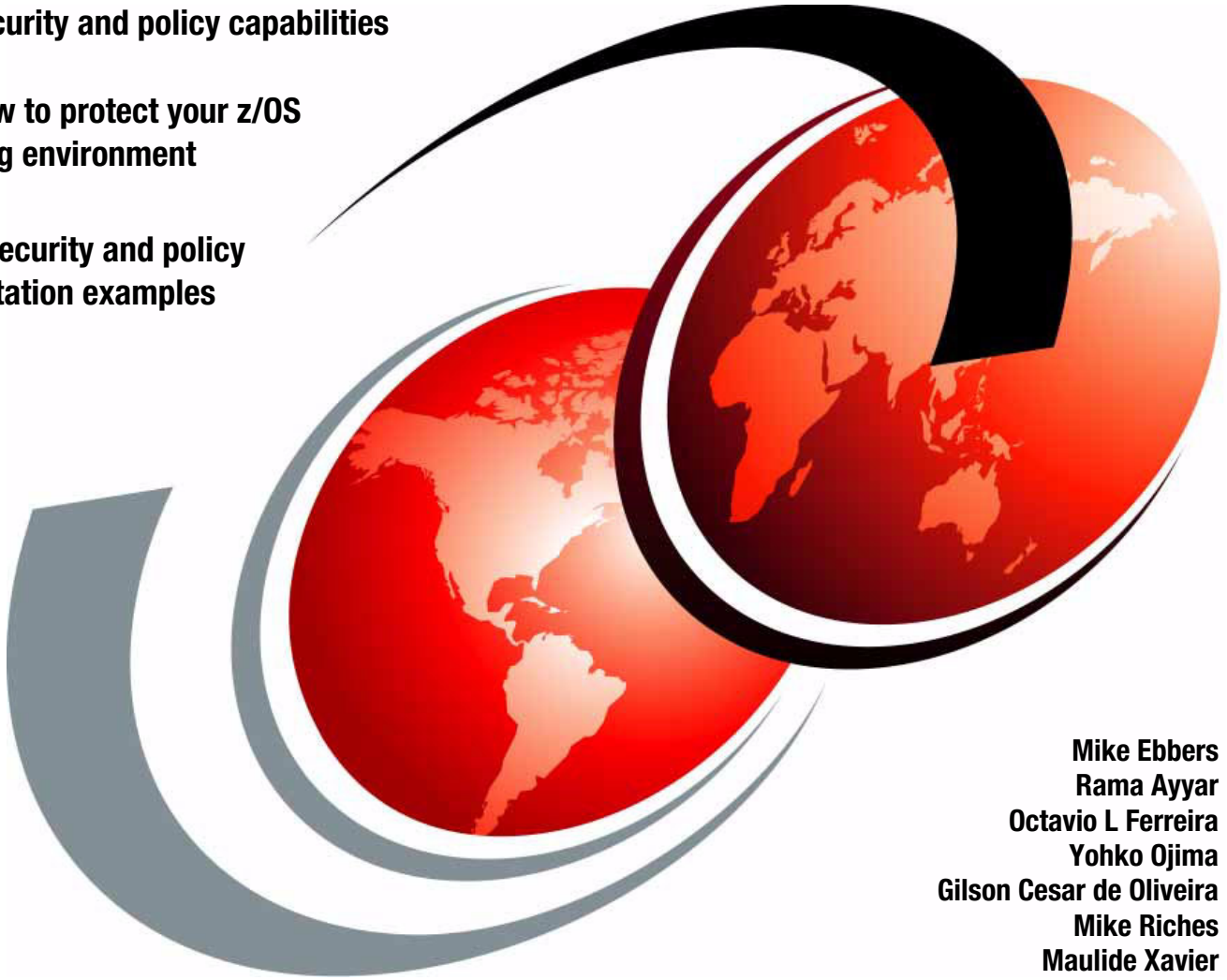
IBM

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking

- Describes z/OS Communications Server TCP/IP security and policy capabilities
- Shows how to protect your z/OS networking environment
- Includes security and policy implementation examples



Mike Ebbers
Rama Ayyar
Octavio L Ferreira
Yohko Ojima
Gilson Cesar de Oliveira
Mike Riches
Maulide Xavier

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM z/OS V1R13 CS TCP/IP Implementation: Volume 4 Security and Policy-Based Networking

March 2012

SG24-7999-00

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

First Edition (March 2012)

This edition applies to Version 1, Release 13 of IBM z/OS Communications Server (product number 5694-A01).

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 illustrate 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.

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. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US 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 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®
DataPower®
DB2®
HiperSockets™
IBM®
IMS™
Language Environment®
MQSeries®
MVS™
NetView®
OMEGAMON®
Parallel Sysplex®
RACF®
RDN®
Redbooks®
Redbooks (logo)  ®
Resource Measurement Facility™
RMF™
System p®
System z®
Tivoli®
VTAM®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries 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 ever more secure, scalable, and highly available mainframe TCP/IP implementations.

The *IBM z/OS Communications Server TCP/IP Implementation* series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of z/OS Communications Server TCP/IP. This IBM Redbooks® publication is for people who install and support z/OS Communications Server. It 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 are from unknown users and untrusted networks such as the Internet, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity. Also, because security technologies are complex and can be confusing, we include helpful tutorial information in the appendixes of this book.

For more specific information about z/OS Communications Server base functions, standard applications, and high availability, see the other volumes in the series:

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7996
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7998

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 that we describe 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 might be useful in your environment. To determine at what level a specific function was introduced, see *z/OS Communications Server: New Function Summary*, GC31-8771.

The team who wrote this book

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

Mike Ebbers is a Consulting IT Specialist and Project Leader at the International Technical Support Organization, Poughkeepsie Center. He has worked with IBM mainframe hardware and software products since 1974 in the field, in education, and in the ITSO.

Rama Ayyar is an independent IT Consultant based in Sydney, Australia and a former employee of IBM Australia. He has also worked for CSC Australia in senior technical roles. He is one of the founding members of HCL India. Rama has over 25 years of experience with the IBM MVS™ operating system. His areas of expertise include TCP/IP, RACF, DFSMS, z/OS, HCD & Configuration Management, Dump Analysis, and Disaster Recovery. Rama has co-authored eight IBM Redbooks. He holds a Master's Degree in Computer Science from the Indian Institute of Technology, Kanpur. Rama has been in the computer industry for more than 35 years.

Octavio L. Ferreira is a Consulting IT Specialist with IBM Brazil. He has 29 years of experience in IBM software support. His areas of expertise include z/OS Communications Server, SNA and TCP/IP, and Communications Server on all platforms. For the last 11 years, Octavio has worked in 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.

Yohko Ojima is an Advisory IT Specialist in IBM Japan and has nine years of experience in enterprise networking. She specializes in TCP/IP and SNA networks with products including z/OS Communications Server, Communications Server for distributed servers, and Communication Controller for Linux on System z, and also devices such as routers and switches. Her responsibility is to provide technical consultations and services on networking design, implementation, and problem solving for customers.

Gilson Cesar de Oliveira is a Senior IT Support Specialist in Brazil working for HSBC as a System Programmer. He holds a degree in Computer Science and specialization in Data Networking. He has 20 years of experience in mainframe networking with expertise in VTAM/SUBAREA/APPN, NCP, TCP/IP, CS for z/OS, JES2/NJE, RACF/RRSF, IBM 3745/46, and OSA-Express.

Mike Riches is a Senior Network Specialist within Maintenance and Technical Support Services, United Kingdom, providing remote technical support and on-site consultancy for IBM clients across Europe. He has 20 years of experience working with IBM networking software, the last nine of those working for IBM. His areas of expertise include z/OS Communications Server, SNA, APPN, HPR, EE and TCP/IP networking.

Maulide Xavier is an IT Specialist in IBM Portugal. He has 13 years of experience in IBM mainframe and networking systems. His current responsibilities include network-related problem-solving in System z and supporting clients to implement a network infrastructure in complex environments. His areas of expertise include z/OS Communications Server, SNA, and TCP/IP networking. He holds a diploma in Business and Economics Applied Math from Instituto Superior de Economia and Gestao and a post graduation in Management of Information Systems from the same school. He is also a member of IBM Networking Software Support in Europe.

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

David Bennin
Richard Conway
Robert Haimowitz
Bill White
International Technical Support Organization, Poughkeepsie Center

Andrew Arrowood
Michael Gierlach
Scott Moonen, the development sponsor
Alan Packett
IBM Communications Server Development, Raleigh

Thanks to the authors of the previous editions of this book:

Finally, we want to thank the authors of the previous *z/OS Communications Server TCP/IP Implementation* series for creating the groundwork for this series: Rama Ayyar, Valirio Braga, WenHong Chen, Demerson Cilloti, Sandra Elisa Freitag, Gwen Dente, Gilson Cesar de Oliveira, Mike Ebbers, Octavio L. Ferreira, Marco Giudici, Adi Horowitz, Michael Jensen, Gazi Karakus**,** Shizuka Katoh, Sherwin Lake, Bob Louden, Garth Madella, Yukihiko Miyamoto, Hajime Nagao, Shuo Ni, Carlos Bento Nonato, Yohko Ojima, Roland Peschke, Joel Porterie, Marc Price, Frederick James Rathweg, Micky Reichenberg, Larry Templeton, Rudi van Niekerk, Bill White, Thomas Wienert, Andi Wijaya, and Maulide Xavier.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email 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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Part 1

SAF-based security

In the first part of this book, 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.

This part contains the following chapters:

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 sets 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 discuss only RACF in this chapter. For the equivalent capabilities in other SAF security products, see that product's documentation.

We discuss the following topics 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 9	How RACF correlates user IDs to STCs
1.5, "Setting up security for daemons in z/OS UNIX" on page 9	How to set up security for daemons working in the z/OS UNIX security environment
1.6, "RACF multilevel security for network resources" on page 9	Basic concepts of MLS
1.7, "Digital certificates in RACF" on page 11	RACF support for keys and certificate management
1.8, "Additional information" on page 11	Resources for additional information

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 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 from 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 SERVAUTH 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 when 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.

  Tip: 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 SERVAUTH 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)` 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 an example in which 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 following RACF commands:

- Defines the RACF profile to restrict the use of the IPV6_NEXTHOP socket option:

  ```
  RDEFINE SERVAUTH EZB.SOCKOPT.*.*.IPV6_NEXTHOP UACC(NONE)
  ```

- Gives access for the user UTSM to use the IPV6_NEXTHOP socket option in the user's programs:

  ```
  PERMIT EZB.SOCKOPT.*.*.IPV6_NEXTHOP CL(SERVAUTH) ID(UTSM) ACCESS(READ)
  ```

You can also protect the use of the socket option by setting up any user to use the socket option, if the user is doing it from a specific program. With this method, you give authority to a program rather than to a user to access the resource (socket option).

The following command allows anyone to access the socket option, provided the user is using program TSOMON:

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

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 13. 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.

See Chapter 2, “Protecting network resources” on page 13, 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 protects the z/OS platform from 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.

1.3.1 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 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 unlikely on z/OS.

1.3.2 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)
```

1.3.3 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 socket option

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

1.3.4 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))
```

1.3.5 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 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 certain 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, see *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 checks whether the user's security level is equal to or above that of the resource. RACF also verifies 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.

1.6.2 Security levels

Also, the user should have the correct 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 might 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, see *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 either by 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 8, "IP Security" on page 245, for information about how to set up digital certificate keys and key rings.

1.8 Additional 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.

We discuss the following topics in this chapter.

Section	Topic
2.1, “The SERVAUTH resource class” on page 14	Basic setup using the RACF SERVAUTH class to protect your resources
2.2, “Protecting your TCP/IP stack” on page 14	How the TCP/IP resource is protected by RACF
2.3, “Protecting your network access” on page 15	How to protect your network
2.4, “Protecting your network ports” on page 18	How RACF protects your ports
2.5, “Protecting the use of socket options” on page 21	How to restrict the use of sensitive socket options
2.6, “Protecting sensitive network commands” on page 22	How to set up security for your sensitive network commands to prevent unauthorized use
2.7, “Protecting FTP” on page 30	The setup to protect FTP resources
2.8, “Protecting network management resources” on page 33	How to use RACF to protect network management resources (such as data collection agents)
2.9, “Protecting miscellaneous resources” on page 33	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

In this section, we explain how to control access to the TCP/IP stack running with a job name of TCPIPD on the z/OS system SC33. 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, the profile name is `EZB.STACKACCESS.SC33.TCPIPD`, as shown in Example 2-1.

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

```
RDEFINE SERVAUTH EZB.STACKACCESS.SC33.TCPIPD UACC(NONE)
PERMIT EZB.STACKACCESS.SC33.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 **setropts raclist(servauth) refresh**. 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
  10.1.100.0/24      MYSUBNET    ;my workstation subnet
  10.1.100.223/32    MYPC        ;my workstation
  DEFAULT 0          WORLD       ;everything else
ENDNETACCESS
```

In this example, access to hosts on subnet 10.1.100.0 is mapped to RACF profile MYSUBNET, access to host 10.1.100.223 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*.*security_zonename*

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.
- *security_zonename* is the name specified in the NETACCESS block.

Tip: 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 15 was on a z/OS image named SC33 with a TCP/IP stack name of TCPIPD. Therefore, you need to define the following profiles:

- EZB.NETACCESS.SC33.TCPIPD.MYSUBNET
- EZB.NETACCESS.SC33.TCPIPD.MYPC
- EZB.NETACCESS.SC33.TCPIPD.WORLD

If you define these profiles to RACF with UACC(NONE), users must be specifically permitted access to these profiles 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, access is automatically allowed. However, if a DEFAULT statement is present, 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 user IDs will 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 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: EZB.NETACCESS.SC33.TCPIPD.WORLD          SECLABEL: <NONE>
MYSUBNET  10.1.100.0/24
  PRFNM: EZB.NETACCESS.SC33.TCPIPD.MYSUBNET       SECLABEL: <NONE>
MYPC      10.1.100.223/32
  PRFNM: EZB.NETACCESS.SC33.TCPIPD.MYPC           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 15. 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 15 shows the configuration.

When the RACF profile names shown in 2.3.1, “Network access control overview” on page 15 have been defined to RACF, we issue the TSO command PING 10.1.100.223 to send a packet to workstation 10.1.100.223. Because we have not been permitted access to the 10.1.100.223 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(CSO6    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  EZB.NETACCESS.SC33.TCPIPD.MYPC CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS: Pinging host 10.1.100.223
sendto(): EDC5111I Permission denied. (errno2=0x7442730C)
```

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

Tip: Even though both the host 10.1.100.223 and its subnet 10.1.100.0 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, 10.1.100.224, we expect a RACF error when the TCP/IP address space does a RACF check for access to the MYSUBNET profile, because this is the most specific entry for that host address. We attempt to ping the IP address 10.1.100.224, and we receive the error as expected (Figure 2-3).

```
ICH408I USER(CSO6    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  EZB.NETACCESS.SC33.TCPIPD.MYSUBNET CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS: Pinging host 10.1.100.224
sendto(): EDC5111I Permission denied. (errno2=0x7442730C)
```

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 10.1.40.20, 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.SC33.TCPIPD.WORLD.

```
 ICH408I USER(CSO6    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  EZB.NETACCESS.SC33.TCPIPD.WORLD CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
CS: Pinging host 10.1.40.20
sendto(): EDC5111I Permission denied. (errno2=0x7442730C)
```

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.
- A PORT or PORTRANGE statement has been coded to reserve the port for the application by job name, which can include an asterisk (*), OMVS, or a TSO user ID. 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 an asterisk (*), which is the wildcard character, is normally used with the SAF keyword.
- A PORT or PORTRANGE statement has been coded to reserve the port for user IDs that have permission to an SAF resource that controls access to that port. In this case, the

user ID of the binding process must be permitted to the resource by the security product. For more information, see 2.4.1, “The PORT/PORTRANGE SAF keyword” on page 19.

Using the PORT or PORTRANGE statements for low ports: If you have a PORT or PORTRANGE statement for any of the low ports, the access provided by that statement supersedes RESTRICTLOWPORTS.

For more information about RESTRICTLOWPORTS, see *z/OS V2R1 Communications Server: IP Configuration Reference*, SC27-3651.

The RESERVED keyword of the PORT and PORTRANGE statement can be used to prevent a port from being used at all. The keyword can also be specified on the PORTRANGE statement. This readily allows an installation to clamp down tightly on usage of ports, if such control is desired.

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.

Specifying the SAF keyword and resource name provides a mapping between a port (and protocol) to an SAF SERVAUTH class resource. A server that is attempting to bind to this port (and protocol) is checked for SAF access to that resource.

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 SC33 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
PORTRANGE 5000  3    TCP USER1* ; Wildcard JOBNAME on PORTRANGE
```

Figure 2-5 PORT statements for stack TCPIPD

Given the PORT statements in Figure 2-5, UDP port 512 is reserved and therefore, is 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 SC33 and the TCP/IP stack name is TCPIPD):

- EZB.PORTACCESS.SC33.TCPIPD.FTP20
- EZB.PORTACCESS.SC33.TCPIPD.FTP21

This form of port reservation might be used when a reserved low port needs to be accessed by many potential users using 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.

SAF protection note: The use of SAF protection on TCP and UDP ports only ensures that user IDs with proper permission can bind to or listen on the given port. It does not provide any enforcement over the purpose for which the port is used. So in the previous example, as long as a user ID is authorized to use TCP port 21, that user ID can use the port for any purpose—as an FTP server or for any other program. For this reason, using dedicated user IDs for various server processes allows better control over the ports.

The FTP daemon is the only user that needs to access the FTP control port 21. Thus, the daemon should have access to the RACF SERVAUTH profile EZB.PORTACCESS.SC33.TCPIPD.FTP21. If the FTP server does not have access to the profile, you get a RACF error similar to that shown in Figure 2-6.

```
S FTPDD
$HASP100 FTPDD    ON STCINRDR
IEF695I START FTPDD    WITH JOBNAME FTPDD    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 FTPDD    STARTED
$HASP395 FTPDD    ENDED
+EZY2714I FTP server shutdown in progress
+EZYFT59I FTP shutdown complete.
ICH408I USER(TCPIP   ) GROUP(TCPGRP  ) NAME(####################) 924
  EZB.PORTACCESS.SC33.TCPIPD.FTP21 CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
+EZY2714I FTP server shutdown in progress
+EZYFT59I FTP shutdown complete.
```

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 19, then all 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.SC33.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 19. 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
5000  TCP  USER1*   DAR      05000-05002
00500 UDP  IKED     DA
00512 UDP  RESERVED DA
04500 UDP  IKED     DA
8 OF 8 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 SC33.
- *tcpname* is the jobname of the TCP/IP stack. Our stack was TCPIPD.

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

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

```
RDEFINE SERVAUTH EZB.SOCKOPT.SC33.TCPIPD.SO_BROADCAST UACC(NONE)
PERMIT EZB.SOCKOPT.SC33.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 powerful and should be authorized only for 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 might want to restrict these commands so that people are unable to obtain information about your TCP/IP configuration, perhaps in preparation for an attack.

From the z/OS console (or from TSO and IBM NetView®), you can use the EZACMD System REXX command to issue selected z/OS Communications Server UNIX shell commands:

- ipsec
- nssctl
- pasearch
- ping
- trmdstat

You might want to restrict access to issuing some or all of these UNIX shell commands.

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

Tip: 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 might 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.

See Table 2-1 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 23. At this stage, no user has 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. 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 shows 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 24. 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 on page 25 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 23 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 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 a generic profile with an asterisk (*) as the last qualifier to cover all NETSTAT/onetstat commands. Example 2-7 shows how this is done using RACF commands, assuming that the system name is SC33 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.SC33.TCPIPD.*) UACC(NONE)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

The **SETROPTS GENERIC(SERVAUTH)** command activates generic profile-checking for the SERVAUTH class. Checking needs to be done only once; it is probably already active at your installation.

Protecting NETSTAT/onetstat at the command option level

You might 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 SC33.

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

```
RDEFINE SERVAUTH (EZB.NETSTAT.SC33.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 when 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 SC33 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.

Tip: In the discussions that follow, where 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.SC33.TCPIPD.* was added to the SERVAUTH class with UACC(NONE), as shown in Example 2-7 on page 27. 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.SC33.TCPIPD.HOME, but because that did not exist, the profile EZB.NETSTAT.SC33.TCPIPD.* was checked.

Example 2-9 Access denied by generic profile

```
NETSTAT HOME TCP TCPIPD

ICH408I USER(CS06    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  EZB.NETSTAT.SC33.TCPIPD.HOME CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  FROM EZB.NETSTAT.SC33.TCPIPD.* (G)
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
EZZ2385I Access to Netstat HOME denied - SAF RC is 00000008
```

Defining the NETSTAT profile to protect the NETSTAT HOME command

The specific profile EZB.NETSTAT.SC33.TCPIPD.HOME was added to the SERVAUTH class with UACC(NONE), as shown in Example 2-8 on page 27. At this stage, no user had any access to this profile. User CS06 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.SC33.TCPIPD.HOME rather than EZB.NETSTAT.SC33.TCPIPD.*.

