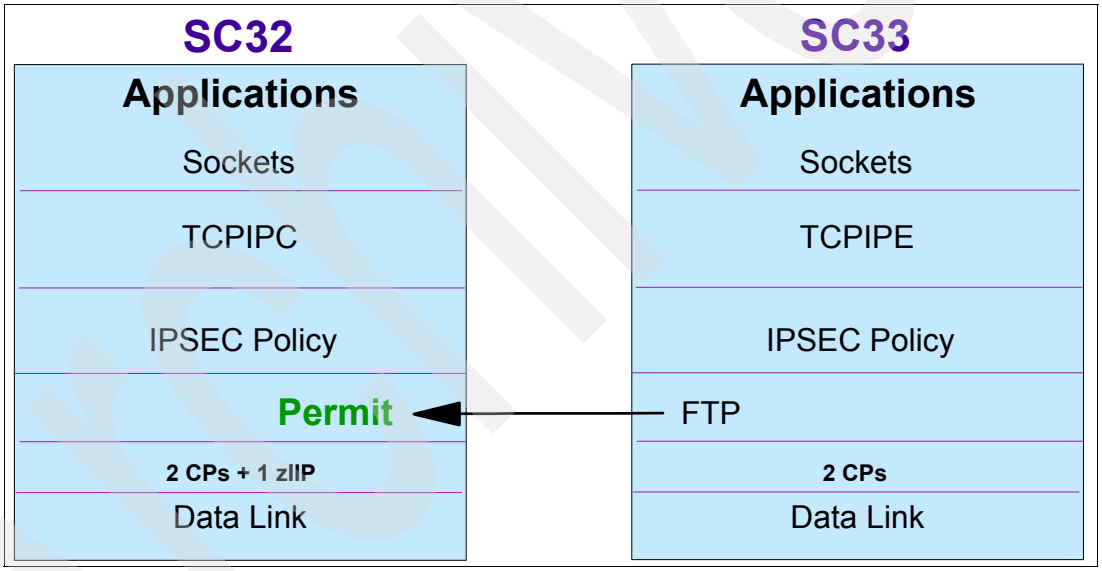


Figure D-3 Network configuration

To verify that you have a zIIP processor available to your LPAR, issue the following command:

```
D M=CPU
```

Example D-1 shows the result of this command on SC32. We can see that there are two CPs (indicated by a plus (+) sign) and one zIIP (indicated by +I).

Example: D-1 Display CPU from SC32

```
IEE174I 11.38.56 DISPLAY M 521
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                    25991E2094
01  +                    25991E2094
```

```
02  +I                    25991E2094
03  -
04  -

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.000000000002991E
CPC ID = 00
CPC NAME = SCZP101
LP NAME = A25        LP ID = 25
CSS ID  = 2
MIF ID  = 5

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

I        INTEGRATED INFORMATION PROCESSOR (zIIP)
CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME  LOGICAL PARTITION NAME
LP ID    LOGICAL PARTITION IDENTIFIER
CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
MIF ID   MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

Example D-2 shows the result of this command on SC33. Here we only have two CPs defined.

Example: D-2 Display CPU from SC33

```
IEE174I 11.43.46 DISPLAY M 148
PROCESSOR STATUS
ID  CPU                SERIAL
00  +                   29991E2094
01  +                   29991E2094

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.000000000002991E
CPC ID = 00
CPC NAME = SCZP101
LP NAME = A29        LP ID = 29
CSS ID  = 2
MIF ID  = 9

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

CPC ND  CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID  CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME  LOGICAL PARTITION NAME
LP ID    LOGICAL PARTITION IDENTIFIER
CSS ID   CHANNEL SUBSYSTEM IDENTIFIER
MIF ID   MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

To define a VPN in z/OS it is necessary to configure the following components:

- IPSECURITY in TCP/IP profile.
- Policy Agent (PAGENT) address space to handle all the configurations and install them in the TCP/IP stack.
- Traffic Regulation Management Daemon (TRMD) address space to log all the IPSEC messages.
- Internet Key Exchange daemon (IKED) address space to perform the key management.
- SYSLOGD address space to write the messages on a log file.

IPSECURITY is configured by using the IPCONFIG statement shown in Figure D-4.

```
IPCONFIG
  IPSECURITY
```

Figure D-4 IPCONFIG configuration

When IPSECURITY is configured, the TCP/IP stack automatically enables an implicit *deny all* firewall rule, blocking all the traffic to the stack. In our implementation we defined a rule to allow all the traffic to flow before defining the other rules using the pagent configuration. VPNs are only supported by using the pagent IPSEC configuration rules. The IPSEC definition in the TCP/IP profile is called the *default policy*.

Important: Use caution when defining IPSECURITY in TCP/IP. The TCP/IP stack has to be restarted or activated to use this function. If the TCP/IP stack is activated with no rules defined either by the IPSEC statement or by the PAGENT daemon, then all traffic to and from the stack will be blocked.

Our IPSEC statement to define the IPSEC default policy is shown in Figure D-5. This configuration will install the default policy allowing all traffic to flow in and out the stack.

```
IPSEC
   IPSECRULE * * NOLOG PROTOCOL *
ENDIPSEC
```

Figure D-5 IPSEC configuration example

The following definition shows the PAGENT configuration for both systems, SC32 and SC33. Figure D-6 and Figure D-7 show the main pagent configuration file. This file points to configuration files to specific TCP/IP stacks.

```
LogLevel 127
TcpImage TCPIPE /etc/pagent33_TCPIPE.conf  FLUSH PURGE 600
```

Figure D-6 SC33 pagent configuration file referenced by the pagent daemon

Figure D-7 shows the main SC32 pagent configuration file referenced by the pagent daemon.

```
LogLevel 127
TcpImage TCPIPC /etc/pagent32_TCPIPC.conf  FLUSH PURGE 600
```

Figure D-7 SC32 pagent configuration file referenced by the pagent daemon

Figure D-8 and Figure D-9 show the configuration files for the TCP/IP stack in both systems. They point to another file which contains the IPSEC policy definitions.

```
IPSecConfig  /etc/pagent33_TCPIPE_IPSec.conf
```

Figure D-8 SC33 TCP/IP stack policy

Figure D-9 shows the SC32 TCP/IP stack policy.

```
IPSecConfig  /etc/pagent32_TCPIPC_IPSec.conf
```

Figure D-9 SC32 TCP/IP stack policy

We used the IPSEC configuration for the SC32 and SC33 created in 9.5, “Configuring IPSec between two z/OS systems” on page 394. The IBM Configuration Assistant was used to build this environment.