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

```
ICH408I USER(CS06    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  EZB.NETSTAT.SC33.TCPIPD.HOME CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
EZZ2385I 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 on page 28 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

```
CS06 @ SC33:/u/cs06>onetstat -h -p tcpipd
EZZ2385I Access to Netstat -h denied - SAF RC is 00000008
CS06 @ SC33:/u/cs06>
```

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.image.policy_type* in SERVAUTH class to individually protect each policy type, where:

- *sysname* is the z/OS SMFID.
- *image* is TCP name, policy client name, or import request name for policy information that is being requested.
- *policy_type* is the policy type, which can be one of the following types:
  - QOS for Policy QoS
  - Intrusion detection system (IDS) for Policy IDS
  - IPSec for Policy IPSec
  - TTLS for Policy AT-TLS
  - Routing for Policy 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.tcpname.command_type*
- EZB.IPSECCMD.*sysname*.DMD_GLOBAL.*command_type*
  - *command_type* can be DISPLAY or CONTROL. You can also use * for both.
  - DMD_GLOBAL means control access of the IPsec command for global defensive filters.

2.6.5 EZACMD console command security

EZACMD is delivered as a compiled REXX program in two separate system libraries:

- One library is SYS1.SAXREXX, which is the System REXX library. This library is a VB, LRECL=255 library.

  SYS1.SAXREXEC is used from the z/OS console by means of the System REXX infrastructure, which requires a VB, 255 library.
- The second library is tcpip.SEZAEXEC, which is the z/OS Communications Server REXX library. This library is an FB, LRECL=80 library.

  tcpip.SEZAEXEC is used from TSO and NetView.

In addition to the RACF SERVAUTH profiles available to protect system administrator commands, it is also possible to protect the EZACMD command from being issued from unauthorized z/OS console or SDSF users using the OPERCMDS SAF class.

Defining OPERCMDS class for EZACMD System REXX command

The resource entity for the **MODIFY AXR,<exec name>** or ***command_prefix*<exec name>** command is MVS.SYSREXX.EXECUTE.<exec name> under the OPERCMDS class. We created a RACF profile to protect the EZACMD exec name as shown in Example 2-12.

Example 2-12 OPERCMDS class profile for EZACMD

```
RDEFINE OPERCMDS (MVS.SYSREXX.EXECUTE.EZACMD) UACC(NONE)
SETROPTS GENERIC(OPERCMDS) REFRESH
SETROPTS RACLIST(OPERCMDS) REFRESH
```

This resulted in a RACF authorization failure for a user without READ access to the profile as shown in Example 2-13. Our command prefix defined in SYS1.PARMLIB(AXRxx) is REXX&SYSCLONE. So, to send commands to LPAR SC33, we use REXX33.

Example 2-13 EZACMD authorization failure

```
REXX33EZACMD 'ipsec -p TCPIPE -k display'

ICH408I USER(CSO2    ) GROUP(SYS1    ) NAME(BILL WHITE          )
  MVS.SYSREXX.EXECUTE.EZACMD CL(OPERCMDS)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )

AXR0205I EZACMD EXEC NOT AUTHORIZED
```

2.6.6 Additional information

For more information about the profile names used to protect the various commands and options, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781. For more information on System REXX, see *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608.

2.7 Protecting FTP

In this section, we discuss how to protect FTP using RACF.

2.7.1 Restrict certain users from logging into FTP server

By default, any user ID that is valid on the z/OS host can log in to FTP. For security purposes, you might want to allow only certain user IDs to log in to FTP on a certain host. z/OS FTP currently provides the following methods to allow only certain users to log in to FTP:

- Set up the FTP server for TLS level 3 authentication when sessions are secured with TLS.
- Code and install an FTCHKPWD exit routine that screens the user IDs that are allowed to log in to FTP.
- Code VERIFYUSER TRUE in the server's FTP.DATA and use RACF profile to protect the FTP server port.

In the following sections, we discuss how to restrict users from logging in to an FTP server using the third method.

RACF profile details

The SERVAUTH class is used to define a profile to controls ability to access FTP server. The format of the profile name is:

```
EZB.FTP.sysname.ftpdaemonname.PORTxxxx
```

Where:

- *sysname* is the z/OS image name.
- *ftpdaemonname* is the FTP daemon name.
- *PORTxxxx* specifies the port number that you want to protect (or you can use PORT* to protect all the FTP server ports with a single profile).

Define RACF FTP server port profile

To specify protection of the FTP server port, define the SERVAUTH profile as shown in Example 2-14, assuming that FTP daemon FTPDD is running in system SC33 using port 20 and port 21.

Example 2-14 Sample RACF command to protect FTP server port

```
RDEFINE SERVAUTH (EZB.FTP.SC33.FTPDD1.PORT*) UACC(NONE)
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS GENERIC(SERVAUTH) REFRESH
```

Enable VERIFYUSER in FTP Server

The VERIFYUSER statement for the server's FTP.DATA allows you to configure FTP to require all user IDs logging in to FTP to have at least READ access to the FTP server port profile:

- When VERIFYUSER TRUE, a user ID needs to have READ or greater access to the RACF FTP server port profile to log into the FTP server.
- When VERIFYUSER FALSE, which is the default value, the server ignores the server port profile except when a session is secured with TLS, and TLS level 3 client authentication has been configured.

To restrict a user from logging in to the FTP server by RACF FTP server port profile, specify VERIFYUSER TRUE in the FTP server FTP.DATA.

FTP server port security scenario

The system name in our testing environment was *SC33*, and the FTP start procedure name was *FTPDD*. Because we defined the FTP server port profile EZB.FTP.SC33.FTPDD1.PORT* with UACC(NONE), and the VERIFYUSER statement in server FTP.DATA was set to TRUE, no user ID can have the authority to log in to this FTP server. A user who logs in will see the error message shown in Figure 2-14.

```
 EZA1459I NAME (10.1.1.40:CS06):
cs06
 EZA1701I >>> USER cs06
 331 Send password please.
 EZA1789I PASSWORD:
 EZA1701I >>> PASS
 530 PASS command failed
 EZA1460I Command:
```

Figure 2-14 PASS command error message from unauthorized user log in FTP server

Also in the system log, you can find the RACF messages shown in Figure 2-15. RACF checked the EZB.FTP.SC33.FTPDD1.PORT21 profile, but because that profile did not exist, the profile EZB.FTP.SC33.FTPDD1.PROT* was checked instead.

```
ICH408I USER(CS06    ) GROUP(SYS1    ) NAME(BILL WHITE          ) 378
  EZB.FTP.SC33.FTPDD1.PORT21 CL(SERVAUTH)
  INSUFFICIENT ACCESS AUTHORITY
  FROM EZB.FTP.SC33.FTPDD1.PORT* (G)
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Figure 2-15 RACF error from unauthorized user log in FTP server

After user CS06 was given READ access to the profile EZB.FTP.SC33.FTPDD1.PORT*, as shown in Example 2-15, that user can log in to the FTP server FTPDD1 in system SC33.

*Example 2-15 Giving READ access to user for EZB.FTP.SC33.FTPDD1.PORT**

```
PERMIT EZB.FTP.SC33.FTPDD1.PORT*  CLASS(SERVAUTH) ID(CS06) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS GENERIC(SERVAUTH) REFRESH
```

2.7.2 Protect other FTP related resources

In this section, we discuss other FTP related resources that can be protected by RACF.

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 FTP daemon name.

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.

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 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.

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, although 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 user IDs of those subagents that should be permitted to connect through TCP to the SNMP Agent.

Alternatively you could use the **-C** parameter in the SNMP agent start-up to grant or revoke access to subagents not defined as superusers. For more information, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776

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 user IDs 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 DVIPA activation and movement Control

The profile that protects the DIVIPA is as follows:

```
EZB.BINDDVIPARANGE.sysname.tcpname.resname
```

With the optional SAF keyword **resname** on the VIPARANGE statement, you can use granular profiles. By creating separate profiles for separate VIPARANGE statements, you can authorize different applications to various DVIPAs.

The VIPARANGE statements are as follows:

```
VIPARANGE DEFINE 255.255.255.0     20.20.20.5 SAF APPLX
VIPARANGE DEFINE 255.255.255.255   30.30.30.8 SAF APPL1
VIPARANGE DEFINE 255.255.255.0      40.40.40.0
```

The corresponding RACF profiles are as follows:

```
EZB.BINDDVIPARANGE.sysname.tcpname.APPLX
EZB.BINDDVIPARANGE.sysname.tcpname.APPL1
```

EZB.BINDDVIPARANGE.sysname.tcpname

2.9.4 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 following profile protects this resource:

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

2.9.5 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 following profile protects this resource:

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

2.9.6 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
```

2.9.7 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`*

2.9.8 RPCBIND application registration control

Registration control provides an additional level of security for RPCBIND when registering or unregistering applications. You can define a SERVAUTH resource profile that controls which applications or users can register and unregister with RPCBIND. RPCBIND is also supported in a multilevel secure (MLS) environment. When a target assistance procedure is running in a multilevel security environment, RPCBIND switches to the security label of the requester to ensure that the correct port-of-entry information is presented to the target server. The profile that protects this resource is:

`EZB.RPCBIND.`*`sysname`*`.`*`rpcbindname`*`.REGISTRY`

Tip: In environments that do not implement MLS, defining a RACF profile is optional. If you do define a SERVAUTH profile, all applications that register and unregister with RPCBIND must be invoked by user IDs that have READ access to the profile. However, in an MLS environment, all applications cannot register with RPCBIND by default. After you define a SERVAUTH profile with READ access for user IDs that are associated with the applications, RPCBIND accepts registration and deregistration of applications.

Part 2

Managing security

In this part of the book, we describe how to implement and use digital certificates with a z/OS system. We introduce the terms and technologies that you use when creating certificates and defining the repositories in which they reside. We also explain how to set up the various servers and clients to use security certificates for specific security technologies. Further, we provide examples of certificate definitions and the utilities to manage them. Finally, we show how to use digital certificates in a policy-based z/OS environment.

As you read, the following points are important to keep in mind.

- Digital certificates are used in a variety of applications. Within z/OS Communications Server, the two main applications are SSL/TLS and IPsec.
- This chapter focuses on the usage of digital certificates in SSL/TLS. However, the basic concepts also apply to IPSec also.
- Unlike SSL/TLS, z/OS IPSec support supports only SAF-based (such as RACF) key rings; it does not support certificates managed through the gskkyman utility.

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 provides 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, see 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

We discuss the following topics in this chapter.

Section	Topic
3.1, "Digital certificates overview" on page 40	Basic concepts of digital certificates
3.2, "Digital certificate types" on page 42	Definition and use of types of digital certificates
3.3, "Configuring the utilities to generate certificates in z/OS" on page 47	Identification of utilities and their command structures that generate digital certificates in z/OS
3.4, "Using certificates in sample IBM environments" on page 55	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 relevant RACF commands. Because **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	Location	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 can 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 keys (contained within the X.509 certificate) and private keys 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. 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 digitally sign the certificate, the certificate issuer first generates a unique fixed-length binary value called a message digest based on the certificate contents using a hashing algorithm like SHA1 or SHA2. The exact length of the digest depends on the algorithm. The digest value is unique based on the input data (the certificate contents), but it is also re-creatable, such that another user of the same hashing algorithm and the same input data will generate the exact same value. The message digest is then encrypted using the issuer's private key, creating the issuer's digital signature. The hashing algorithm is included in the certificate.

When a digital certificate is received as proof of identity, the receiver uses the issuer's public key (found in the issuer's own certificate) to decrypt the message digest. Next, it uses the hashing algorithm specified in the certificate to create its own message digest of the certificate contents. If the newly-generated digest exactly matches the decrypted digest, then the receiver is assured that the certificate was actually signed by the trusted issuer because the issuer's public key successfully decrypted data that was encrypted using its private key, and that the certificate has not been altered since it was issued because the message digests match.

But how did this hypothetical remote endpoint come to already possess the public key that was able to decrypt the digest? The public key must 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 42.

3.2 Digital certificate types

We describe the following digital certificate types in this section:

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

We also discuss how to obtain a digital certificate.

3.2.1 Certificate authority certificates

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

root CA A root CA is the 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 46). 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 to sign and validate other internal certificates.

Well-known certificates

In 3.1.2, “How digital certificates work” on page 41 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 would be 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:

- As part of the environment setup process, a CA certificate (intermediate or root) can be used to sign another certificate.
- 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

Private keys must be carefully protected such that they are only known by their owners. Public keys, however, can be freely distributed to others. A subject uses its private key to digitally sign data in an effort to prove its identity (similar to the process described previously). If a private key is properly protected, then the act of successfully decrypting data with a given public key proves that it was encrypted by the owner of the key pair. Likewise, data encrypted by a public key can only be decrypted by its associated private key. This provides an effective means to secure data such that only the owner of the key pair can see it. If anyone else has access to a subject's private key, they could easily impersonate the actual owner.

The type of encryption that uses public/private key pairs is called *asymmetric encryption* due to the asymmetry of the keys involved. Asymmetric encryption, although effective, is computationally expensive and is therefore rarely ever used for encrypting and decrypting

large amounts of data. Rather, asymmetric encryption is typically used to protect small amounts of data such as message digests or encryption keys for other algorithms that can be used to efficiently and effectively encrypt large amounts of data. Such algorithms are called *symmetric encryption* algorithms because both the sender and the receiver of the data use the same key to encrypt and decrypt, respectively.

For more information about cryptography, including public and private key pairs, see Appendix A, “Basic cryptography” on page 805.

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 can 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:

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 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. This level cannot be used if the z/OS server is using a key database that is created 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 (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 certain 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 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 another 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. For example, 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.

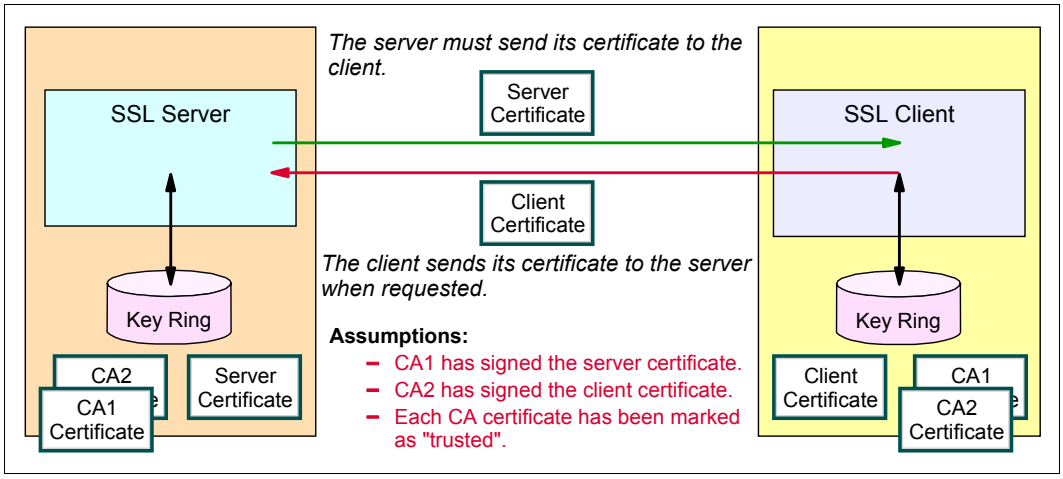


Figure 3-1 SSL certificate management: CA-signed certificates

The example in Figure 3-1 has the client sending a certificate back to the server, implying 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) 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.

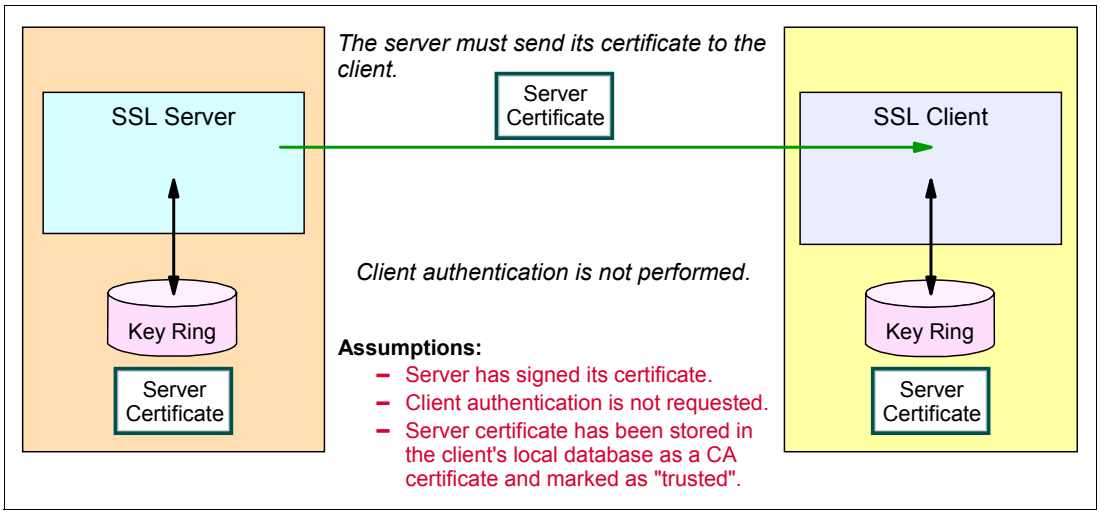


Figure 3-2 SSL certificate management: Self-signed certificate without client authentication

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 *z/OS Cryptographic Services System SSL Programming*, SC24-5901. For information about the RACDCERT command, see *z/OS Security Server RACF Command Language Reference*, SA22-7687.

We discuss the following certificate management utility topics in this section:

- Utilities in z/OS for managing certificates
- Digital certificate field formats
- Using the RACF RACDCERT command
- Using 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, and 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, and certificate requests for other CAs.

Using RACF key rings is the preferred method in many organizations 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.
  - 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.

> **Tip:** Choose an expiration 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 a formal reminder in place so that an expired certificate does not come as a surprise.

Certain 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 separate 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, and private keys can 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. 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.

We discuss the following topics in this section:

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

RACDCERT command format

The RACDCERT command uses the following 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 example, 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

  Associated with CAs and are used to verify s in other certificates.

- Site certificates

  Associated with a location composed 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

  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 windows 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 might or might 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 separate user ID.

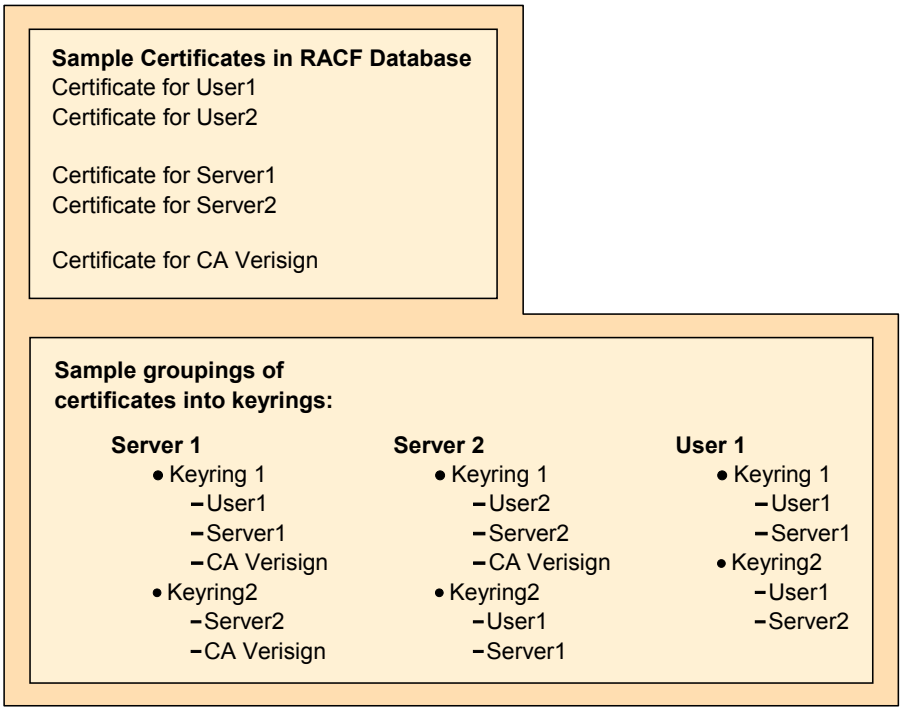


Figure 3-3 Example showing how key rings contain pointers to certificates

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 can 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.

Tip: 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 the client certificate must be in trusted status in the RACF database.

Tip: 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

Although 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/key ring. (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 45.

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.

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 using the HFS.

In the examples shown later in this chapter, we assume that a **gskkyman** key database is 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 53 shows the initial window. 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** window.

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, as 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 owner's 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 user's RACF user ID is `CS08`. The user's user ID is only needed when you are storing client certificates in RACF using RACDCERT.

The sections that follow show how to create, verify, and manage the indicated certificate types:

- Host On-Demand and certificates
- Shared site certificate and shared key ring
- Self-signed certificates
- Internal (local) certificate authority
- External (well-known) certificate authority

Tip: Using a shared site certificate connected to a shared key ring is the usual certificate management scheme because it provides the most efficient and least complex method of certificate management. As mentioned earlier, this does mean that the same private key can 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.

3.4.1 Host On-Demand and certificates

A 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, see the Host On-Demand product documentation. Host On-Demand, and many other SSL/TLS-capable clients, are 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 workload. 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 a unique user ID, the burden of key ring administration grows. 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 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, and 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. Therefore, you might 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 similar to the Internal CA signed server certificate described in “Internal-CA signed server certificate RACDCERT procedure” on page 82. 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 is needed at the client. The steps are as follows:

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