There are only two IPSEC rules defined on each of the preceding examples:

- One rule that allows all inbound and outbound traffic in the TCP/IP stack. Usually this rule does not exist, but in our example we define it to facilitate the tests.
- Another rule that creates a VPN on demand between SC33 and SC32 when any type of traffic occurs. “On demand” means that the VPN will be activated when any traffic matching a specific rule is encountered.

> **Note:** An implicit rule denying all traffic is always created by having either the default policy rules or the pagent policy.

To start the VPN between the systems, we simply issue a **`ping`** command to the other side, or generate data traffic between the two systems.

For this implementation, we created one batch FTP job to transfer data. Each one of the systems (SC32 and SC33) will have a client and a server version talking to each other and sending packets. This batch only sends (client) and receives (server) data.

Example D-3 shows the FTP batch job we used.

Example: D-3 FTP job

```
//FTPBAT1 JOB (999,POK),'Batch FTP',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
/*JOBPARM L=999,SYSAFF=SC33
//FTP      EXEC PGM=FTP PARM='/-d (EXIT'
//SYSTCPD  DD  DISP=SHR,DSN=TCPIPE.TCPPARMS(DATAE33)
//SYSFTPD  DD  DISP=SHR,DSN=TCPIPE.TCPPARMS(FTPSE33)
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*,LRECL=160,BLKSIZE=16000,RECFM=FB
//INPUT    DD  *
 10.1.1.30
 userid
 password
 bin
 put 'input dataset-name(member)' 'output dataset-name(member)'
 close
 quit
/*
```

After starting the tasks, we can monitor the zIIP Assisted IPSec environment and configuration.

The **netstat stats** command (see Figure D-10) shows how many packets are being handled in the zIIP processor.

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPC
IP Statistics (IPv4)
  Packets Received                    = 64580
  Received Header Errors              = 0
  Received Address Errors             = 0
  Datagrams Forwarded                 = 0
  Unknown Protocols Received          = 0
  Received Packets Discarded          = 0
  Received Packets Delivered          = 65975
  Output Requests                     = 29797
  Output Discards No Route            = 0
  Output Discards (other)             = 0
  Reassembly Timeouts                 = 0
  Reassembly Required                 = 0
  Reassembly Successful               = 0
  Reassembly Failures                 = 0
  Datagrams Successfully Fragmented   = 0
  Datagrams Failing Fragmentation     = 0
  Fragments Created                   = 0
  Inbound Packets handled by zIIP     = 35553
  Outbound Packets handled by zIIP    = 20096
```

Figure D-10 A netstat -S command example

The **netstat config** command (see Figure D-11) can be used to verify whether the zIIP support is enabled.

```
MVS TCP/IP NETSTAT CS V1R9       TCPIP Name: TCPIPC
Global Configuration Information:
TcpIpStats: No   ECSALimit: 0000000K  PoolLimit: 0000000K
MlsChkTerm: No   XCFGRPID:      IQDVLANID:  0
SegOffLoad: No   SysplexWLMPoll: 060
ExplicitBindPortRange:  00000-00000
Sysplex Monitor:
  TimerSecs: 0060  Recovery: No   DelayJoin: No   AutoRejoin: No
  MonIntf:   No    DynRoute: No
zIIP:
  IPSecurity:Yes
```

Figure D-11 A netstat -f command example

The Resource Measurement Facility (RMF™) monitor can be used to check how much CPU is being used by the zIIP. In our test we used the Monitor III to show some details of how the zIIP is being utilized by LPARs and address spaces.

The CPC Capacity panel in Monitor III shows how the CPU is being used by partitions (see Example D-4 on page 748).

Example: D-4 RMF CPC capacity example

```
RMF V1R9   CPC Capacity                  Line 1 of 61
 Command ===>                                          Scroll ===> CSR

 Samples: 90      System: SC32  Date: 09/28/07  Time: 09.35.00  Range: 100   Sec

 Partition:   A25        2094 Model 710
 CPC Capacity:     640   Weight % of Max: ****   4h Avg:    2   Group:   N/A
 Image Capacity:   256   WLM Capping %:    0.0   4h Max:    3   Limit:   N/A

 Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
              Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total

 *CP                       72.0                       3.6    25.6   29.1
 A25            0     3  NO   2.0        2.3     2.4    0.0     0.5    0.5
 A29            0     3  NO   2.0        2.3     2.4    0.0     0.5    0.5

 *IIP                      13.0                       2.5    25.1   27.6
 A25                      NO   1.0       50.0    51.1    0.5    25.0   25.6
```

In Figure D-12, note how the logical partitions are using the zIIP on A29 (SC33) and A25 (SC32).

The RMF examples (Figure D-12 and Figure D-13 on page 749) show how the zIIP is being used by the address spaces on SC33 and SC32 in the RMF Processor Usage panel.

```
 RMF V1R9   Processor Usage               Line 1 of 8
Command ===>                                          Scroll ===> CSR

Samples: 94      System: SC33  Date: 09/28/07  Time: 11.16.40  Range: 100   Sec

            Service    --- Time on CP % ---   ----- EAppl % -----
Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

TCPIPE   SO SYSSTC      41.8    0.0    0.0     41.8
FTPBAT2  BO SYSOTHER     2.0    0.0    0.0      2.0
RMFGAT   SO SYSSTC       0.7    0.0    0.0      0.7
XCFAS    S  SYSTEM       0.5    0.0    0.0      0.5
NET      SO SYSSTC       0.5    0.0    0.0      0.5
WLM      S  SYSTEM       0.2    0.0    0.0      0.2
GRS      S  SYSTEM       0.1    0.0    0.0      0.1
JES2     S  SYSSTC       0.1    0.0    0.0      0.1
```

Figure D-12 SC33 RMF Processor Usage example

As we can see in Figure D-12, the TCPIPE address space is using most of the CPU. It is responsible for the ESP and AH protocol processing, encrypting and decrypting all the data.

Figure D-13 on page 749 shows the SC32 system.

```
 RMF V1R9   Processor Usage                    Line 1 of 9
Command ===>                                              Scroll ===> CSR

Samples: 90      System: SC32  Date: 09/28/07  Time: 11.16.40  Range: 100   Sec

            Service    --- Time on CP % ---   ----- EAppl % -----
Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

TCPIPC   SO SYSSTC       7.1    0.0    0.0      7.1           46.8
CSO3     O  SYSOTHER     4.1    0.0    0.0      4.1            0.0
SYSLOGDC O  SYSOTHER     1.8    0.0    0.0      1.8            0.0
RMFGAT   SO SYSSTC       0.7    0.0    0.0      0.7            0.0
XCFAS    S  SYSTEM       0.5    0.0    0.0      0.5            0.0
NET      SO SYSSTC       0.4    0.0    0.0      0.4            0.0
WLM      S  SYSTEM       0.3    0.0    0.0      0.3            0.0
*MASTER* S  SYSTEM       0.1    0.0    0.0      0.1            0.0
JES2     S  SYSSTC       0.1    0.0    0.0      0.1            0.0
```

Figure D-13 SC32 RMF Processor Usage example

In Figure D-13 we can see that most of the CPU utilization was displaced to the zIIPs.

It is possible to enable and disable the zIIP usage by using the OBEYFILE command. We simply change the option in GLOBALCONFIG to ZIIP NOIPSECURITY and update the profile to disable the zIIP utilization. To enable, simply configure back to ZIIP IPSECURITY.

Now we can turn the zIIP off on SC32 and see how the system will behave (see Figure D-14).

```
 RMF V1R9   Processor Usage                   Line 1 of 12
 Command ===>                                             Scroll ===> CSR

 Samples: 89      System: SC32  Date: 09/28/07  Time: 11.30.00  Range: 100
Sec

             Service    --- Time on CP % ---   ----- EAppl % -----
 Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

 TCPIPC   SO SYSSTC      18.6    0.0    0.0     18.6            0.0
 CSO3     O  SYSOTHER     0.8    0.0    0.0      0.8            0.0
 RMFGAT   SO SYSSTC       0.7    0.0    0.0      0.7            0.0
 XCFAS    S  SYSTEM       0.5    0.0    0.0      0.5            0.0
 WLM      S  SYSTEM       0.2    0.0    0.0      0.2            0.0
 NET      SO SYSSTC       0.2    0.0    0.0      0.2            0.0
 CSO3     TO SYSOTHER     0.2    0.0    0.0      0.2            0.0
 *MASTER* S  SYSTEM       0.1    0.0    0.0      0.1            0.0
 GRS      S  SYSTEM       0.1    0.0    0.0      0.1            0.0
 JES2     S  SYSSTC       0.1    0.0    0.0      0.1            0.0
 RMF      S  SYSSTC       0.1    0.0    0.0      0.1            0.0
 OMPC     SO SYSSTC       0.1    0.0    0.0      0.1            0.0
```

Figure D-14 SC32 RMF Processor Usage example 2

The jobname TCPIPC represents the TCP/IP address space. Without the zIIP, the CPU utilization was displaced back to the general CPs. For the CPC activity panel, see Figure D-15.

```
 RMF V1R9   CPC Capacity                  Line 17 of 61
Command ===>                                          Scroll ===> CSR

Samples: 89      System: SC32  Date: 09/28/07  Time: 11.30.00  Range: 100   Sec

Partition:   A25        2094 Model 710
CPC Capacity:     640   Weight % of Max: ****   4h Avg:    3   Group:   N/A
Image Capacity:   256   WLM Capping %:    0.0   4h Max:   16   Limit:   N/A

Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
             Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total
*CP                      72.0                         2.9    17.7   20.6
A25            0    17  NO   2.0       13.1    13.4    0.1     2.6    2.7
A29            0    14  NO   2.0       11.0    11.2    0.0     2.2    2.2

*IIP                     13.0                         0.3     0.1    0.3
A25                  NO   1.0        0.0     0.0     0.0     0.0    0.0
```

Figure D-15 RMF CPC Capacity example 2

For zIIP capacity planning information, refer to “Capacity Planning for zIIP Assisted IPSec”, at:

http://www.ibm.com/support/docview.wss?rs=852&uid=swg27009459

zIIP performance projection

If you are running IPSec, zIIP may significantly reduce the CPU utilization of your standard CPs. In planning for zIIP, you need to determine the following:

- How much of your workload is eligible to move to zIIP?
- How many zIIPs would be required to fully handle that load?
- How much CPU busy relief can you expect on your standard CPs?

There are two general methods for projecting zIIP effectiveness:

- If you are already running IPSec, projection is straightforward by using the PROJECTCPU function in z/OS Workload Manager.
- If you are not yet running IPSec, some traffic modeling may be necessary.

Using PROJECTCPU for zIIP performance projection

In our case, we used one logical partition with two CPs and no zIIP.

You need to customize your environment to be able to test this function.

1. Code PROJECTCPU=YES in SYSx.PARMLIB member IEAOPTxx.
2. Issue the following z/OS command to take this parameter in effect dynamically:

   ```
   /SET OPT=xx
   ```

3. Use WLM to create a Service Class for IPSEC (see Example D-5 on page 751).

Example: D-5 IPSECCL definition

```
Service-Class  View  Notes  Options  Help
------------------------------------------------------------------------------
                       Service Class Selection List           Row 1 to 2 of 2
Command ===> ____________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar


Action  Class     Description                      Workload
  __    BATCHHI   Batch Hi importance              TEST01
  __    IPSECCL   IPSec traffic service class      IPSECWK
```

4. Create an IPSEC Workload for this Service Class (see Example D-6).

Example: D-6 Creating IPSECWK Workload

```
Workload  View  Notes  Options  Help
 ------------------------------------------------------------------------------
                          Workload Selection List            Row 1 to 2 of 2
 Command ===> ____________________________________________________________

 Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
               /=Menu Bar
                                                      ----Last Change-----
 Action  Name      Description                        User      Date
   __    IPSECWK   IPSec Workload Group               CS03      2007/10/01
   __    TEST01                                       HAIMO     2005/09/12
 ******************************* Bottom of data ********************************
```

Use the parameters shown in Example D-7.

Example: D-7 IPSECWK parameters

```
***************************** Top of Data ****************************
* Workload IPSECWK - IPSec Workload Group

  Created by user CS03 on 2007/10/01 at 11:50:55
  Last updated by user CS03 on 2007/10/01 at 11:50:55

  1 service class is defined in this workload.

    * Service Class IPSECCL - IPSec traffic service class

      Created by user CS03 on 2007/10/01 at 11:56:39
      Base last updated by user CS03 on 2007/10/01 at 11:56:39

      Base goal:
      CPU Critical flag: NO
#  Duration   Imp  Goal description
       -  ---------  -    ---------------------------------------
       1            3    Execution velocity of 20
```

5. Create a Report Class for the Service Class from Example D-5.

See Example D-8 on page 752.

Example: D-8 Report Class creation

```
Report-Class  View  Notes  Options  Help
------------------------------------------------------------------------------
                       Report Class Selection List            Row 1 to 1 of 1
Command ===> ____________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar
                                                    ----Last Change-----
Action  Name      Description                       User      Date
  __    ZIIPRPT   ZIIP Report class                 CS03      2007/10/01
****************************** Bottom of data ********************************
```

6. Create a Classification Rule for IPSEC (see Example D-9).

   The keyword **TCP** is required and is known by WLM.

Example: D-9 TCP creation

```
Subsystem-Type  View  Notes  Options  Help
------------------------------------------------------------------------------
                Subsystem Type Selection List for Rules     Row 1 to 15 of 15
Command ===> ____________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar

                                                    ------Class-------
Action  Type      Description                       Service   Report
  __    ASCH      Use Modify to enter YOUR rules
  __    CB        Use Modify to enter YOUR rules
  __    CICS      Use Modify to enter YOUR rules
  __    DB2       Use Modify to enter YOUR rules
  __    DDF       Use Modify to enter YOUR rules
  __    IMS       Use Modify to enter YOUR rules
  __    IWEB      Use Modify to enter YOUR rules
  __    JES       Use Modify to enter YOUR rules
  __    LSFM      Use Modify to enter YOUR rules
  __    MQ        Use Modify to enter YOUR rules
  __    OMVS      Use Modify to enter YOUR rules
  __    SOM       Use Modify to enter YOUR rules
  __    STC       Use Modify to enter YOUR rules
  1_    TCP       zIIP workload for TCPIP           IPSECCL   ZIIPRPT
  __    TSO       Use Modify to enter YOUR rules
****************************** Bottom of data ********************************
```

7. As shown in Figure D-10 on page 747, first install the definitions on the WLM couple data set, and then activate the service policy:
   - Select **Utilities** from the top of your ISPF panel.
   - Select **1** `Install definition`, then press Enter.
   - Select **3** `Activate service policy`, then press Enter.

Example: D-10 Install and Activate options

```
Utilities  Notes  Options  Help
--------------------------------------------------
e    1. Install definition                       e
e    2. Extract definition                       e
e    3. Activate service policy                  e
e    4. Allocate couple data set                 e
```

```
e    5. Allocate couple data set using CDS values e
e    6. Validate definition                       e
-------------------------------------------------
```

PROJECTCPU example without zIIP

You need to customize your environment with the following parameters:

1. Insert PROJECTCPU in the IEAOPTxx member in SYSx.PARMLIB (see Example D-11).

Example: D-11 IEAOPTxx member

```
EDIT       SYS1.PARMLIB(IEAOPT00) - 01.02                Columns 00001 00072
Command ===>                                               Scroll ===> CSR
****** ***************************** Top of Data ******************************
001000 ERV=500
001100 CPENABLE=(10,30)
001200 PROJECTCPU=YES
****** **************************** Bottom of Data ****************************
```

2. Modify your TCP/IP profile with the following parameter (see Example D-12).

Example: D-12 zIIP insertion

```
GLOBALCONFIG ZIIP IPSECURITY
IPCONFIG DATAGRAMFWD SYSPLEXROUTING SOURCEVIPA
```

3. Start your IPSEC workload.

   In our case, we use an FTP job batch with multiple PUT statements (see Example D-13).

Example: D-13 FTP batch JCL

```
//FTPBAT2 JOB (999,POK),'Batch FTP',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
/*JOBPARM L=999,SYSAFF=SC33
//FTP      EXEC PGM=FTP PARM='/-d (EXIT'
//SYSTCPD  DD  DISP=SHR,DSN=TCPIPE.TCPPARMS(DATAE33)
//SYSFTPD  DD  DISP=SHR,DSN=TCPIPE.TCPPARMS(FTPSE33)
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*,LRECL=160,BLKSIZE=16000,RECFM=FB
//INPUT    DD  *
 10.1.1.30
 userid 1
 password 2
 ebcdic
 mode b
 site recfm=u blksize=27998 cylinders volume=COMMQ1 unit=3390
 site primary=1611 secondary=50
 PUT 'RC33.HOM.SEQ1.D070926' SEQ1
 PUT 'RC33.HOM.SEQ1.D070926' SEQ2
 PUT 'RC33.HOM.SEQ1.D070926' SEQ3
 PUT 'RC33.HOM.SEQ1.D070926' SEQ5
 PUT 'RC33.HOM.SEQ1.D070926' SEQ6
 quit
```

   1 and 2: Use your own user ID and password.

4. As shown in Example D-14 on page 754, you can verify that the IPSEC traffic goes to CPs instead of zIIP by entering the following command under SDSF:

   ```
   ENC
   ```

Example: D-14 SDSF display - ENC