```
//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(DIGTCERT) REFRESH
  RACDCERT CERTAUTH LIST /*
```

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

In this figure, the number corresponds to the following information:

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

In this figure, the numbers correspond to the following information:

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` indicates 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 can 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

In this figure, the numbers correspond to the following information:

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, and 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 is sent using 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 83 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 84 must be imported into the key database as a trusted certificate. See Figure 3-13.

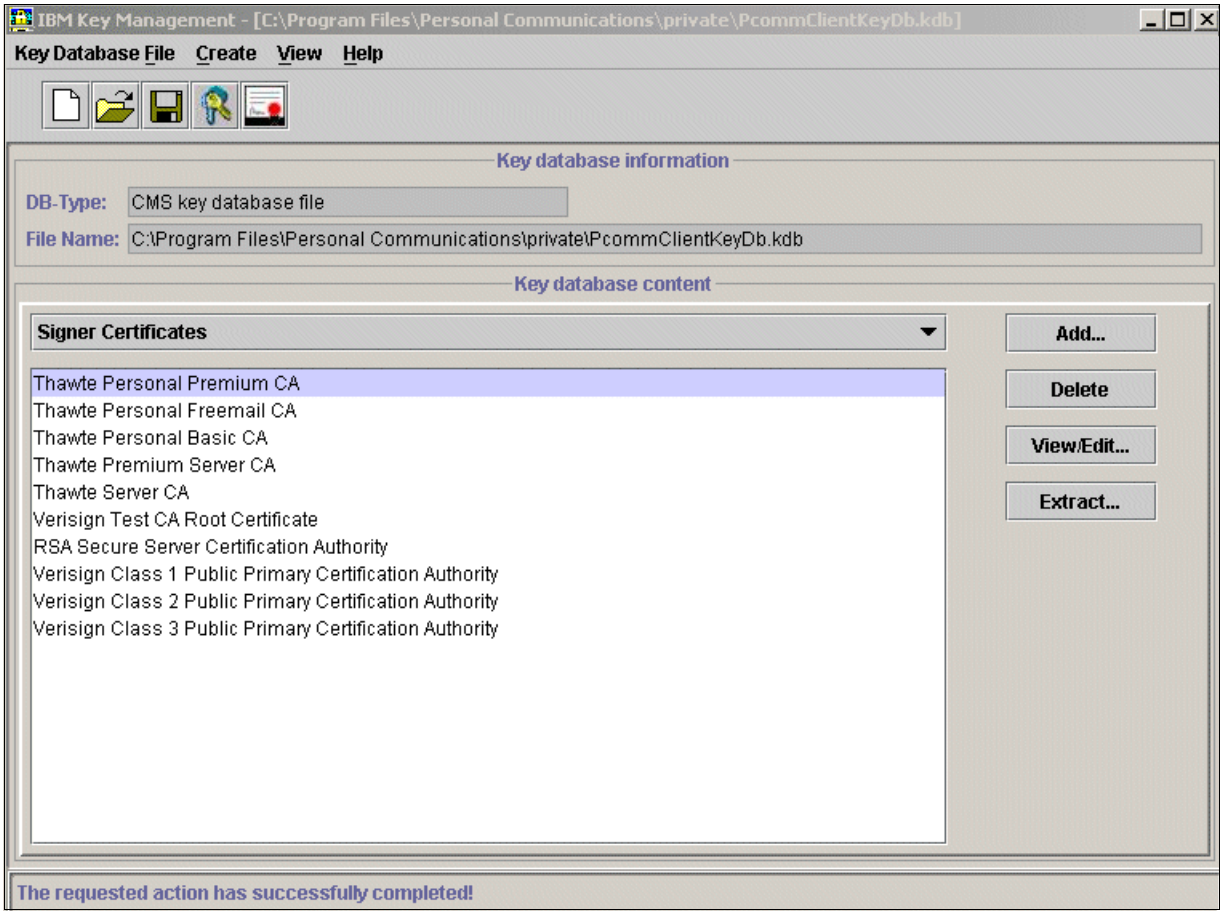


Figure 3-13 Personal Communications client certificate management window

Depending on the type of client, there are a number of 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. After the key database file is opened, the window in Figure 3-14 opens. Import the certificate using the **Add** button.

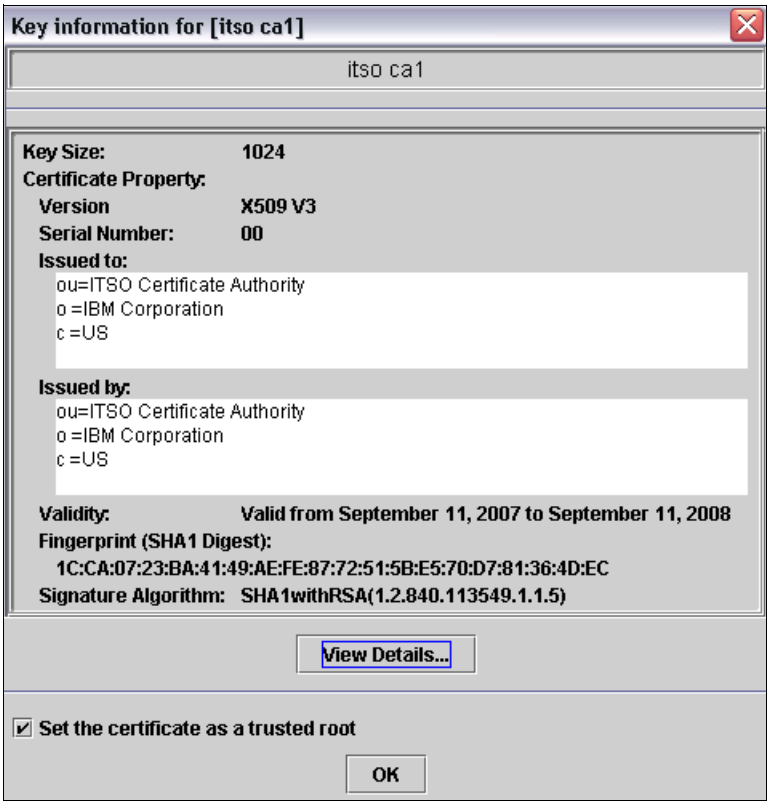


Figure 3-14 Personal Communications client: display of imported server certificate

When the certificate is added, the window in Figure 3-14 then 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 check mark.

10. Test the client-to-server connection. As the example in Figure 3-15 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.

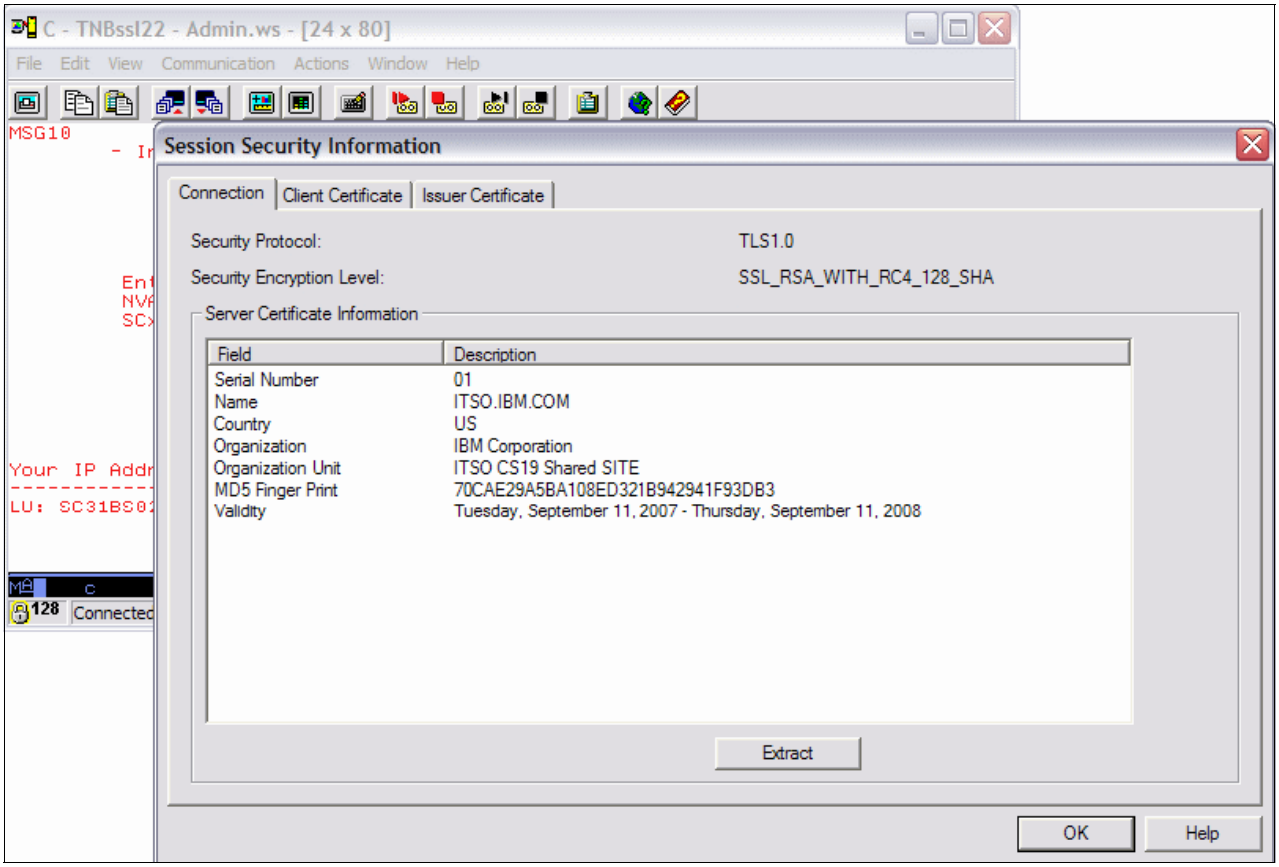


Figure 3-15 Personal Communications client: Site certificate and signer's details

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 82.

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, see the following chapters:

- Chapter 16, "Telnet security" on page 677
- Chapter 17, "Secure File Transfer Protocol" on page 719

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, because 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 81, 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 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. When 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 68.

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 48.

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 steps necessary to create the key ring for the server.

   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.

```
//CS081   JOB 'SET UP TN3270 CERT','CS08',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

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

Tip: 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.

```
//CSO81   JOB 'EXPORT SERVER CERT','CSO8',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('CSO8.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 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 to validate the same certificate when presented by a server in an SSL exchange. Depending on the number of clients in an enterprise, this can 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 (a TN3270 client), select the **Certificate Management** or **Certificate Wizard** icon from the Personal Communications (PCOMM) utilities folder. This displays the window (after the key database file is opened) shown in Figure 3-13 on page 62. Import the certificate by

clicking **Add**. When the certificate is added, the window shown in Figure 3-19 shows 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).

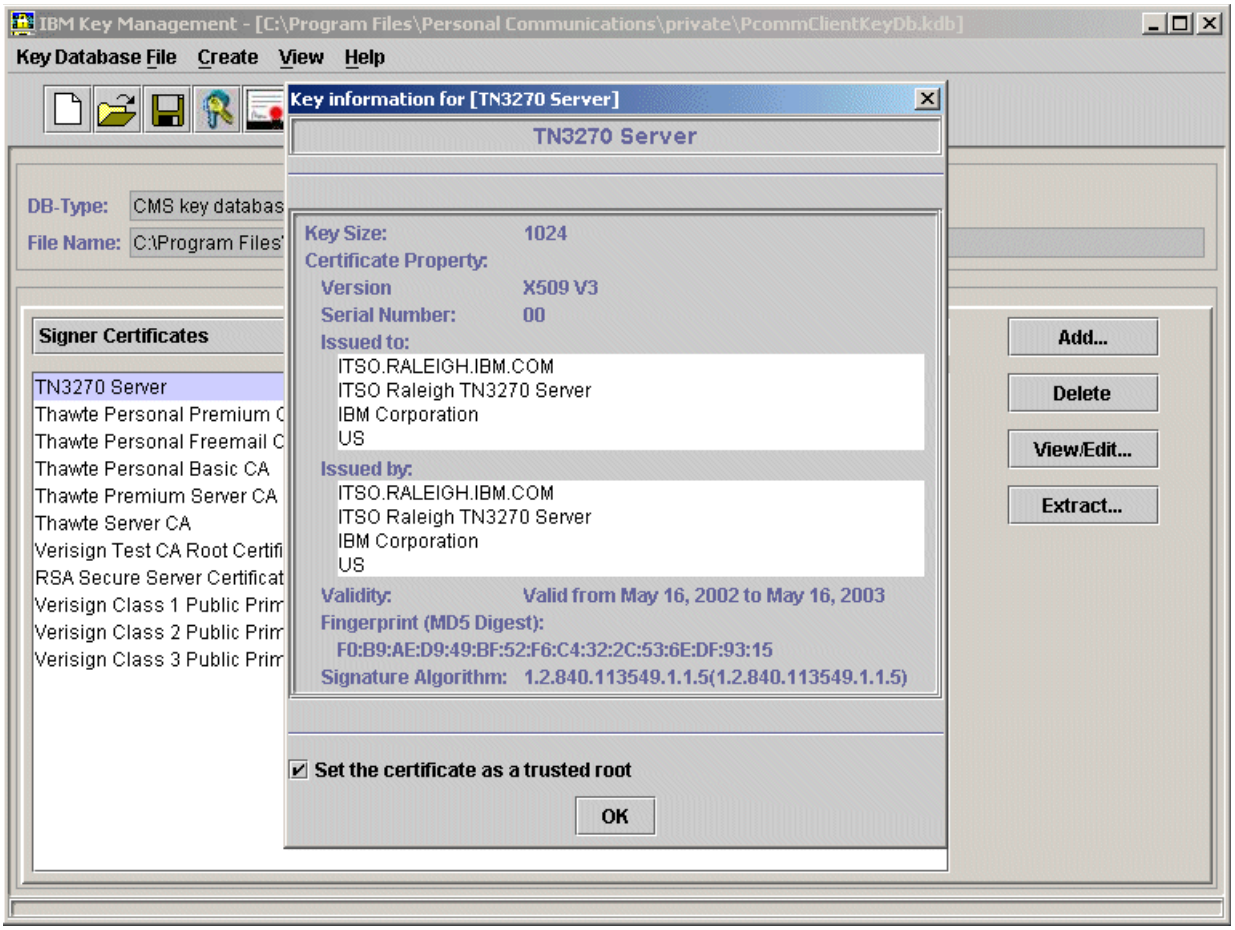


Figure 3-19 Personal Communications client: display of imported server certificate

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 64 for an example.

Although this section showed how to generate the certificate for a TN3270 server, how to export to the client, and how to place the certificate in the client's key database as a trusted certificate, the procedure for any client or 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 65. Although RACDCERT is used on the z/OS endpoint, the certificate will be created at the client endpoint.

A good practice is 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.

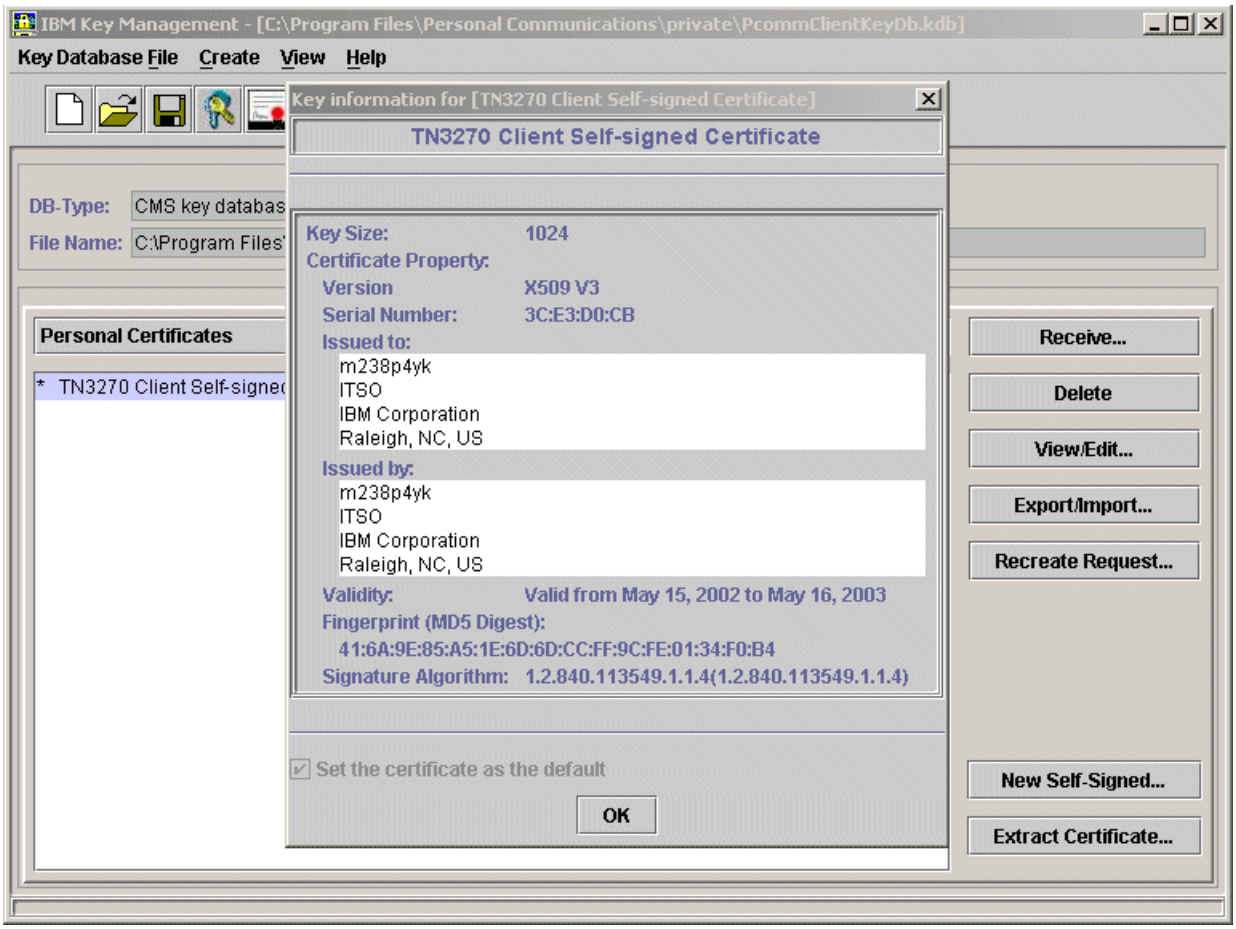


Figure 3-20 Personal Communications client: Newly added self-signed client certificate

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 on page 69 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. Note that the Data Type field specifies Base64-encoded ASCII, and that the certificate will be written to the `cert.arm` file.

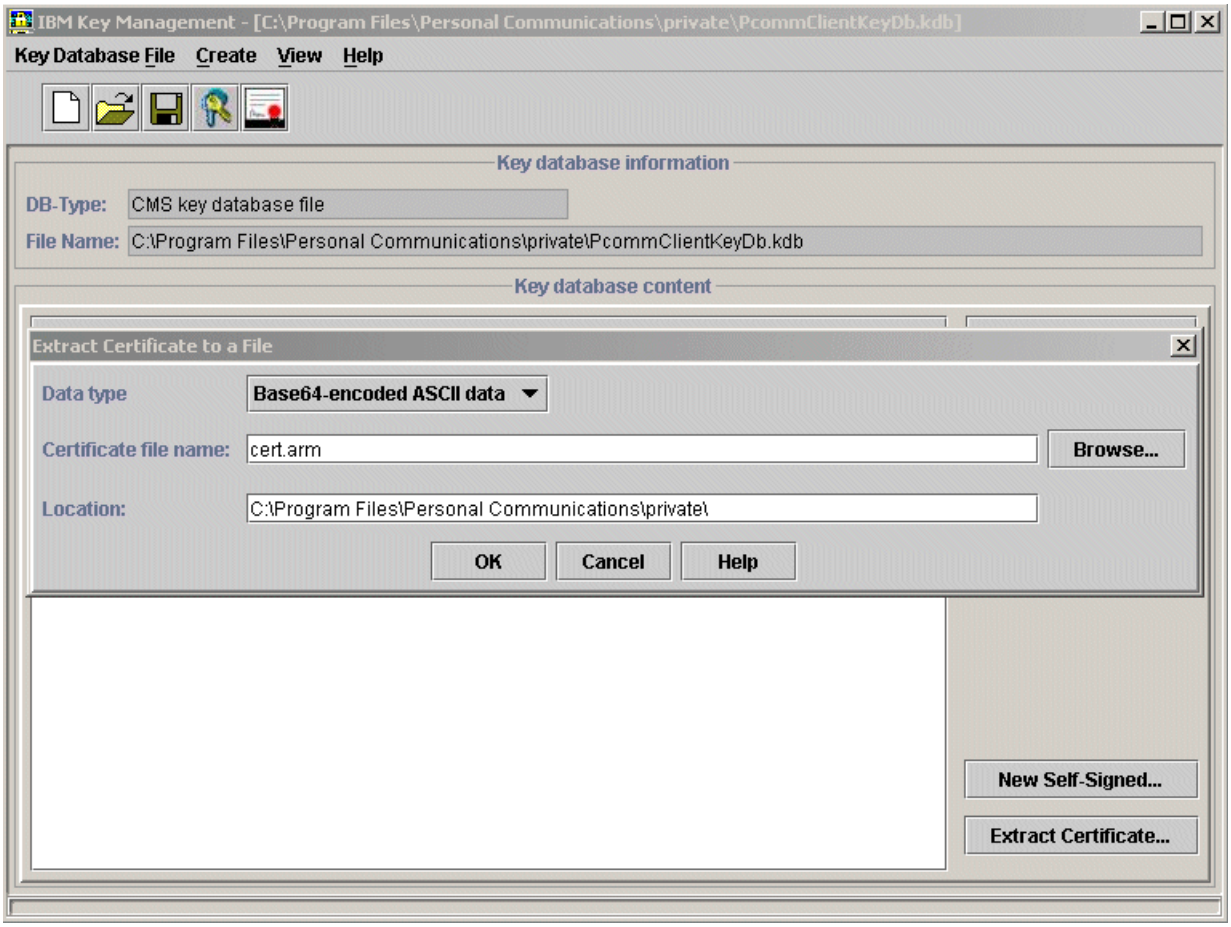


Figure 3-21 Exporting a client certificate to a file

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 69 (`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 that 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 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.