```
SDSF ENCLAVE DISPLAY  SC33     ALL                        LINE 1-6 (6)
COMMAND INPUT ===>                                              SCROLL ===> CSR
NP   NAME             SSType Status   SrvClass Per PGN RptClass ResGroup   CPU-TIME
     2400000037       STC    INACTIVE SYSSTC     1
     2C00000047       STC    INACTIVE SYSSTC     1
     3400000051       STC    INACTIVE SYSSTC     1
     2000000038       TCP    INACTIVE IPSECCL    1     ZIIPRPT
     2800000048       TCP    INACTIVE IPSECCL    1     ZIIPRPT
     3000000052       TCP    ACTIVE   IPSECCL    1     ZIIPRPT              803.96
```

5. Collect RMF data using the RMF monitor I (see Example D-15).

Example: D-15 Workload activity without zIIP

```
                                W O R K L O A D   A C T I V I T Y
                                                                                                    PAGE   1
     z/OS V1R9              SYSPLEX COMMPLEX          DATE 10/01/2007          INTERVAL 19.59.994   MODE = GOAL
                            RPT VERSION V1R9 RMF      TIME 18.40.00

                                    POLICY ACTIVATION DATE/TIME 10/01/2007 16.03.02

------------------------------------------------------------------------------------------------------------------- SERVICE CLASS(ES

REPORT BY: POLICY=POL01      WORKLOAD=IPSECWK    SERVICE CLASS=IPSECCL    RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE
                                                 DESCRIPTION  =IPSec traffic service class

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----  --SERVICE TIMES--  ---APPL %---  -----STORAGE-----
AVG       3.00  ACTUAL                 0  SSCHRT   0.0  IOC          0  CPU        104.889  CP      8.74  AVG          0.00
MPL       3.00  EXECUTION              0  RESP     0.0  CPU      29756K  SRB          0.000  AAPCP   0.00  TOTAL        0.00
ENDED        0  QUEUED                 0  CONN     0.0  MSO          0  RCT          0.000  IIPCP   8.74  SHARED       0.00
END/S     0.00  R/S AFFIN              0  DISC     0.0  SRB          0  IIT          0.000
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.0  TOT      29756K  HST          0.000  AAP      N/A  --PAGE-IN RATES--
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC     24796   AAP            N/A  IIP      N/A  SINGLE       0.0
AVG ENC   3.00  STD DEV                0                                 IIP            N/A                  BLOCK        0.0
REM ENC   0.00                                          ABSRPTN  8265                                       SHARED       0.0
MS ENC    0.00                                          TRX SERV 8265   PROMOTED     0.000                  HSP          0.0
```

CP 8.74: The two CPs are each averaging 8.74/2 = ~4.37% busy handling this IP workload. Therefore, the percentage of CPU time that was zIIP eligible in this test is 8.74%.

If one zIIP is added to this configuration, then 8.74% of the CPU consumption for this IP workload would move to zIIP, and off the standard CPs. This workload is especially well-suited to zIIP because of the high degree of enclave SRB execution within Communications Server for inbound bulk-data workload.

IIPCP 8.74: This is the percentage of CPU time used by zIIP-eligible work (in the IPSECCL Service Class) running on standard CPs. This statistic is normalized to the capacity of a single standard CP, so the interpretation is that this workload would be handled by one zIIP.

IIP N/A: Because zIIP is not configured.

PROJECTCPU example with zIIP

In our case, we use one logical partition with two CPs and one zIIP. The customization is done in the same way as in “PROJECTCPU example without zIIP” on page 753. See the RMF report in Example D-16.

Example: D-16 Workload activity with zIIP

```
W O R K L O A D   A C T I V I T Y
                                                                                                    PAGE   1
     z/OS V1R9              SYSPLEX COMMPLEX          DATE 10/01/2007          INTERVAL 19.59.923   MODE = GOAL
                            RPT VERSION V1R9 RMF      TIME 18.20.00

                                    POLICY ACTIVATION DATE/TIME 10/01/2007 16.03.02
```

```
---------------------------------------------------------------------------------------------------------- SERVICE CLASS(ES)

REPORT BY: POLICY=POL01      WORKLOAD=IPSECWK    SERVICE CLASS=IPSECCL    RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE
                                                 DESCRIPTION  =IPSec traffic service class

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----  --SERVICE TIMES--  ---APPL %---  -----STORAGE-----
AVG       3.00  ACTUAL                 0  SSCHRT   0.0  IOC         0  CPU       102.467  CP      0.00  AVG           0.00
MPL       3.00  EXECUTION              0  RESP     0.0  CPU     29069K  SRB         0.000  AAPCP   0.00  TOTAL         0.00
ENDED        0  QUEUED                 0  CONN     0.0  MSO         0  RCT         0.000  IIPCP   0.00  SHARED        0.00
END/S     0.00  R/S AFFIN              0  DISC     0.0  SRB         0  IIT         0.000
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.0  TOT     29069K  HST         0.000  AAP      N/A  --PAGE-IN RATES--
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    24225  AAP           N/A  IIP     8.54  SINGLE        0.0
AVG ENC   3.00  STD DEV                0                               IIP       102.467                BLOCK         0.0
REM ENC   0.00                                          ABSRPTN  8075                                  SHARED        0.0
MS ENC    0.00                                          TRX SERV 8075  PROMOTED    0.000              HSP           0.0
```

CP 0.00: All the workload is taken by zIIP.

IIPCP 0.00: There is no workload eligible for zIIP.

IIP 8.54: All workload running for IPSEC is taken by the zIIP processor.

Note: For more information about “Capacity Planning for zIIP Assisted IPSec”, refer to http://www.ibm.com/support/docview.wss?rs=852&uid=swg27009459

The difference between APPL% in the system running without zIIP and the system running with zIIP is due to the Large System Performance Report (LSPR).

- APPL% without zIIP = 8.74
- APPL% with zIIP = 8.54

The formula used to calculate this value is:

$$\text{APPL\% CP} = \frac{\text{CPU + SRB + RCT + IIT + HST -AAP - IIP}}{\text{Interval length}} * 100$$

For more information about RMF, refer to *RMF User's Guide,* SC33-7990.

E

Our implementation environment

We wrote the four *Communications Server for z/OS TCP/IP Implementation* publications at the same time. Given the complexity of this project, we needed to be somewhat creative in organizing the test environment so that each team could work with minimal coordination and interference from the other teams.

In this appendix we show the complete environment used for the four books, and the environment used for this IBM Redbooks publication.

The environment used for all four books

To enable concurrent work on each of the four books, we set up and shared the test environment illustrated in Figure E-1.

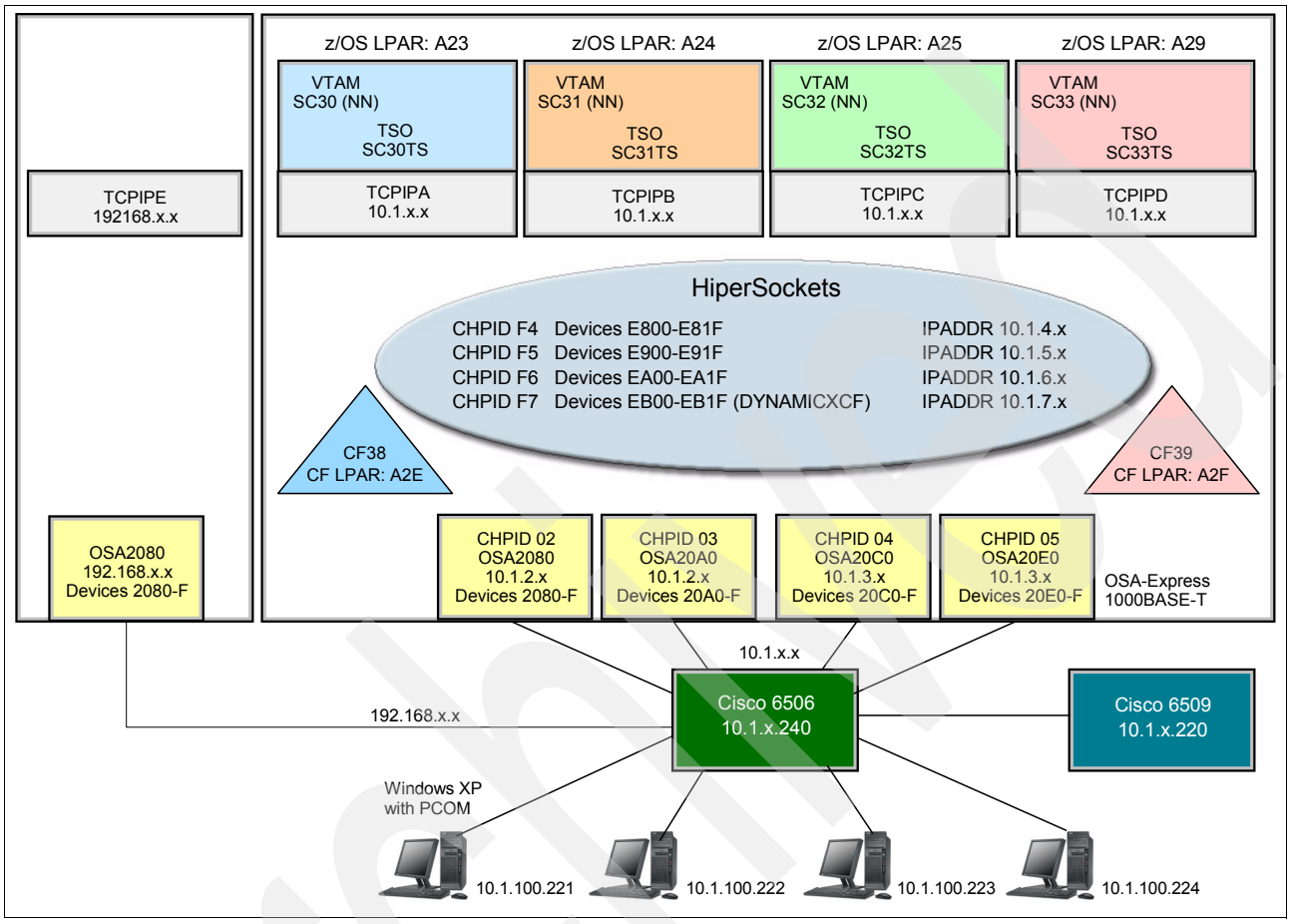


Figure E-1 Our implementation environment

Our books were written (and our implementation scenarios were run) using four logical partitions (LPARs) on an IBM System z9 EC (referred to as LPARs: A23, A24, A25, and A29). We implemented and started one TCP/IP stack on each LPAR. Each LPAR shared the following resources:

- HiperSockets interserver connectivity
- Coupling Facility connectivity (CF38 and CF39) for Parallel Sysplex scenarios
- Four OSA-Express2 1000BASE-T Ethernet ports cross-connected to a pair of Cisco 6500 switches.

Note that this environment is based on the premise of high availability (no single points of failure).