```
//CS081   JOB 'IMPORT SERVER CERT','CS08',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(CS08) ADD('CS08.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, to the server's RACF key database. In the RACDCERT CONNECT statement, the key ring name is what 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 59 for an illustration of the RACDCERT ADDRING command.

```
//CS081   JOB 'CONNECT SERVER CERT','CS08',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(CS08)              -
                           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 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 52, 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.

Tip: 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.

To use the **gskkyman** command to set up a self-signed certificate for server authentication only, follow these steps:

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 on page 74, 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 typing zero (0) and then pressing Enter to return to the Key Management Menu.

Figure 3-26 on page 75 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 on page 75, at **2** we can see that option 6 allows us to export this certificate to a file.

Tip: Do not use option 7 - the private key is only required for the endpoint that wants 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 appears. 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.

Tip: 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-----
MIID6gYJKoZIhvcNAQcCoIID2zCCA9cCAQExADALBgkqhkiG9wOBBwGgggO/MIID
uzCCAqOgAwIBAgIIRvv/9wAEmyEwDQYJKoZIhvcNAQEFBQAwczELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAk5DMRUwEwYDVQQHEwxQb3VnaGtlZXBzaWUxDDAKBgNVBAoT
AOlCTTEbMBkGA1UECxMSSVRTTyBUTjMyNzAgU2VydmVyMRUwEwYDVQQDEwxJVFNP
LklCTS5DTOOwHhcNMDcwOTI3MTkwOTQzWhcNMTcwOTIOMTkwOTQzWjBzMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCTkMxFTATBgNVBAcTDFBvdWdoa2VlcHNpZTEMMAoG
A1UEChMDSUJNMRswGQYDVQQLExJJVFNPIFROMzI3MCBTZXJ2ZXIxFTATBgNVBAMT
DElUUO8uSUJNLkNPTTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKYM
AfAaVB/eAG3Z7iw8gEnn8epssKm5PlibWVXvYm81WzJYnbbI+Gv+bU52xBMVxIrb
N3IQOBcAzayFLKLugfhWczWzcsVOX2Ea62q4w3kgdhvjhF+vHq69NhBKdphjypCU
bJDMwKimU+CVIrMpco/FJ7IjmA5h971K+OmlxxGnMEu/EsiOTwV/cG1Oy4ylOHU4
CU39zrrvQD3seDM/SNgkZd9edAofqD+t+bJ2ZdvUSOwPSxlFmKyqqJbZkt6rKQdq
OmSJRLgJSVsRQjX9OA5Ah+7toPAgv9pUGNA+EwUwMEjnflCadhc4caCbwbuKMryD
6NmbD2/yeDpsYd+hsgsCAwEAAaNTMFEwHQYDVROOBBYEFEcKBLLH71Slb+dHGgOb
OgSChyyJMB8GA1UdIwQYMBaAFEcKBLLH71Slb+dHGgOb0gSChyyJMA8GA1UdEwEB
/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 can 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 is corrupted somehow.

3. Use FTP for the file exported in step 2 on page 74 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 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 62.

   Now, import the certificate by clicking **Add**. When the certificate is added, the window in Figure 3-28 on page 78 (the detail display from the certificate) displays and shows the key size of 1024 (set by **gskkyman** when creating the certificate in Figure 3-25 on page 74), the certificate version, and the `Issued To` name are the same as the `Issued By` name (this is a self-signed certificate).

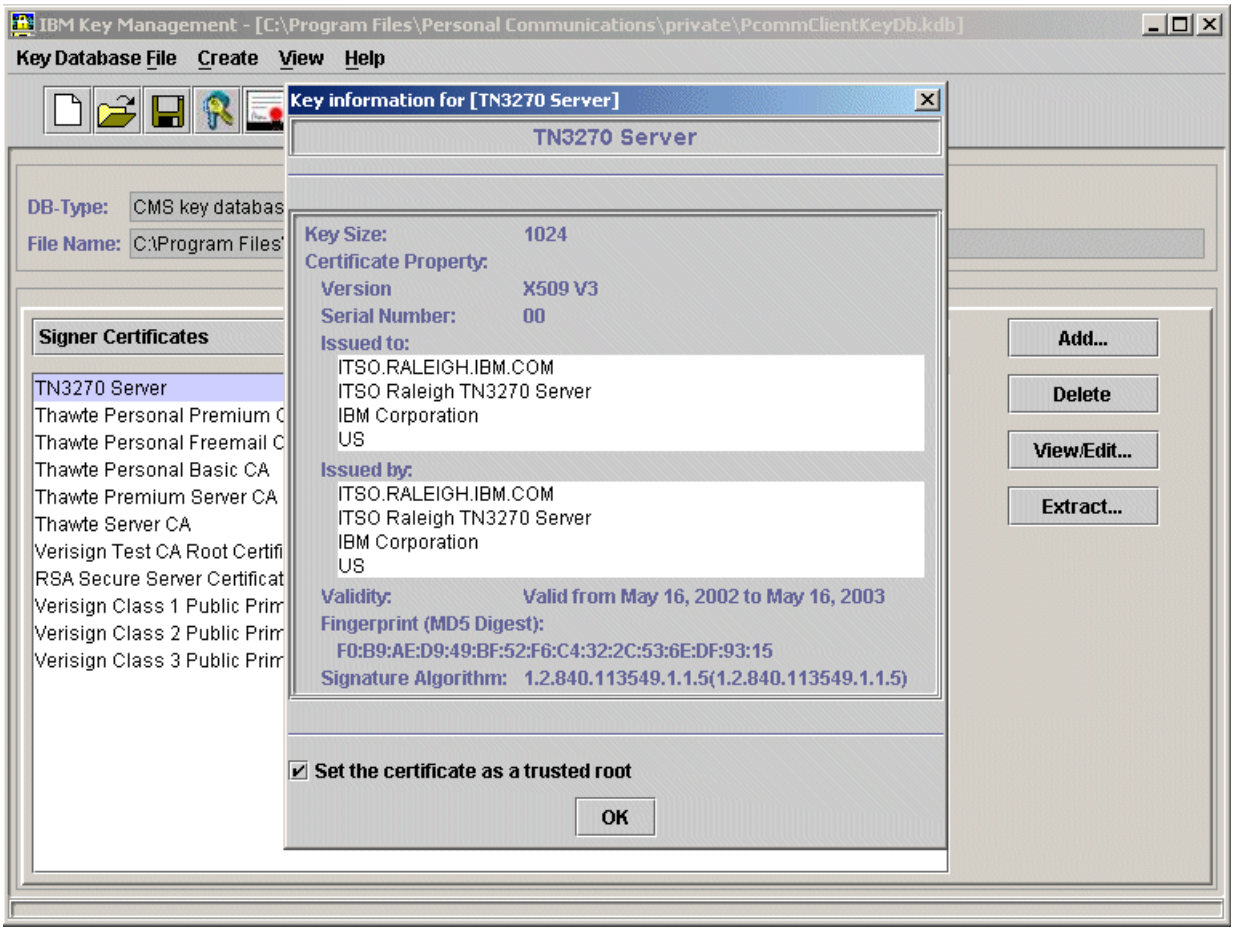


Figure 3-28 Personal Communications client: Display of imported server certificate

Now the client is 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 to those in the RACDCERT environment, except that the certificates are stored in a z/OS UNIX key database rather than the RACF database.

The implementation steps are as follows:

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.

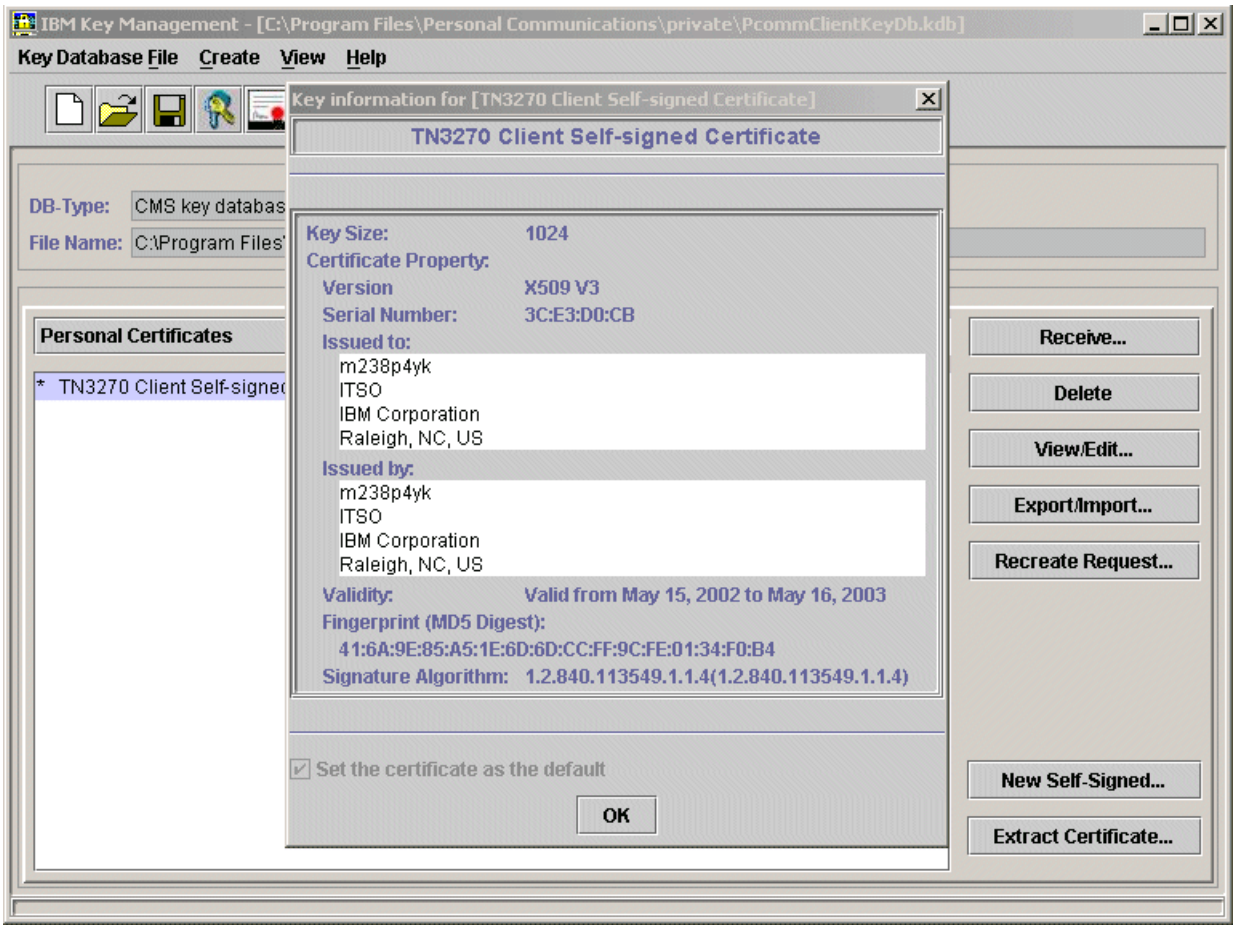


Figure 3-29 Personal Communications client: Newly added self-signed client certificate

2. Export the client certificate to a file.

   We used Base64 ASCII format. Click the **Extract Certificate** button to write a certificate to a file. The window that is presented is shown in Figure 3-30. Note that the `Data Type` field specifies Base64-encoded ASCII, and that the certificate will be written to the `cert.arm` file.

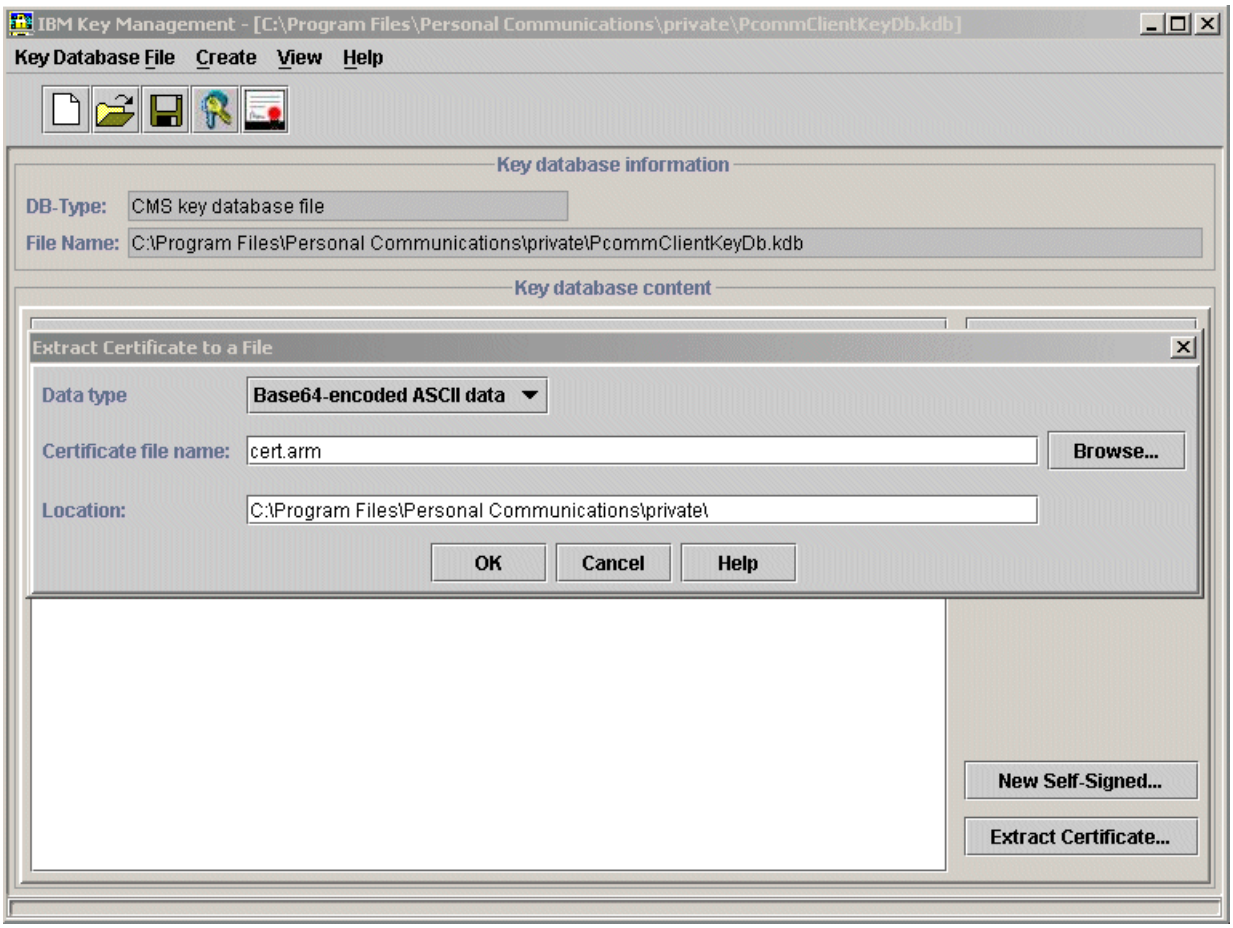


Figure 3-30 Exporting a client certificate to a file

3. Use FTP on 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 (`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, 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`**.

   See Figure 3-24 on page 73 for instructions about opening an existing database using **`gskkyman`**.

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

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`, and was placed there in the FTP transfer in step 3 on page 80.

```
 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

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 65, 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 using 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 83 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, 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 Personal Communications (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 62. Import the certificate by clicking the **Add** button.

   When the certificate is added, the window shown in Figure 3-36 on page 85 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).

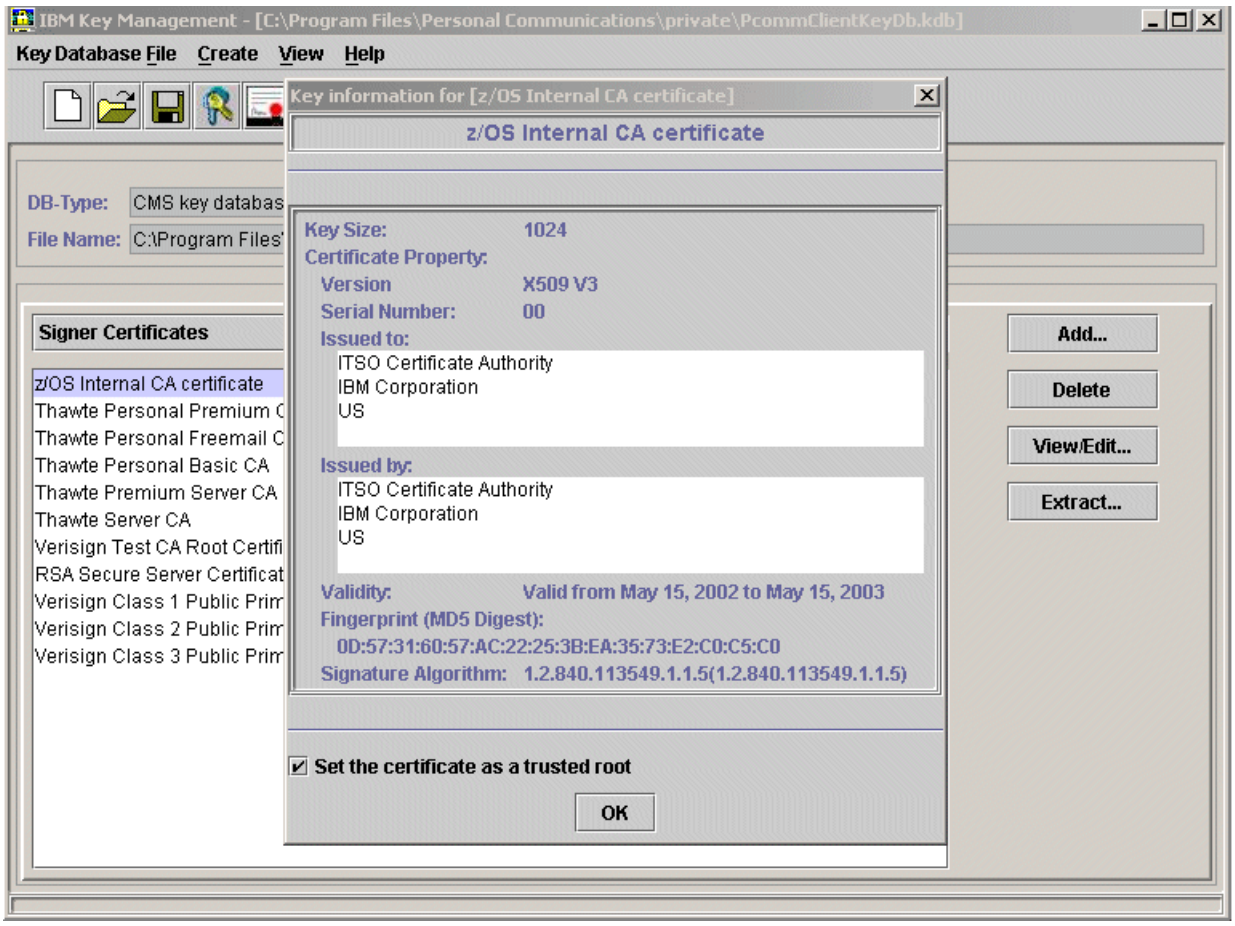


Figure 3-36 The IBM Personal Communications client: display of imported server certificate

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 Personal Communications, select **Communication** → **Security Information**. You should see something similar to what is shown in Figure 3-15 on page 64.

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.