Finally, we shared four Windows workstations, representing corporate network access to the z/OS networking environment. The workstations could be connected to either one of the Cisco switches. For verifying our scenarios, we used applications such as TN3270 and FTP.

The IP addressing scheme used allowed us to build multiple subnetworks so that we would not impede ongoing activities by other team members.

VLANs were also defined to isolate the TCP/IP stacks and portions of the LAN environment; see Figure E-2.

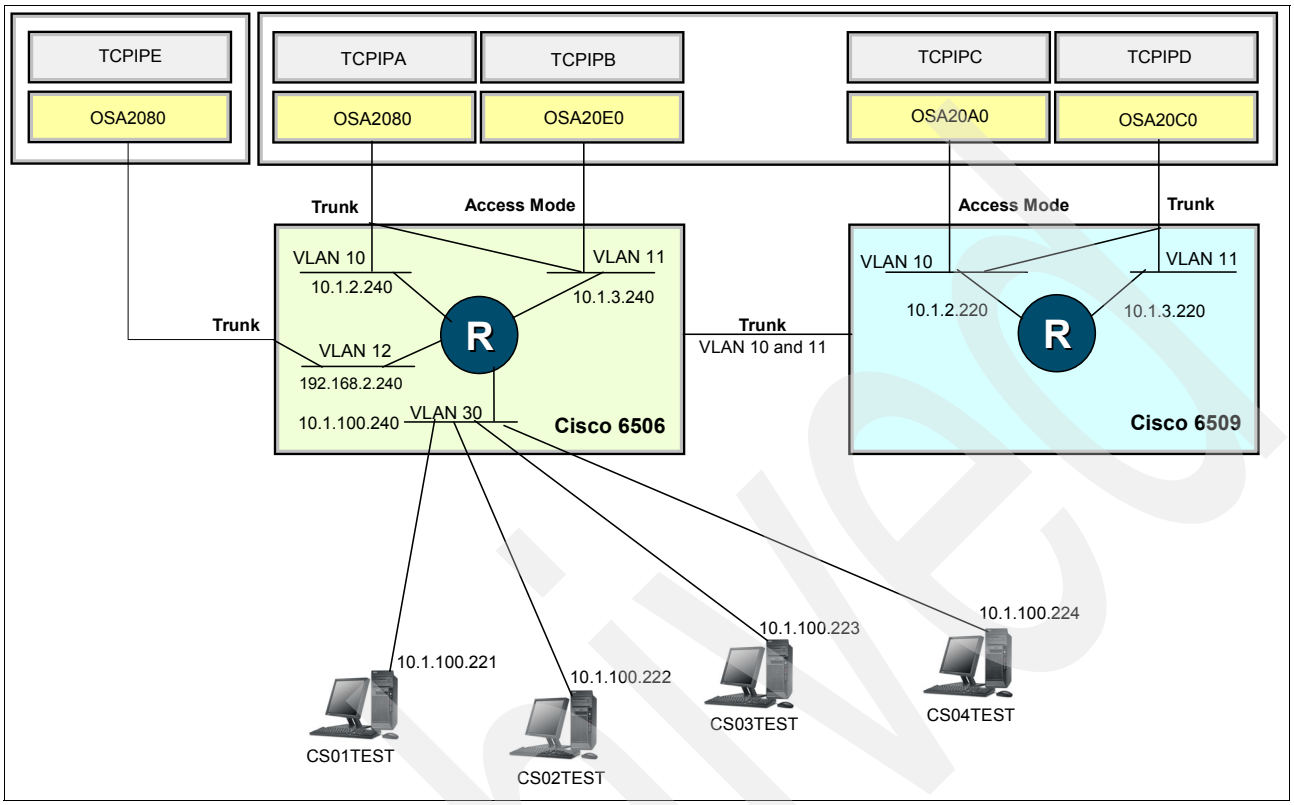


Figure E-2 LAN configuration - VLAN and IP addressing

Our focus for this book

Figure E-3 on page 760 depicts the environment that we worked with, as required for our security and policy-based implementation scenarios. To prevent interference with other activities, some of our scenarios used different addressing schemes and an additional TCP/IP stack. For example, the 192.1.x.x IP address range and TCPIPE were used for some security scenarios.

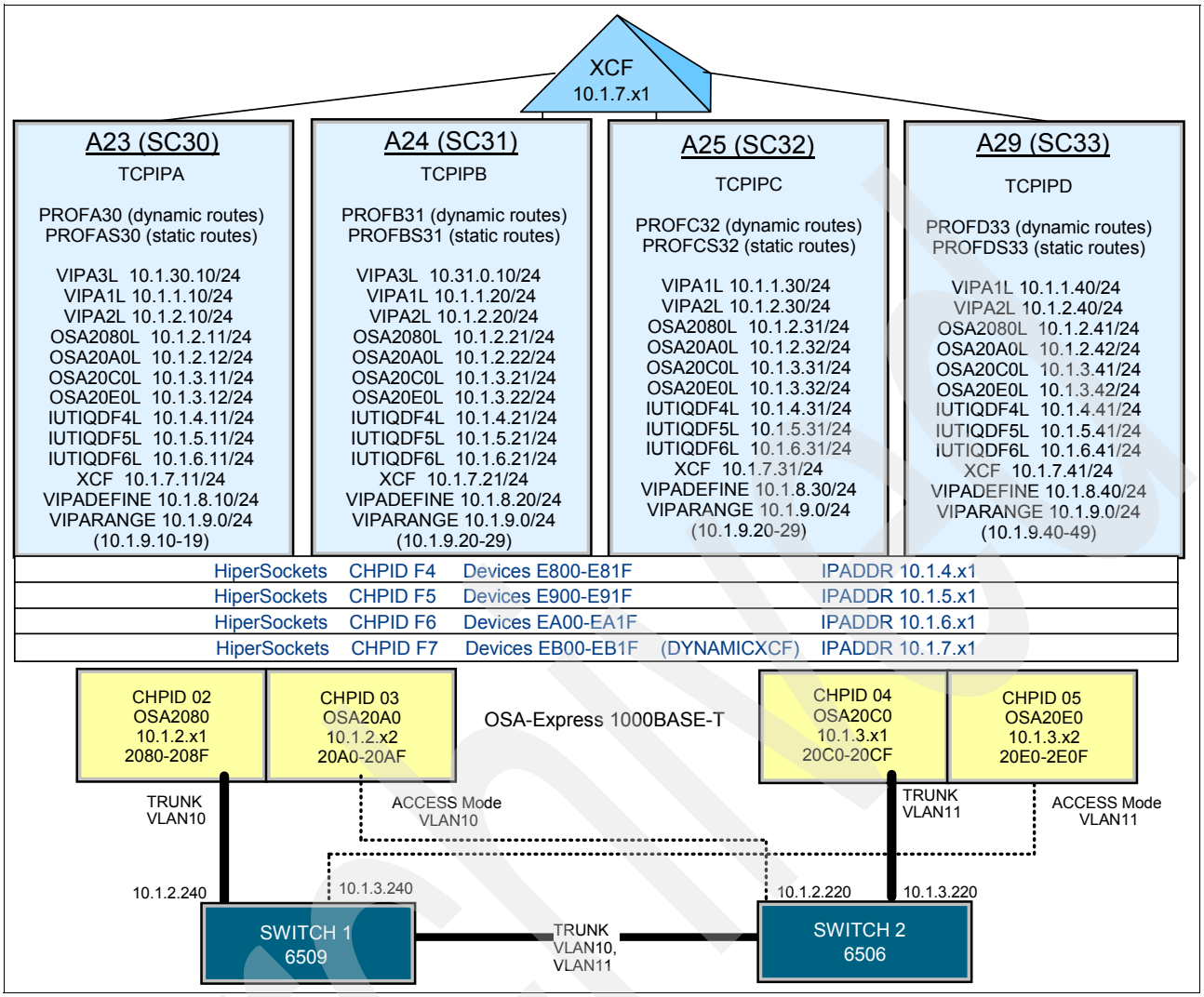


Figure E-3 Our environment for this book

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 763. Note that some of the documents referenced here may be available in softcopy only.

- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *IP Network Design Guide*, SG24-2580
- *Deploying a Public Key Infrastructure*, SG24-5512
- *OSA-Express Implementation Guide*, SG24-5948
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *z/OS Infoprint Server Implementation*, SG24- 6234
- *z/OS Security Services Update*, SG24-6448
- *HiperSockets Implementation Guide*, SG24-6816

Other publications

These publications are also relevant as further information sources:

- *z/OS Migration*, GA22-7499
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS Communications Server: New Function Summary*, GC31-8771
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS V1R8.0 Cryptographic Services ICSF Overview*, SA22-7519
- *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520
- *z/OS V1R8.0 Cryptographic Services ICSF Administrator’s Guide*, SA22-7521
- *z/OS MVS Interactive Problem Control System (IPCS) Commands*, SA22-7594

- *z/OS MVS System Commands*, SA22-7627
- *z/OS Security Server RACF System Programmer's Guide*, SA22-7681
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS Security Server RACF Command Language Reference*, SA22-7687
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Programming Tools*, SA22-7805
- *z/OS UNIX System Services Messages and Codes*, SA22-7807
- *z/OS UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS V1R8.0 XL C/C++ Run-Time Library Reference*, SA22-7821
- *System z9 and zSeries Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935
- *z/OS Cryptographic Services System SSL Programming*, SC24-5901
- *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926
- *HTTP Server Planning, Installing and Using, V5.3*, SC31-8690
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: SNA Operations*, SC31-8779
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS Communications Server: CSM Guide*, SC31-8808
- *z/OS Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS Communications Server: Quick Reference*, SX75-0124
- *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*, by Electronic Frontier Foundation, John Gilmore, editor, 1988, ISBN 13: 9781565925205

Online resources

These Web sites and URLs are also relevant as further information sources:

- z/OS Communications Server product support

  http://www-306.ibm.com/software/network/commserver/zos/support/

- Mainframe networking

  http://www.ibm.com/servers/eserver/zseries/networking/

- z/OS Communications Server product overview

  http://www.ibm.com/software/network/commserver/zos/

- z/OS Communications Server publications

  http://www-03.ibm.com/systems/z/os/zos/bkserv/r9pdf#commserv

- IBM Configuration Assistant for z/OS Communications Server

  http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=D400&uid=swg24013160&loc=en_US&cs=UTF-8&lang=en

- z/OS downloads

  http://www.ibm.com/servers/eserver/zseries/zos/downloads/

- The default behavior of IPsec NAT traversal (NAT-T) is changed in Windows XP Service Pack 2

  http://support.microsoft.com/kb/885407/en-us

- TOOLS - IDSAUTO

  http://www.ibm.com/support/docview.wss?uid=swg24001743

- Manage Intrusion Detection Services with Tivoli Risk Manager

  http://www-1.ibm.com/support/docview.wss?uid=swg24006973

- *Cryptanalysis of MD5 Compress*, by Hans Dobbertin

  http://www.cs.ucsd.edu/users/bsy/dobbertin.ps

- IETF RFC Page

  http://www.ietf.org/rfc.html

- VeriSign

  http://www.verisign.com/repository/crptintr.html

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Redbooks

CS for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Communications Server for z/OS V1R9 TCP/IP Implementation Volume 4: Security and Policy-Based Networking

Understand CS for z/OS TCP/IP security and policy capabilities

See CS for z/OS security and policy implementation examples

Protect your z/OS networking environment

This Communications Server (CS) for z/OS TCP/IP Implementation series provides easy-to-understand, step-by-step how-to guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

With the advent of TCP/IP and the Internet, network security requirements have become more stringent and complex. Because many transactions come from untrusted networks such as the Internet, and from unknown users, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity. In addition, there are certain applications shipped with TCP/IP such as File Transfer Protocol (FTP) that, without proper configuration and access controls in place, could allow unauthorized users access to system resources and data.

This IBM Redbooks publication explains how to set up security for your z/OS networking environment. For more specific information about CS for z/OS base functions, standard applications, and high availability, refer to the other volumes in the series. These are:

- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing*, SG24-7532
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 2: Standard Applications*, SG24-7533
- *Communications Server for z/OS V1R9 TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance*, SG24-7534

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7535-00 ISBN 0738485055