Tip: 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 certain 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)
          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 email 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 example, `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 database at this time. You might need to go to the CA vendor's website to pick up any intermediates, or they might 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 on page 86. 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 only contain 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))
   ```

   Tip: 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 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 can 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 52, 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`.

To create a key database in z/OS UNIX and to create certificate requests for external CAs for the IBM HTTP Server for z/OS, follow these steps:

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

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

```
          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

In this figure, the numbers correspond to the following actions:

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 88, 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
DQEBBAUAAOEAc7OFskVCHrzZXkyoIa6NnDdrtt6CHhMKLJKtIitStFPXZVIMQxPK
1ER2vdsdzpQtIqgTromX2Jf4l4qm47gcWA==
-----END NEW CERTIFICATE REQUEST----
```

Figure 3-38 The content of the certificate request file

As shown in Figure 3-39, 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, and Equifax.

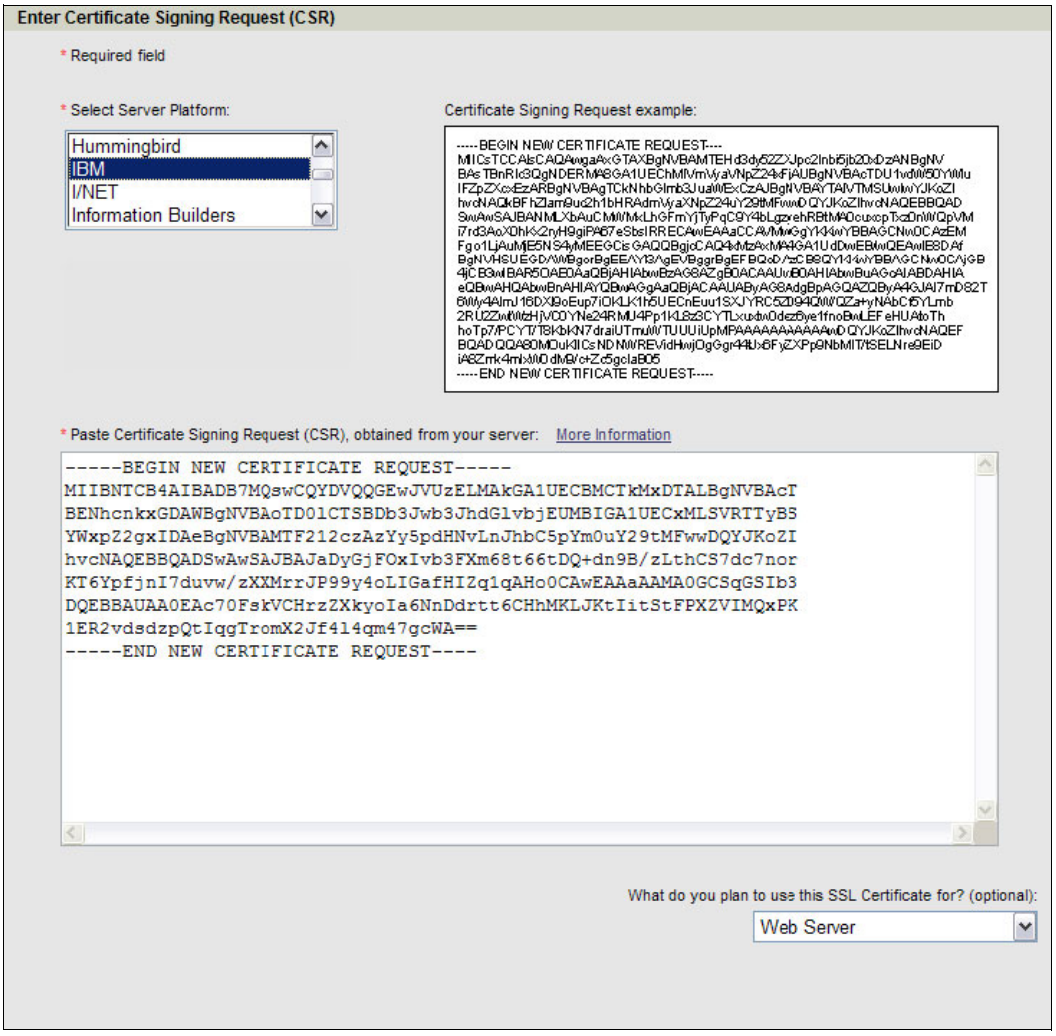


Figure 3-39 Certificate request submit form at VeriSign website

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 through email, 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, 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.

   Tip: If your CA vendor requires an intermediate certificate, you should obtain one. The intermediate might have been delivered through an email, or you might need to go to the vendor's website 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 (see Figure 3-24 on page 73 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 92.

You should set this certificate as the default. See Figure 3-26 on page 75 for an example of setting a default certificate.

Tip: 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.

Use the following basic procedure:

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 Personal Communications window to request a certificate.

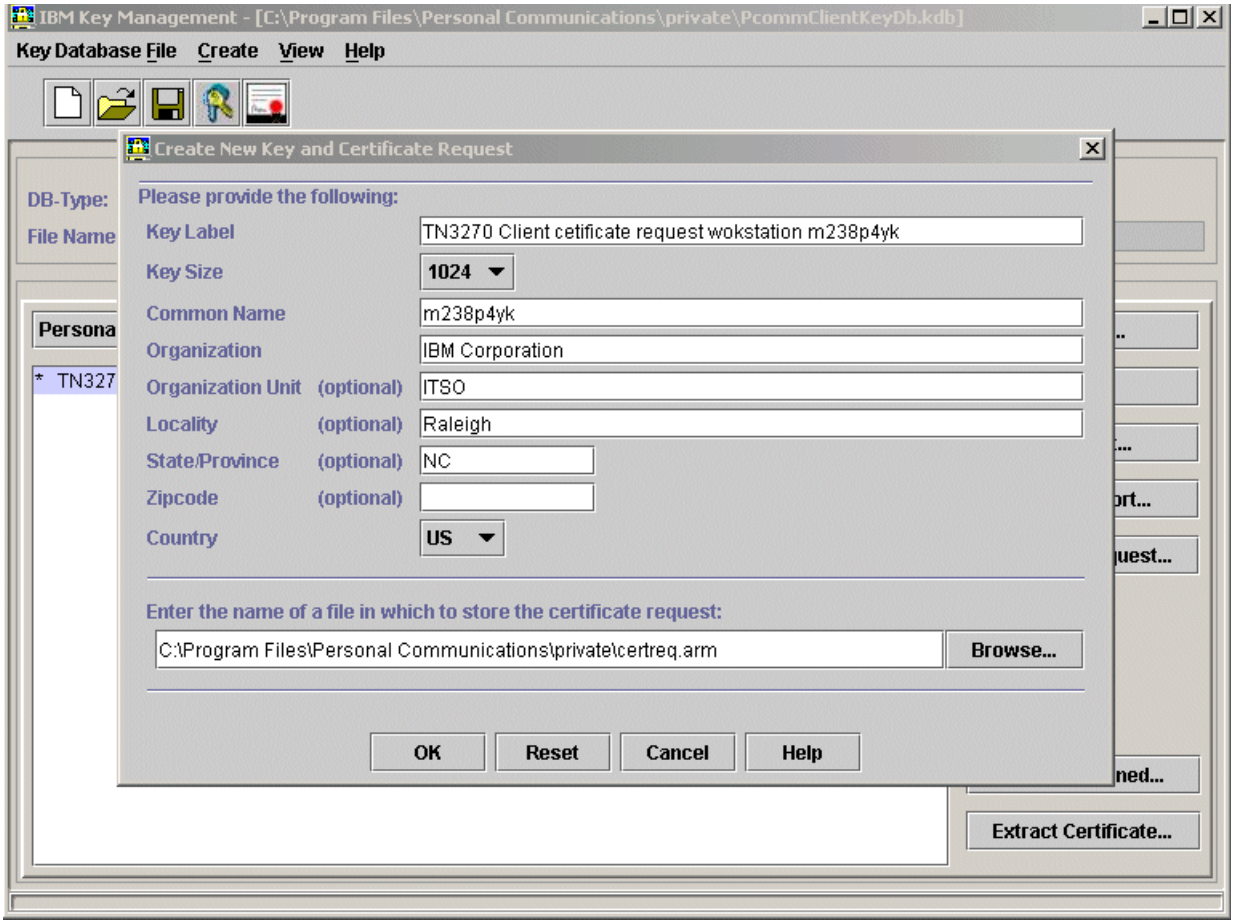


Figure 3-42 Using a client to request a certificate

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.

   Email the certificate request output from the client, either by using cut and paste, or as an attachment. The CA might 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 might have to go to a secure website to download your client certificate and key, or it might be sent to you in a secure email. Whatever method is chosen, you must end up with a file, probably in PKCS #12 format, that 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 similar to the example illustrated in Figure 3-13 on page 62.

   Just below the heading `Key database content`, there is a drop-down box that contains 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 **Import**, select the certificate file that you exported in step 1, and click **OK**. You are asked for the password that was provided by your CA.

Figure 3-44 shows the window after you have imported the key/certificate into the key database. The file received from the CA can contain the client certificate and all intermediate certificates, up to and including the root signer's certificate. The import function might 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.

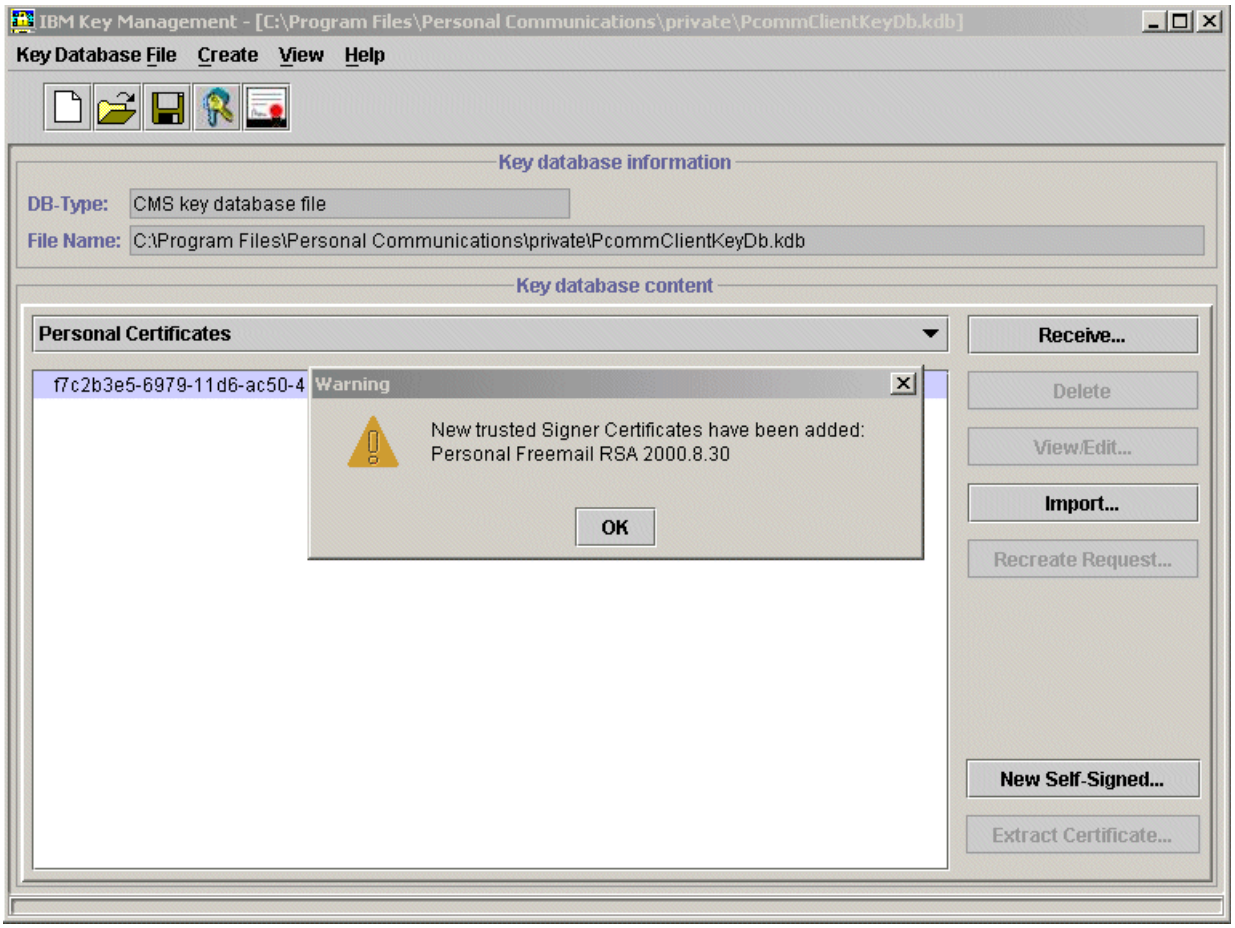


Figure 3-44 Personal Communications certificate window after importing a certificate and key

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, two possible variations of this step exist:

   – 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.

   – 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 68.

External CA-signed client authentication, gskkyman procedure

The basic procedure to follow for **gskkyman** is the same as that described in "External CA-signed client authentication, RACDCERT procedure" on page 94:

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.

   – 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 81.

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 4, “Policy agent” on page 103) 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 Communications 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 Security (IPSec) and filtering
- Defense Manager Daemon (DMD))
- Network Address Translation (NAT) traversal support
- Application Transparent TLS (AT-TLS)
- Intrusion detection services (IDS)
- Policy Based Routing (PBR)

Configuration files created by earlier GUIs can be imported into the IBM Configuration Assistant GUI, which makes migration easy.

The IBM 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 in Figure 3-45.

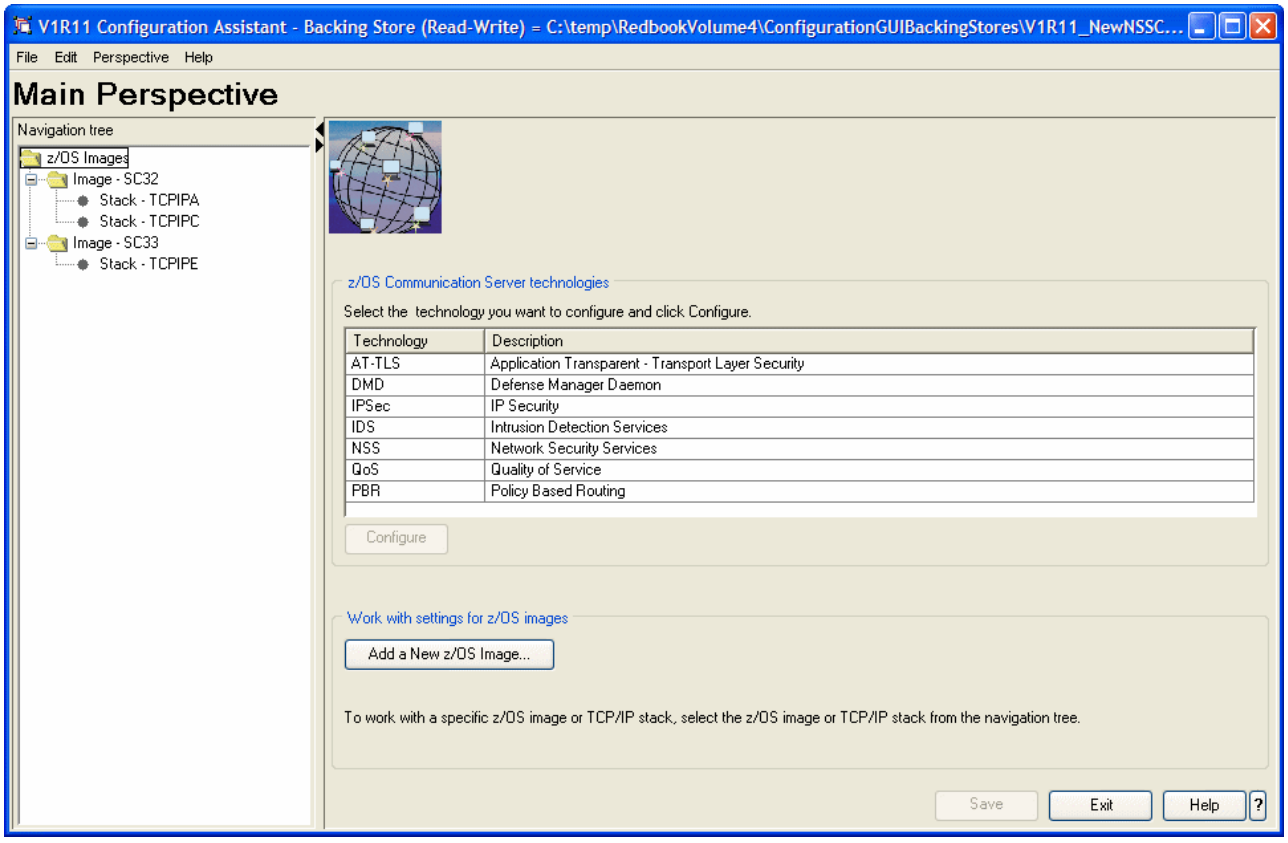


Figure 3-45 IBM Configuration Assistant GUI

To provide this view, the IBM 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 IBM 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 website listed at 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

Centralizing security services

As dependence on digital data at rest and data in transit grows, it becomes increasingly important to secure that data from the dangers that threaten it. This means being able to respond in the affirmative to the following questions:

- Are the users of the data being authenticated?
- Are only the users with a right to know being granted access to the data?
- Is the data being kept away from unauthorized users, either through access authorization techniques or through privacy protection techniques like encryption or masking?
- Are intruders being prevented from altering the data and corrupting its integrity?
- Are the systems at which the data is processed or stored protected from denial of service so that authorized access to the data is not impinged upon?

Security breaches of any kind on data, on the systems that process the data, and on the devices and networks that store and transport the data impose increasing risk on users and businesses as this dependence on the data increases. The impacts of these risks manifest themselves in damage to economic outcomes and reputations:

- Decline in value of savings, profits, investments
- Rise in fines, lawsuits, bankruptcies
- Lowering of credit scores, trust, consumer confidence, market share

Historically, security has been addressed on a platform-by-platform basis, with each application and platform implementing its own brand of authentication, access control, confidentiality, data integrity, non-repudiation, and governance. However, over time IT environments have grown in complexity, with the costs to maintain them soaring. A large contributor to these growing financial outlays has been the security management arena.

Methods and technologies have emerged to streamline security management costs, including the introduction of the following items:

- Model definitions to simplify the scaling of security definitions to fit a growing environment and user base
- GUIs to reduce the complexity of creating the definitions to provide security
- Shared security databases, as with RACF, to reduce the amount of administrative activity necessary to control definitions on multiple platforms
- LDAP to centralize the repository of policy definitions, including security policies
- The Public Key Infrastructure (PKI) from IBM to manage security certificates within the corporation instead of purchasing the services from entities outside the corporate entity
- Kerberos to centralize the issuing of passtickets for access to *kerberized* applications on any platform

Of course, many other examples could be cited. Although many architects view centralization of security as a means to lower overall IT costs, in fact centralization should also be viewed as a security mechanism itself. Note how shared security databases, LDAP, PKI, and even

Kerberos are examples of centralized security technologies. With centralization of security we can do the following tasks:

- Reduce the number of personnel requiring specialized, platform-specific security skills scattered throughout the network.
- Reduce the administrative overhead of managing separate sets of definitions.
- Store and manage security definitions at a single node that lies in an extremely secure part of the network. Such centralization can locate the definitions in a secure zone, thus protecting the security data itself from intentional and accidental violations.

More about this part of the book

In this part of the book, we introduce technologies that rely on centralization of security for multiple platforms and, in some cases, disparate platforms. The following chapters illustrate the implementation and benefits of centralized security technologies in z/OS Communications Server:

- Chapter 5, “Central Policy Server” on page 161
- Chapter 9, “Network Security Services for IPSec clients” on page 325
- Chapter 10, “Network Security Services for WebSphere DataPower appliances” on page 429

Other topics in this part include the following chapters:

- Chapter 4, “Policy agent” on page 103
- Chapter 6, “Quality of service” on page 199
- Chapter 7, “IP filtering” on page 217
- Chapter 8, “IP Security” on page 245
- Chapter 11, “Network Address Translation traversal support” on page 535
- Chapter 12, “Application Transparent Transport Layer Security” on page 551
- Chapter 13, “Intrusion detection services” on page 583
- Chapter 14, “IP defensive filtering” on page 617
- Chapter 15, “Policy-based routing” on page 641

4

Policy agent

The policy agent runs in a z/OS address space and provides various services in managing dependent components of the z/OS Communications Server policy infrastructure. Depending on your configuration options, the policy agent can act in the following roles:

- When acting as a *policy decision point* (PDP), the policy agent executes on a single system and installs policies on one or more z/OS Communications Server stacks on a z/OS image.
- When acting as a *policy client*, the policy agent retrieves remote policies from the policy server.
- When configured as a *central policy server*, the policy agent provides PDP services to one or more remote policy clients.

We discuss the following topics in this chapter.

Section	Topics
4.1, “Policy agent description” on page 104	Using a policy agent as a central repository for policies and basic concepts of the policy agent
4.2, “Implementing PAGENT on z/OS” on page 109	The basic installation steps for the policy agent
4.3, “Setting up the Traffic Regulation Management daemon” on page 125	Setting up Traffic Regulation Management daemon (TRMD) to handle syslogd event recording for intrusion detection services (IDS) and IPSec services
4.4, “Configuration Assistant for z/OS Communications Server” on page 127	How to use z/OSMF Configuration Assistant to generate and install the policy agent configuration files Create and work with reusable IPSec connectivity rules.
4.6, “Backup and migration considerations” on page 151	Finer points of migrating from one release of policy agent to a newer release and of merging separate policy files
4.7, “Additional information” on page 159	Resources for additional information

4.1 Policy agent description

As illustrated in Figure 4-1, the z/OS Communications Server policy agent (PAGENT) implements policy-based networking for the z/OS Communications Server environment.

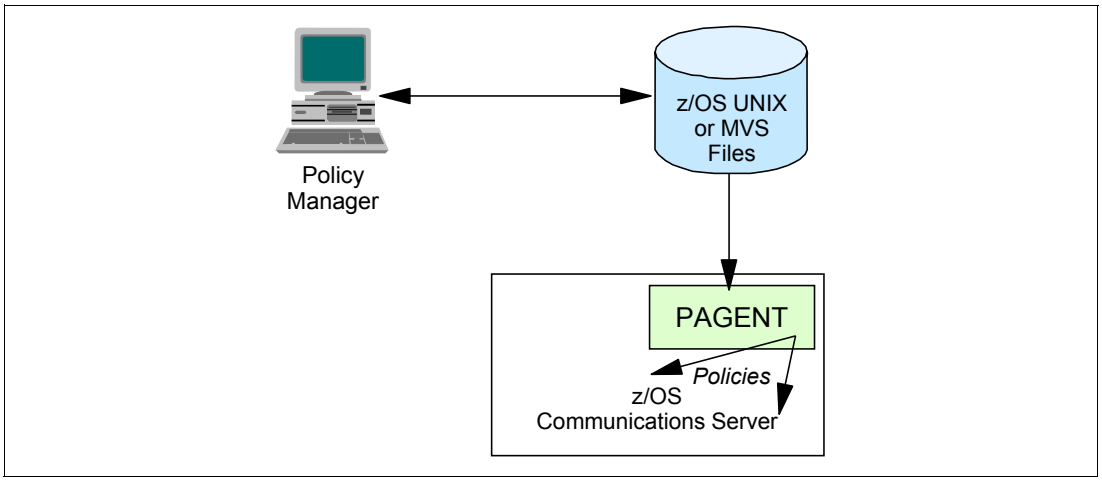


Figure 4-1 Policy-based networking and PAGENT

Policy Agent provides a number of services:

- Process policy definitions, install and maintain policies in TCP/IP components. This is described in “Defining the TcpImage statements” on page 114.
- Import policy definitions through the policy file import service. This is described in “Importing the policy file to Configuration Assistant” on page 156.
- Import TCP/IP profile information through the discovery of TCP/IP profile function. This is described in “Additional information” on page 159.
- Automatically monitor and manage certain policy infrastructure applications. This is described in “Policy infrastructure management” on page 119

Policy definitions are usually contained in local configuration flat files. Alternatively, some policy types can be stored in an LDAP server. The policies are used to control network security, traffic prioritization, bandwidth management, network behavior, routing, 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. Most policy types are then acted on by an TCP/IP stack. IPSec dynamic VPNs are implemented by the Internet Key Exchange daemon (IKED).

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. Therefore, use flat text files for building networking policies.

As a result, in our test environment, we exclusively used the z/OS Management Facility (z/OSMF) Configuration Assistant to create flat text configuration files.

We discuss the following policy disciplines and security features in subsequent chapters:

- Quality of service (QoS)
- Intrusion detection system (IDS)
- Defense Manager Daemon (DMD)
- IP security (IPSec) virtual private network (VPN)
- IP filtering
- Application Transparent Transport Layer Security (AT-TLS)
- Policy-based routing
- Network Security Services

4.1.1 Basic concepts

Architecturally, policy-based networking typically involves the following components:

- Policy management service

  This is a GUI for specifying, editing, and administering policies, such as the z/OSMF Configuration Assistant.

- Policy repository

  This is a place to store and retrieve policy information, such as a configuration file.

- Policy decision point (PDP)

  This resource manager or policy server is responsible for handling events, making decisions based on those events, and updating the Policy Enforcement Point configurations appropriately. PAGENT is our PDP, as shown in Figure 4-2.

- 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 4-2, the TCP/IP stack serves as our PEP.

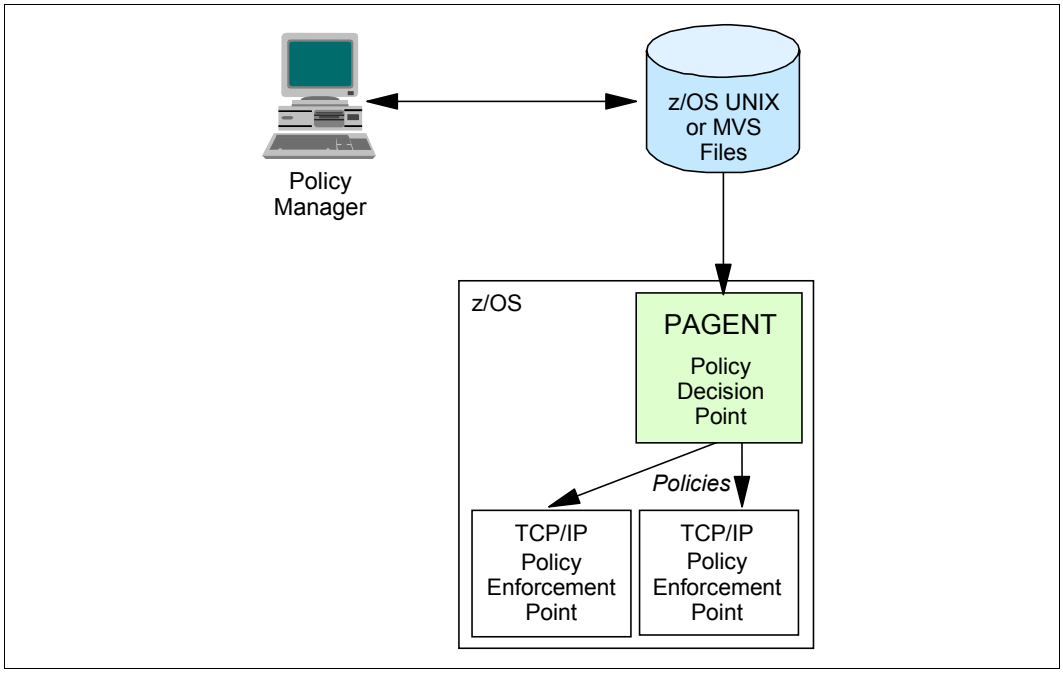


Figure 4-2 Policy system model

4.1.2 Where and how to define policies

Policies are defined in the policy agent configuration file as shown in Figure 4-3. Keep policy names unique because policy objects with duplicate names run the risk of being deleted by PAGENT.

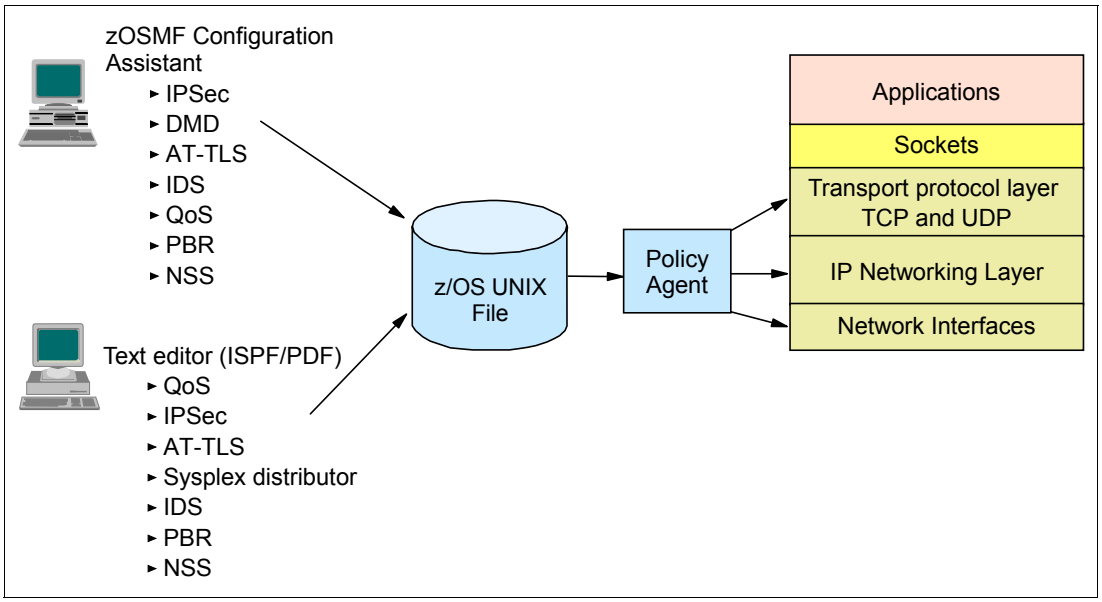


Figure 4-3 Configuring policy agent

PAGENT, in cooperation with other z/OS daemons such as TRMD, IKED, NSSD and DMD, supports the following policy and security technologies, described in subsequent chapters:

- Chapter 6, “Quality of service” on page 199
- Chapter 7, “IP filtering” on page 217
- Chapter 8, “IP Security” on page 245
- Chapter 9, “Network Security Services for IPSec clients” on page 325
- Chapter 10, “Network Security Services for WebSphere DataPower appliances” on page 429
- Chapter 11, “Network Address Translation traversal support” on page 535
- Chapter 12, “Application Transparent Transport Layer Security” on page 551
- Chapter 13, “Intrusion detection services” on page 583
- Chapter 14, “IP defensive filtering” on page 617
- Chapter 15, “Policy-based routing” on page 641

Figure 4-4 shows the components involved in the networking policies of z/OS Communications Server.

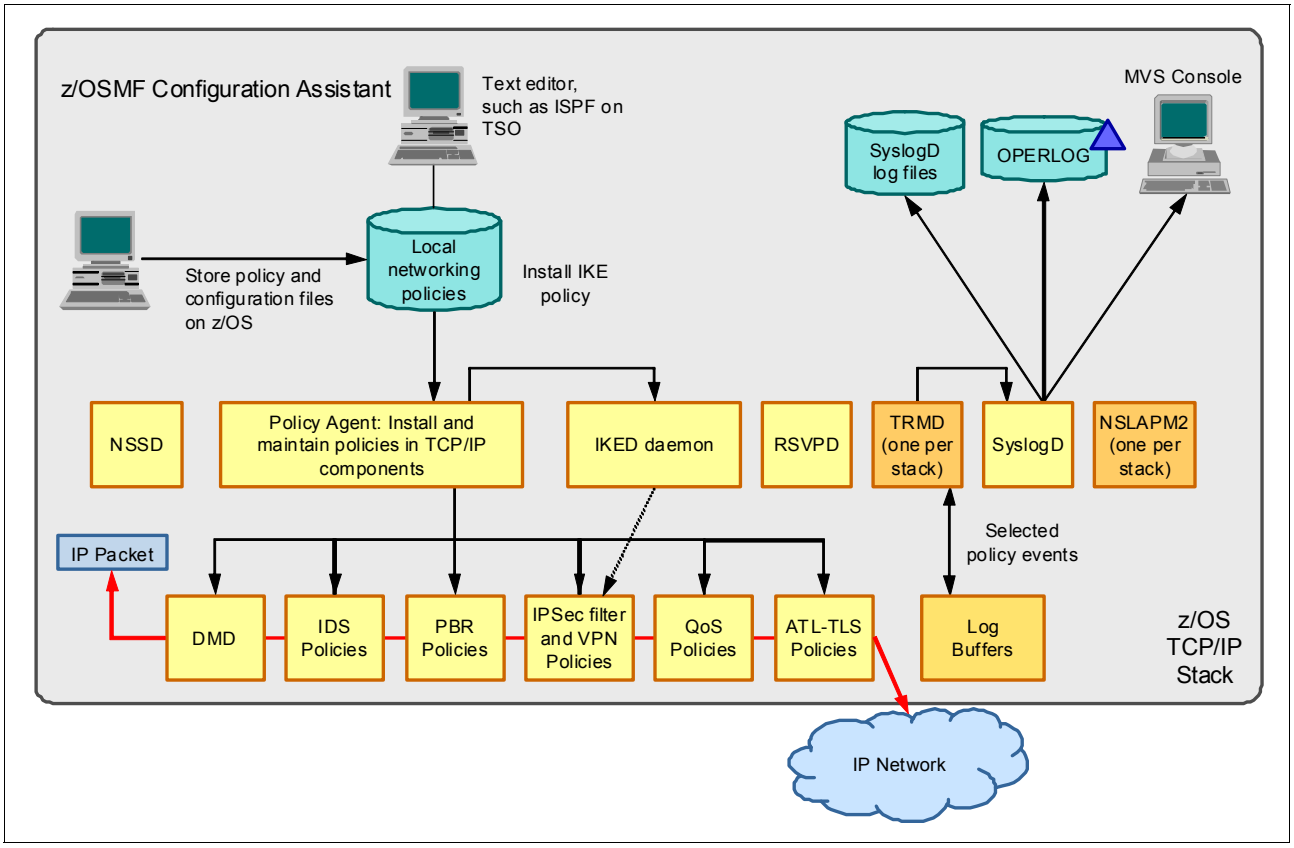


Figure 4-4 Full policy agent implementation on z/OS

Table 4-1 shows the policy types that are used by the TCP/IP stack. Certain 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 4-1 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	
DMD configuration file	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, use the Configuration Assistant. It can build the configuration flat files for QoS, IDS, AT-TLS, IPSec, and policy-based routing (PBR) policies. Some configuration files needed for the PAGENT implementation, however, still must be built manually:

- The policy agentmain configuration files
- The SyslogD files
- The RSVPD configuration files

The DMD, IKED, and NSSD configuration files, can be defined using either the manual edit process or the z/OSMF Configuration Assistant

Tip: Use the z/OSMF Configuration Assistant when possible to avoid syntax errors that can occur with manual editing. There is also a “health check” feature in the GUI that can detect many of the logic errors that you might introduce when building policies

For more information about the z/OSMF Configuration Assistant, see 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

4.2 Implementing PAGENT on z/OS

The PAGENT runs as a UNIX process. As such, it can be started either from the UNIX System Services shell or as a started task. In our environment, we used a started task procedure to start policy agent.

4.2.1 Starting PAGENT as started task

You can find the sample started task procedure for PAGENT in TCPIP.SEZAINST(EZAPAGSP). We used the PAGENT started task procedure shown in Example 4-1 on our system.

Example 4-1 PAGENT started task procedure

```
//PAGENT   PROC
//
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//* For information on the above environment variables, refer to the
//* IP CONFIGURATION GUIDE.  Other environment variables can also be
//* specified via STDENV.
//*
//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)
```

In this example, the number corresponds to the following information:

1. 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 4-2 on page 110.

Tip: 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 4-2 STDENV data set contents

```
LIBPATH=/usr/lib:.
PAGENT_CONFIG_FILE=/SC30/etc/pagent.sc30.conf 1
PAGENT_LOG_FILE=SYSLOGD                       2
TZ=EST5EDT
```

We configured the following environment variables for the policy agent:

1. PAGENT_CONFIG_FILE specifies the specific PAGENT configuration file to use.
2. PAGENT_LOG_FILE specifies the log file name used by PAGENT. In our case, we use SYSLOGD.

To configure PAGENT to use SYSLOGD to log messages, you can define PAGENT_LOG_FILE=SYSLOGD or code the -l/L SYSLOGD start option. In this case, PAGENT_LOG_FILE_CONTROL has no meaning. The -l/L start option overrides the PAGENT_LOG_FILE environment variable.

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.

Tip: Most z/OS UNIX applications that start as MVS started tasks cannot use environment variables that are 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 PAGENT 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 might already have been done, and therefore you might not have to issue commands to define the STARTED class RACLIST and GENERIC. We simply mention them 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 4-3 shows the RACF commands for this.

Example 4-3 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.

In our environment. the user ID is defined with UID=0. As we automatically monitor our policy infrastructure applications, superuser authority is required in this case. For more information about automatically monitoring policy infrastructure applications, see 4.2.8, “Policy infrastructure management” on page 119

However, 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 to use the SHARED parameter in the definition.

If you chose not to use UID(0), you must assign PAGENT appropriate file access authority for necessary files. See the *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for more information.

A home directory is also assigned to this user ID. Example 4-4 shows the command that we used.

Example 4-4 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 4-5 shows the command that we used.

Example 4-5 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, 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 4-6 shows the commands that we used.

Example 4-6 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 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 4-7 shows the commands that we used.

Example 4-7 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 AT-TLS, z/OS does not allow socket-based applications to start before PAGENT is running to make sure that all the security policies are enforced. However, certain essential applications need to start before PAGENT. To allow them, 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

Example 4-8 shows the RACF commands that we used for this scenario.

Example 4-8 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
```

4.2.2 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 (&).

See *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782 to resolve any EZZ errors encountered at PAGENT startup time.

4.2.3 Stopping PAGENT

You can stop PAGENT 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 4-9 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 4-9 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.

4.2.4 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
```

4.2.5 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 can 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.

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. You can define a secondary configuration file for each stack, or a set of stacks can share configuration information in the main configuration file. You can also use a combination of these techniques.

To install separate sets of policies to separate stacks, configure each image with a different secondary configuration file. In this case, each image can be configured with a separate policy refresh interval, if required. The refresh interval used for the main configuration file will be the smallest of the values specified for the stacks.

Figure 4-5 shows that the PAGENT configuration file identifies, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed and the separate configuration files that are used by each.

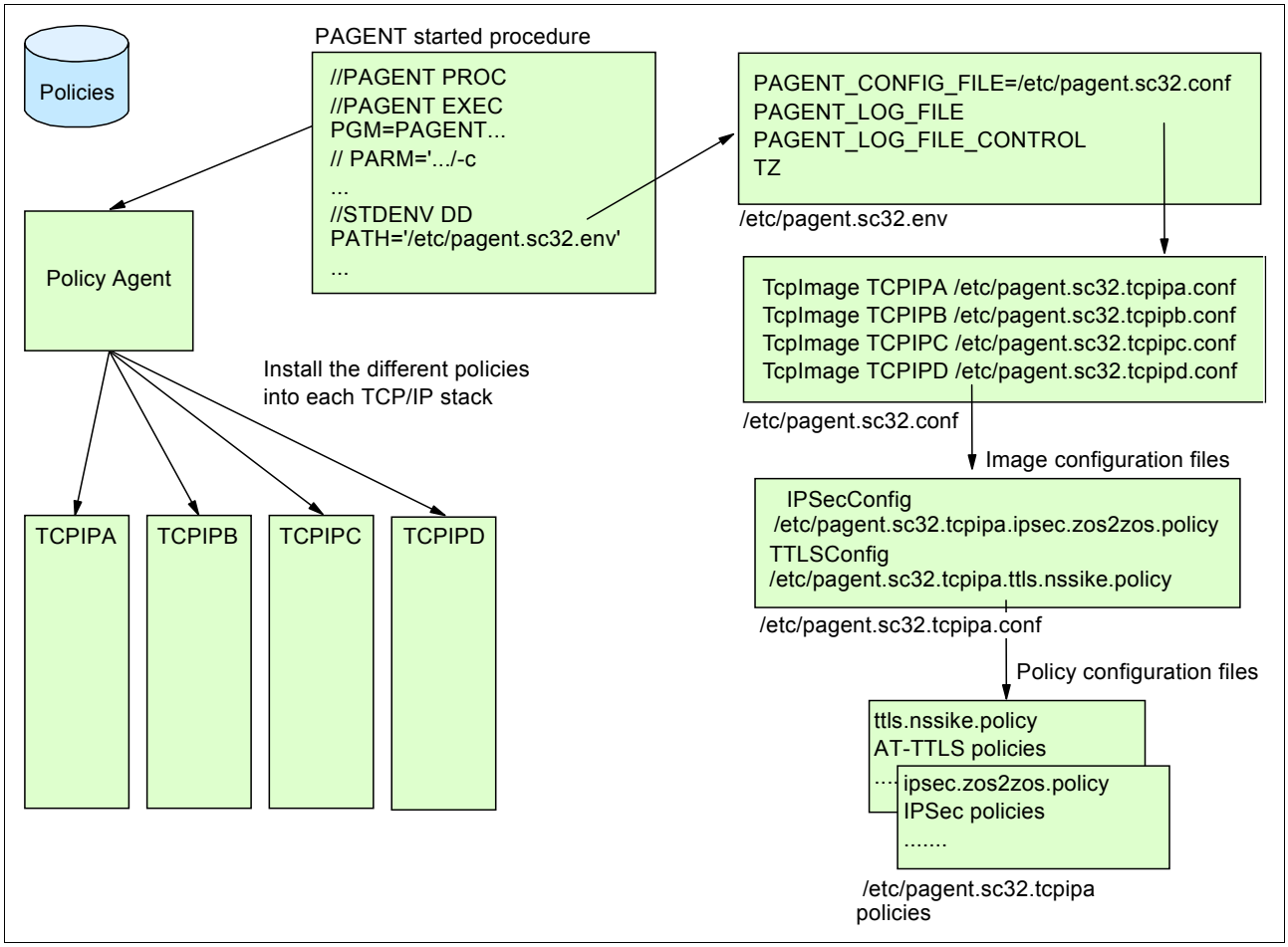


Figure 4-5 Multiple stacks, multiple policy definitions

Tip: 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. Separate refresh intervals can also be configured for each image, but that might not be useful in this case.

Figure 4-6 shows that the PAGENT configuration file identifies, through TcpImage statements, the names of the TCP/IP stacks on which policies are to be installed. However, in this case, the same policies are installed into each stack.

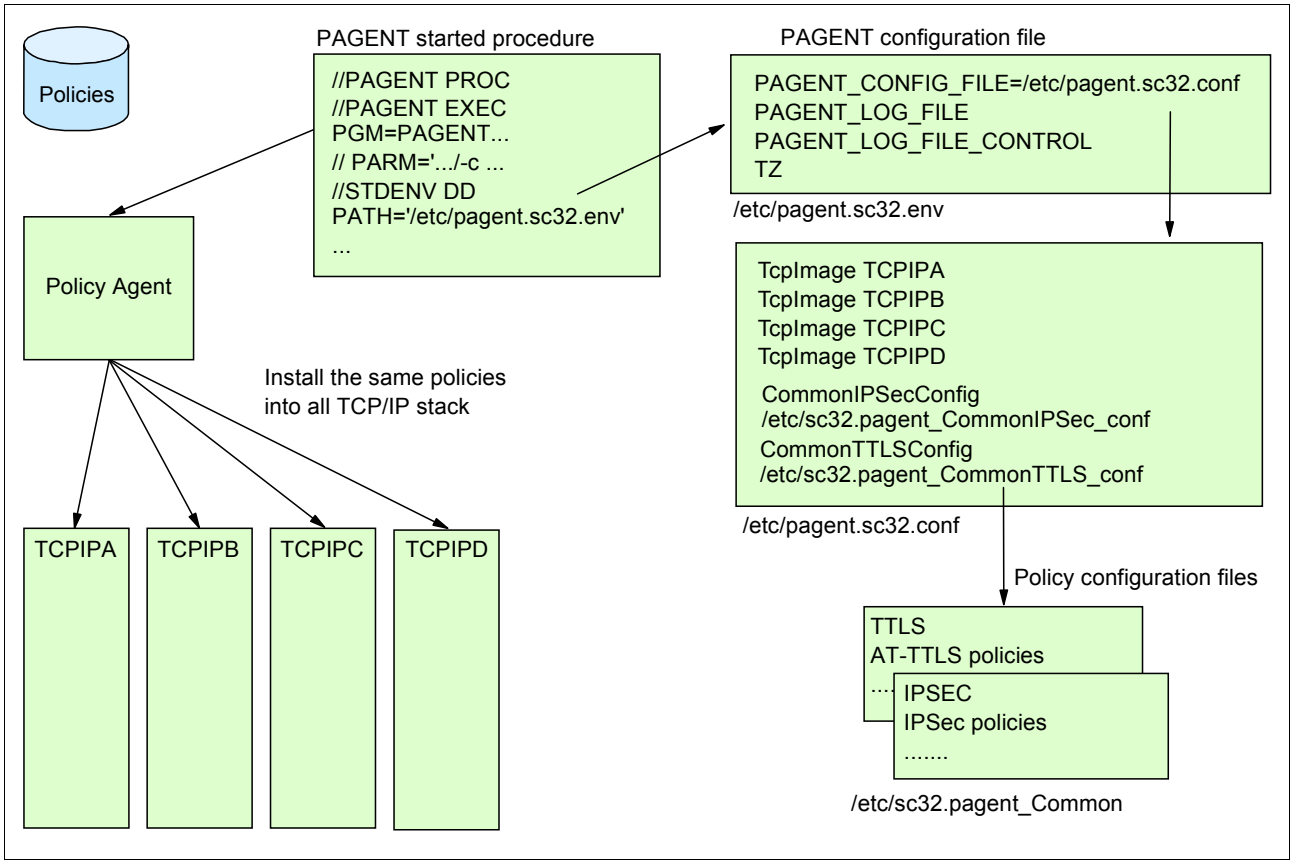


Figure 4-6 Multiple stacks, single policy definition

It is possible that TCP/IP stacks that are configured to the policy agent are not started or even defined. The policy agent fails when trying to connect to those stacks and logs the appropriate error messages.

The policy agent does not end when any (or all) stacks end. When the stacks are restarted, active policies are reinstalled automatically. When the policy agent is shut down normally (that is, using the 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.sc32.conf` to the TCPIPA TCP/IP image after flushing the existing policy control data:

```
TcpImage TCPIPA /etc/pagent.sc32.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.

Tip: 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 the 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.

4.2.6 Coding policy definitions in a configuration file

The configuration shown in Example 4-10 is based on the “Multiple stacks, multiple policy definitions” scenario shown in Figure 4-5 on page 115. In this scenario, the policy definitions are configured in the PAGENT configuration file. We use a two-level PAGENT configuration file to define the policy in a multiple IP stacks environment.

Example 4-10 PAGENT configuration file

```
# LogLevel Statement
Loglevel 255 1
# TcpImage Statement 2

TcpImage TCPIPA /etc/pagent.sc32.tcpipa.conf  FLUSH PURGE 600
TcpImage TCPIPB /etc/pagent.sc32.tcpipb.conf  FLUSH PURGE 600
TcpImage TCPIPC /etc/pagent.sc32.tcpipc.conf  FLUSH PURGE 600
TcpImage TCPIPD /etc/pagent.sc32.tcpipd.conf  FLUSH PURGE 600
```

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.

> **Tip:** 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 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).

> **Tip:** 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 you specify in the PAGENT_LOG_FILE environment variable and which defaults to `/tmp/pagent.log` file. The log file is created when the policy agent is activated, if it does not already exist. We suggest sending all messages to SYSLOGD, which can be done by defining PAGENT_LOG_FILE=SYSLOGD.

4.2.7 Refreshing policies

You can use the following commands 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 installs or removes from the stack (as appropriate) only new, changed, or deleted policies.

> **Recommendation:** Use the UPDATE command in most cases. Use the REFRESH command only if you suspect that policies are out of sync between the TCP/IP stack and policy agent.

4.2.8 Policy infrastructure management

The policy infrastructure allows you to define and manage several policy types that help control the following aspects of the system and network:

- Quality of service (QoS)
- Intrusion detection system (IDS)
- IP Security (IPSec) filters
- IP Security (IPSec) dynamic and manual VPNs
- Application Transparent TLS (AT-TLS)
- Policy-based routing (PBR)

Figure 4-7 shows the policy infrastructure components.

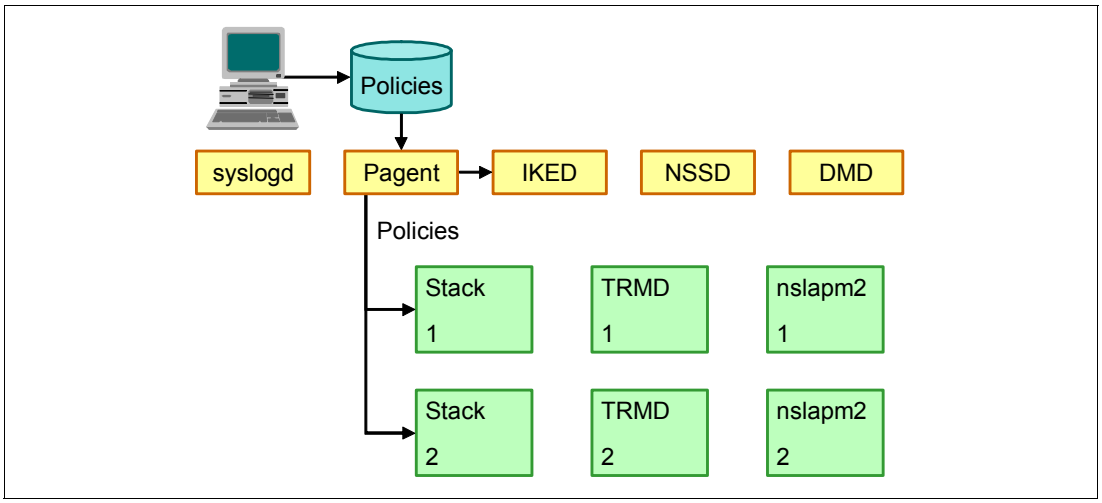


Figure 4-7 Policy infrastructure elements

The policy agent reads policy definitions that are created, either using the Configuration Assistant or manually. The policy agent installs these policies into one or more TCP/IP stacks and then provides them to other policy users, such as the IKED, as follows:

- In a single LPAR, there is one instance of the policy agent, syslogd, IKED, NSSD, and DMD. There is one instance per TCP/IP stack of TRMD and the nslapm2 subagent.
- The TCP/IP stack implements most policy types where the policies are applied as needed for inbound and outbound traffic.
- The syslog daemon (syslogd) provides a system-wide logging mechanism and, as such, is not strictly related to policy. However, several policy components use syslogd as a logging mechanism.
- The Configuration Assistant provides a user-friendly means to define policy definition and configuration files for most of the policy infrastructure components. You can also define these files manually.
- The policy agent (PAGENT) reads and parses policy definitions, installs these policies in the TCP/IP stacks, and provides IPSec VPN-related policies to the IKED.
- The Traffic Regulation Management daemon (TRMD) writes a variety of security-related messages to syslogd. You can use the **`trmdstat`** command to format these messages to produce a variety of reports.

  TRMD also captures alerts that advise you of the TCP/IP connections suffering from storage constraints. Alerts are issued when a TCP data queue enters constrained state, indicating that there is old data on the queue. Alerts are also issued when the queue later

exits the constrained state. Consult *IBM z/OS Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7998. In addition, Chapter 8 of that book provides more information about handling storage congestion on TCP data queues.

- The Internet Key Exchange daemon (IKED) provides support for dynamic and manual VPNs. It uses certain IPSec policies to communicate with its peers on other systems to establish secure tunnels to protect data being sent or received.
- The Network Security Services daemon (NSSD) provides certificate and network management services for IPSec, in addition to other services for XML appliances.
- The Defense Manager daemon allows an external security monitor function to install temporary IP filters to block a specific attack or a pattern of attacks.
- The Network SLAPM2 subagent (nslapm2) helps to monitor the effectiveness of QoS policies.

Policy infrastructure management services

Policy infrastructure management provides the following services:

- Ability to start, stop, and monitor most policy infrastructure components using the policy agent:
  - Syslog daemon (syslogd)
  - Traffic Regulation Management daemon (TRMD)
  - Internet Key Exchange daemon (IKED)
  - Network Security Services Server daemon (NSSD)
  - Defense Manager daemon (DMD)
- Ability to specify the code page for policy infrastructure components that use configuration files.
- New start option for TRMD to specify the stack to run on
- You can configure the policy agent to start, stop, and monitor a set of policy infrastructure components, namely syslogd, TRMD, IKED, NSSD, and DMD.
- You can specify the code page to be used when the policy components read their configuration files.
- Finally, you can specify the TCP/IP stack that TRMD is to run on using a start option, instead of relying only on the resolver configuration file.

Implementation and operations

To configure the policy infrastructure management, follow the steps that we describe in this section. The definitions are embedded in the policy agent configuration file.

1. Configure global application monitor parameters.

   For the application monitoring function, you first need to configure global monitoring parameters. These parameters configure the monitoring interval and retry values.

2. Configure applications to be monitored and their characteristics.

   Configure the applications to be monitored and specify parameters for each application. The policy agent uses the MVS START command to start or restart the applications. Therefore, you need to specify a cataloged procedure name. Optional parameters include the job name, start parameters, and environment variables.

3. Configure the code page for policy agent, DMD, IKED, NSSD, and SYSLOGD.

   If your configuration files do not use code page IBM-1047, you need to configure the code page that policy agent uses and the other components that use configuration files.

4. Specify the stack name using a start option for TRMD.

   If you want to specify the TRMD stack name without using the resolver configuration file, directly specify the stack name in the configuration file.

Example 4-11 shows the definitions that we added to the policy agent configuration file.

Example 4-11 Infrastructure management main configuration file

```
AutoMonitorParms
{
  MonitorInterval    10
  RetryLimitCount    5
  RetryLimitPeriod   600
}

  AutoMonitorApps
  {
    AppName            IKED
    {
      Procname         POLPROC
    }
    AppName            TRMD
    {
      TcpImageName     TCPIPA
      {
        Procname       POLPROC
        Jobname        TRMD
      }

    }
  }
```

AutoMonitorParms

The AutoMonitorParms statement in the policy agent main configuration file configures global application monitoring parameters:

- The MonitorInterval parameter specifies the number of seconds in the monitor interval, which is how often policy agent checks to see if the monitored applications are active. The default is 10 seconds.
- The RetryLimitCount and RetryLimitPeriod parameters work together. They indicate how many times within a given time period applications should be restarted. The default is five times within 600 seconds (10 minutes). If these limits are exceeded, the policy agent stops monitoring the application and does not try to restart the application. After you have resolved the application problem, you can restart the application and resume.

AutoMonitorApps

The AutoMonitorApps statement in the policy agent main configuration file configures the applications to be monitored and the parameters that are specific to those applications:

- The AppName parameter specifies each application and is repeated for the remaining applications that you want to monitor. For those applications that run a unique instance on each TCP/IP stack, specify the TcpImageName parameter for the stack name, and repeat this parameter for each stack.
- The ProcName parameter is required and specifies the name of the cataloged procedure that starts the application. The specified procedures must accept the following parameters, which are passed to the procedure by the policy agent:
  - The PROG parameter specifies the name of the executable program, and one of the supported application names is always passed by the policy agent.
  - The VARS parameter is the name of a file that contains the environment variables. The policy agent creates a temporary file and populates it with the configured environment variables. The policy agent then passes the name of the temporary file to the procedure.
  - The Jobname parameter specifies the job name that is used when the application runs. Its defaults to the AppName parameter. You need to specify a unique job name for each instance of TRMD.

You can find a detailed description of the parameters that are related to the policy infrastructure management in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Tip: When defining monitored application using policy infrastructure management, the JCL statements stored in SYS1.PROCLIB are not used. The parameters, environment variables, and stack affinity are passed by the policy agent to the procedure specified on the Procname parameter.

Starting the policy agent

Tip: Except for syslog, do not start the monitored application before starting the policy agent.

Example 4-12 shows the messages that are generated when starting the policy agent that has the infrastructure management definitions in the policy agent configuration file.

Example 4-12 Policy agent startup with infrastructure management

```
S PAGENT                                                              1
$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
S POLPROC,JOBNAME=IKED,PROG=IKED,VARS='/var/tmp/IKED_AgBGxQ'          2
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 065
        TO START POLPROC WITH JOBNAME IKED.
$HASP100 IKED     ON STCINRDR
S POLPROC,JOBNAME=TRMD,PROG=TRMD,VARS='/var/tmp/TRMD_TCPIPA_cHAfxQ',2
PARMS=' -pTCPIPA'
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 068
        TO START POLPROC WITH JOBNAME TRMD.
$HASP100 TRMD     ON STCINRDR
```

In this example, the numbers correspond to the following information:

1. The policy agent is started.
2. TRMD and IKED are started by the policy manager. Note that the procedure is POLPROC.

Automatic restart

Example 4-13 shows the z/OS log message for automatic restart of IKED. In our scenario, we cancelled the IKED job, and the PAGENT restarted it without operator intervention.

Example 4-13 IKED restart

```
P IKED
EZD0923I IKE HAS RECEIVED THE STOP COMMAND
EZD1094I IKE STATUS FOR STACK TCPIPA   IS DOWN
EZD1094I IKE STATUS FOR STACK TCPIP    IS DOWN
EZD1094I IKE STATUS FOR STACK TCPIPB   IS DOWN
EZD1094I IKE STATUS FOR STACK TCPIPE   IS DOWN
EZD1063I IKE DISCONNECTING FROM PAGENT
EZD1055I IKE TERMINATION COMPLETE
$HASP395 IKED     ENDED
EZD1580I PAGENT IS RESTARTING IKED
S POLPROC,JOBNAME=IKED,PROG=IKED,VARS='/var/tmp/IKED_BAAaxQ'
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 246
        TO START POLPROC WITH JOBNAME IKED.
$HASP100 IKED     ON STCINRDR
IEF695I START POLPROC  WITH JOBNAME IKED     IS ASSIGNED TO USER
IBMUSER , GROUP SYS1
$HASP373 IKED     STARTED
```

Commands

The format of the policy agent commands for application is as follows:

```
F pagproc,MON,operation,application[,P=image]
```

This command includes the following parameters:

operation	START, STOP, or RESTART
application	DMD, IKED, NSSD, SYSLOGD, TRMD, or ALL
image	The TCP/IP stack name for TRMD

Tip: If you want to shut down the entire policy infrastructure, stop all monitored applications before you stop the policy agent.

Example 4-14 shows the output of the MODIFY,PAGENT,DISPLAY. The current status of all applications is displayed regardless of whether the application is monitored.

Example 4-14 Operator command to display status

```
F PAGENT,MON,DISPLAY
EZD1588I PAGENT MONITOR INFORMATION 448
APPLICATION  MONITORED  JOBNAME  STATUS      TCP/IP STACK
DMD          NO         N/A      N/A         N/A
IKED         YES        IKED     ACTIVE      N/A
NSSD         NO         N/A      N/A         N/A
SYSLOGD      NO         N/A      N/A         N/A
TRMD         YES        TRMD     ACTIVE      TCPIPA
```

4.2.9 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.

4.2.10 Centralized Policy Server

The Centralized Policy Server can simplify policy management and administration. Instead of managing a variety of local policy files scattered across multiple TCP/IP images, you can centralize management of the policy configuration on a single host using the Centralized Policy Server. The subject of Centralized Policy Services is discussed in Chapter 5, “Central Policy Server” on page 161.

4.2.11 For additional information

See *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for additional information regarding the policy agent.

4.3 Setting up the 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.

4.3.1 Starting TRMD using PAGENT

In 4.2.8, “Policy infrastructure management” on page 119, we discuss a simple way to start and monitor TRMD for one or more stacks. Alternate methods are described in 4.3.2, “Setting up the started task procedure” on page 125 and in 4.3.3, “Starting TRMD from z/OS UNIX” on page 125.

4.3.2 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 4-15. TRMD can be started from the z/OS UNIX shell or as a started task.

Example 4-15 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 4-16.

Example 4-16 Autologging TRMD from TCP/IP

```
AUTOLOG
   TRMD JOBNAME TRMD
ENDAUTOLOG
```

4.3.3 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 4-17.

Example 4-17 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>
```

4.3.4 Defining the security product authorization for TRMD

RACF definitions are required to start TRMD 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 4-18.

Example 4-18 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
```

4.3.5 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

4.4 Configuration Assistant for z/OS Communications Server

IBM Configuration Assistant for z/OS Communications Server is an optional GUI-based tool that provides a guided interface for configuring TCP/IP policy-based networking functions. With Configuration Assistant, you can easily generate the Policy Agent files through a series of wizards and online help panels. Two types of Configuration Assistant are provided:

- As a task in IBM z/OS Management Facility (z/OSMF)

  z/OSMF provides a web browser interface for a variety of z/OS system management functions. When you invoke the Configuration Assistant in z/OSMF, the Configuration Assistant runs natively in the z/OS system and you can access it through a web browser. To use the Configuration Assistant in z/OSMF, your system must be z/OS V1R11 or later. z/OSMF Configuration Assistant on z/OSMF is officially supported.

  In this book we refer to this type as *z/OSMF Configuration Assistant*.

  For more information about z/OSMF, go to the following website:

  `http://www.ibm.com/systems/z/os/zos/zosmf`

- As a stand-alone application that runs under the Windows operating system

  z/OSMF Configuration Assistant on Windows is provided on an as-is basis. You can download it from the website.

  You can use the Configuration Assistant on your workstation and then later migrate your work to the z/OSMF environment. For information about transferring Configuration Assistant data to z/OSMF, see *IBM z/OS Management Facility Configuration Guide* SA38-0652.

  For download and installation instructions, go to the following website:

  `http://www.ibm.com/support/docview.wss?uid=swg24013160`

Note: A stand-alone application that runs under Microsoft Windows will not be provided for future z/OS releases. Use of z/OSMF Configuration Assistant instead.

4.4.1 Using z/OSMF Configuration Assistant

You can use the z/OSMF Configuration Assistant to generate policy and configuration files for the following technologies:

- Application Transparent TLS (AT-TLS)
- Defense Manager Daemon (DMD)
- IP Security (IPSec)
- Network Security Services (NSS)
- Policy Based Routing (PBR)
- Quality of service (QoS)
- Intrusion detection system (IDS)

You can create configuration files for any number of z/OS images, with any number of TCP/IP stacks per image. The Configuration Assistant is intended to replace manual configuration of the policy disciplines, but it can also incorporate policy data directly from the policy agent. You can also use it as an import requestor to request configuration file import services from the policy agent.

Policy installation

z/OS Communications Server provides tools to help you more easily install policies and track progress towards the goal of implementing those policies. For each technology, Configuration

Assistant creates a set of tasks that must be performed to install and activate the policies that are created. The completion status of each of these tasks is tracked, allowing the administrator to keep track of the progress of installing the policies.

For each of the tasks, a set of detailed, customized instructions describe the steps that must be performed to accomplish that task. When the administrator is satisfied that these steps are complete, the tasks can be marked as complete.

For technologies that use SYSLOG or TRMD, tasks are included to activate those functions, including sample SYSLOGD configuration rules.

One common setup task provides a *base location* to store the generated files. Names are suggested for each of the generated files that provide a simple means for organizing files on the target z/OS system.

While the administrator is making changes to the configuration, the Configuration Assistant determines whether any of these setup tasks must be re-examined to determine whether they must be redone. For example, if image or stack names are changed, certain RACF commands might need to be redone to account for changes in authorizations.

Finally, after configurations are created or modified, you can use the “Install all files” option to show all policy files for a given technology. This option shows clearly which files are changed and must be reinstalled to make policy changes take effect.

4.4.2 General configuration steps using the Configuration Assistant

To configure the policy files, the Configuration Assistant uses logic based on a z/OS image implementation. You create the z/OS image, then the TCP/IP stack for that image, and then the technologies that you need for each stack. In an environment with multiple z/OS images, this process allows you to use a single backing store (the file where the Configuration Assistant configurations are saved) for all your environments, which helps to make the configuration tasks easy to follow.

Create a new backing store or open an existing backing store

Complete the following steps:

1. Start the z/OSMF Configuration Assistant.
2. You may either create a new backing store from scratch or start working on the existing backing store file:
   - Creating a new backing store

     When you start the Configuration Assistant for the first time, the Configuration Assistant operates on the default backing store file. You may create a New BackingStore by selecting **Action** → **Open** → **Create a New Backing Store**, in this case, you are creating the configuration from scratch.

     Since all backing store files are stored on disk by z/OSMF Configuration Assistant, you do not specify the file path to be store; you only specify the file name.
   - Opening an existing backing store

     Select **Action** → **Open** to open an existing backing store file. It gives a list of all existing backing store files which z/OSMF manages. Select the one you would like to open.

     You may also select **Action** → **Save As** to save the current backing store file with a new name, to save it as a new backing store file.

Create z/OS image and TCP/IP stack

The steps here show you a basic configuration for any single technology in a single stack, using z/OSMF Configuration Assistant:

1. Define the z/OS image as follows:

   a. In the Main Perspective panel, click **Add a new z/OS Image**, as shown in Figure 4-8.

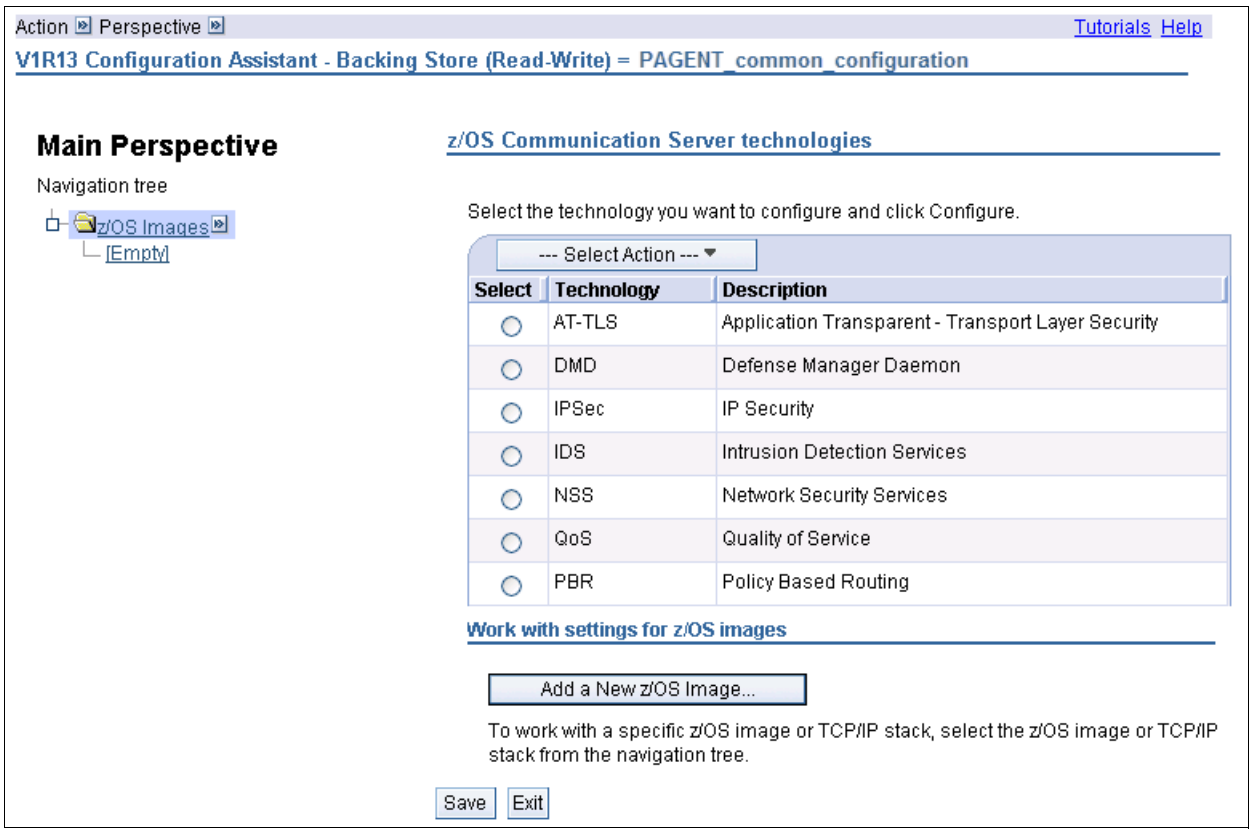


Figure 4-8 Main Perspective panel

   b. In the New z/OS Image: Information panel, enter a name to represent the z/OS System that you want to configure and select the z/OS release this image runs. In our implementation, we used the *SC30*, and V1R13, as shown in Figure 4-9.

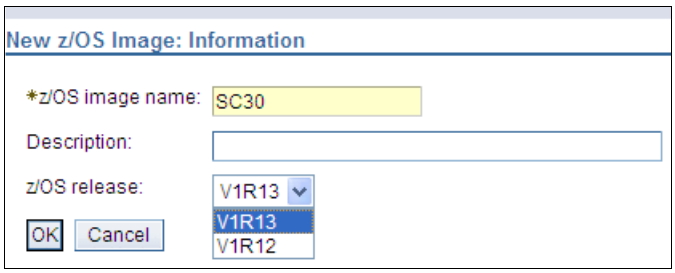


Figure 4-9 New z/OS Image: Information panel

Note: The z/OS release level of an image within Configuration Assistant can be changed at any time by clicking the image name in the navigation tree and selecting the required release from the drop-down menu. If newer release settings that are not applicable to a prior release image are configured in reusable objects, they are ignored for that image, and warnings are issued.

2. Define the TCP/IP stack as follows:

   a. After you add the z/OS image, you are prompted to define a TCP/IP stack for this Image as shown in Figure 4-10. Click **Yes**.

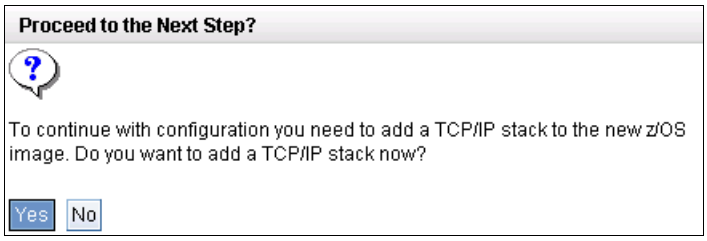


Figure 4-10 Proceed to the Next Step panel

   b. In the New TCP/IP Stack: Information panel, type the TCP/IP stack name, and then click **OK** as shown in Figure 4-11.

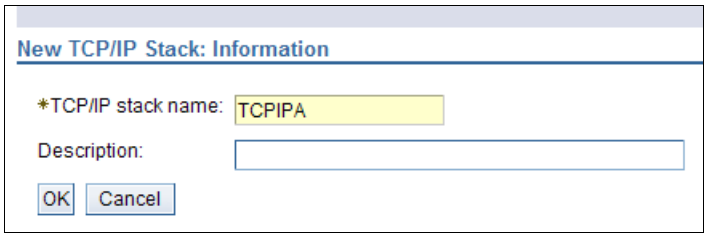


Figure 4-11 New TCP/IP Stack: Information panel

Configuring a specific technology using the Configuration Assistant

To generate a configuration of any given technology using the z/OSMF Configuration Assistant, see the chapter associated with the required technology. We use the AT-TLS technology to illustrate how to define and install the configuration as follows:

1. Start creating the policy of each technology for each TCP/IP stack image:
   a. Back in the Main Perspective, select the technology that you would like to configure from the z/OS Communication Server technologies section.
   b. Click **Select Action** → **Enable**. The status column then shows `Incomplete` (Figure 4-12) because you have not configured the policy yet. Click **Select Action** → **Configure** to start the policy configuration.

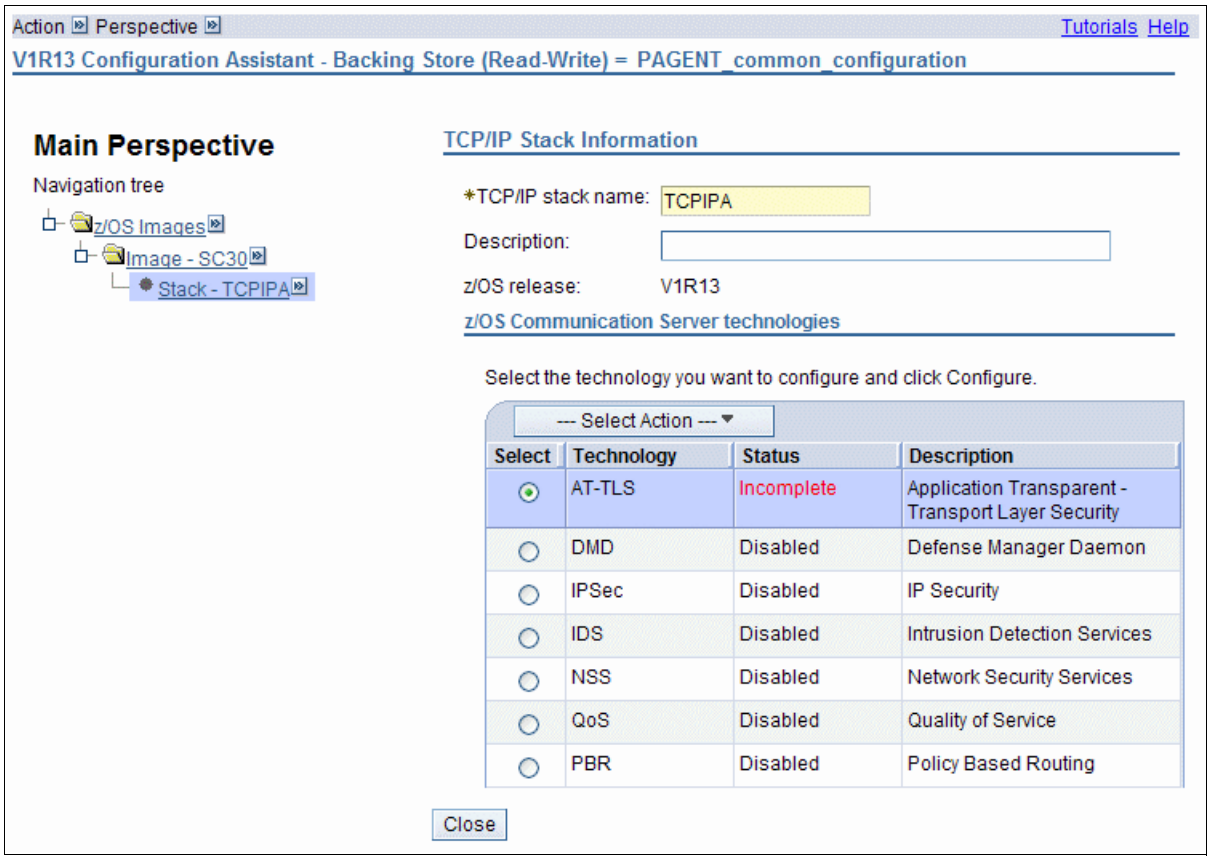


Figure 4-12 Starting the configuration of the selected technology

c. For the next panel, depending on the technology you have selected on the previous step, different Perspectives are displayed (for example, if you select AT-TLS technology, AT-TLS Perspective is displayed, as shown in Figure 4-13). You may start configuration for the technology you have selected (AT-TLS in this example) now.

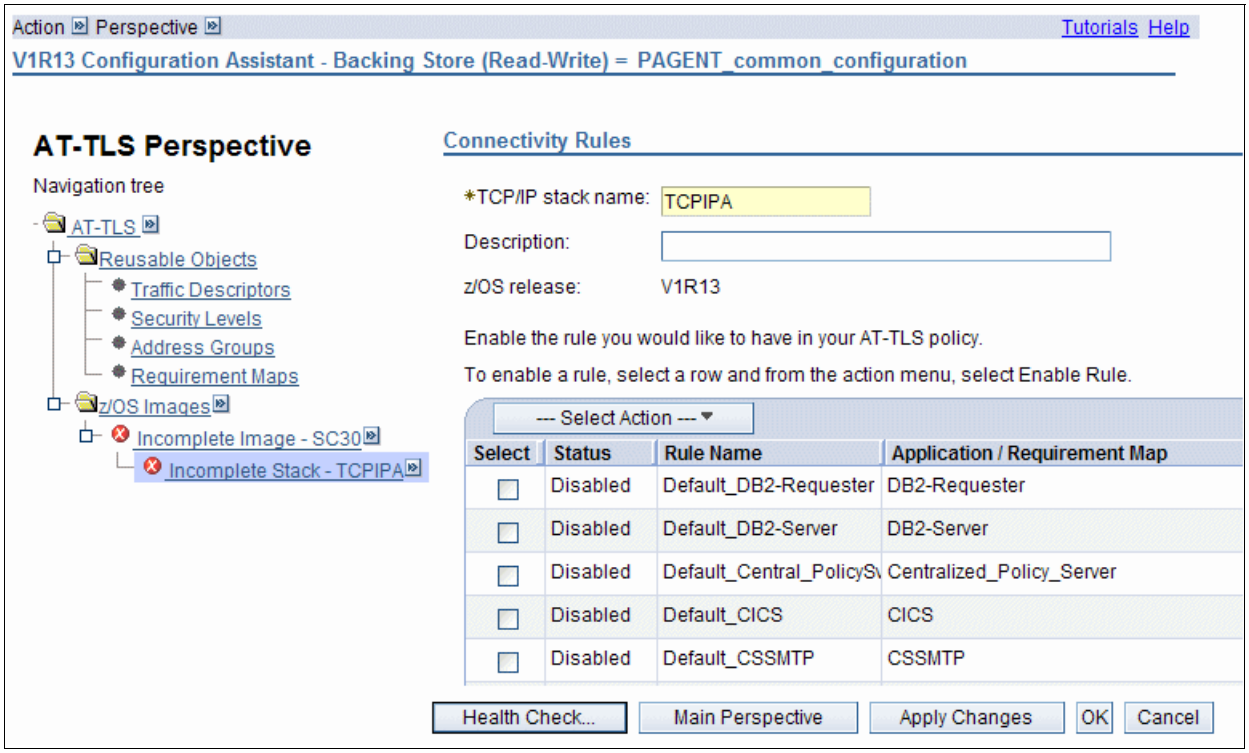


Figure 4-13 Individual perspective for each technology (AT-TLS)

Displaying Application Setup Tasks

To guide you through the configuration setup tasks, Application Setup Tasks menu is provided. Complete the following steps:

1. On the technology perspective, click the image name that you want to configure, and then select the **Image** tab. If the message shown in Figure 4-14 is displayed, specify the z/OS image level settings (such as key ring name) first, and then click **OK**.

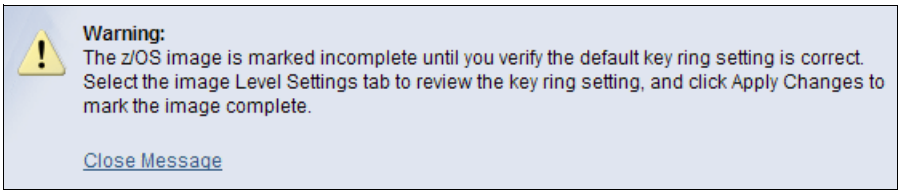


Figure 4-14 Warning on image level settings

2. Click **Application Setup Tasks**. On Application Setup Tasks for Image SC30 panel (Figure 4-15), a list is displayed of all the setup tasks that must be performed to successfully enable the technology. Select a task and click **Select Action** → **Task Details**. A panel opens (Figure 4-16), which lists the actions you can perform.

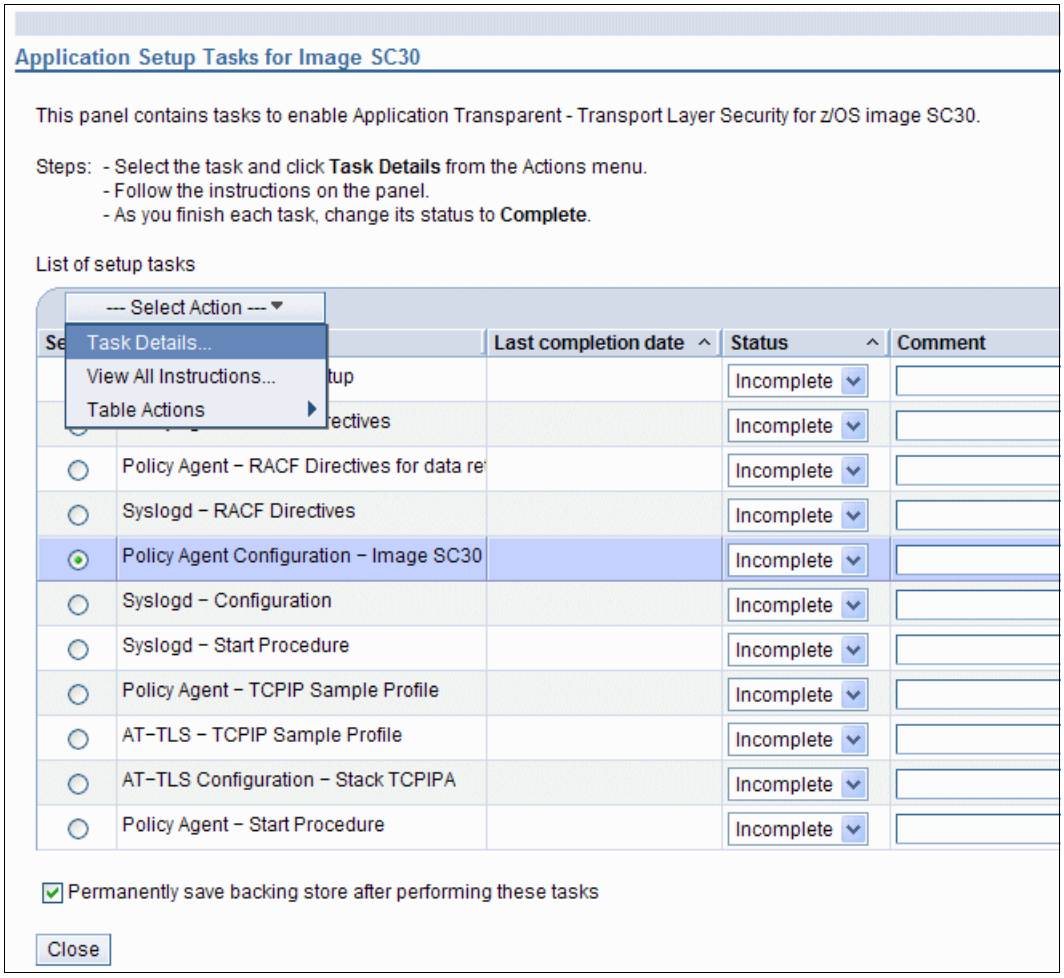


Figure 4-15 Application setup tasks panel

Figure 4-16 Task panel

3. On the task panel, click the following options:
   – Click **Instructions** to display a set of detailed steps to accomplish this specific task (Figure 4-17).

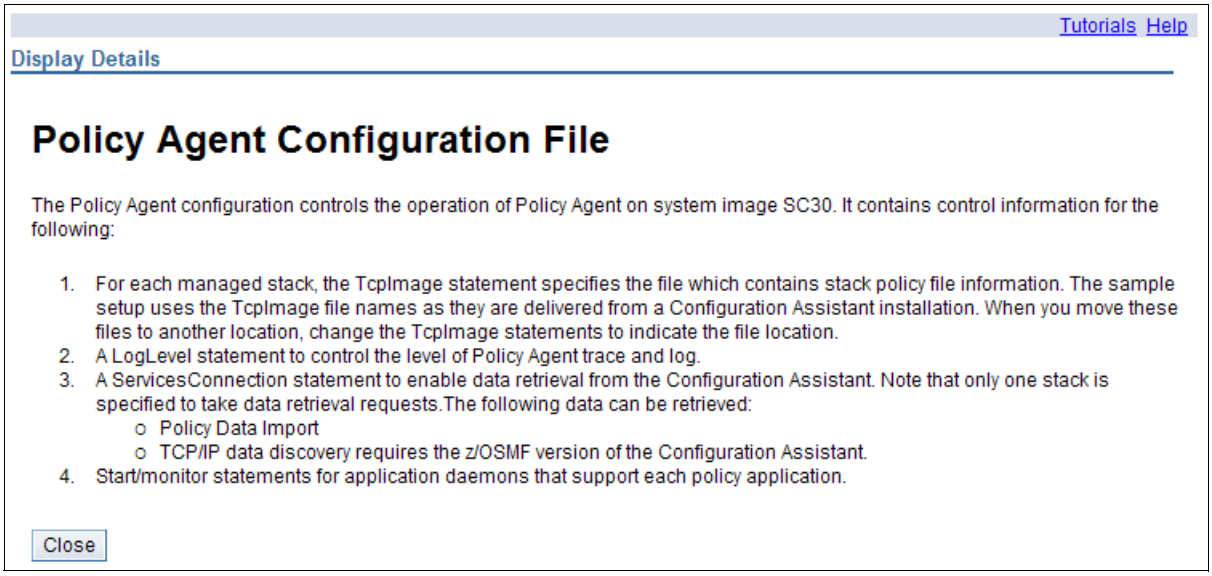


Figure 4-17 Instructions panel

   – Click **View Contents** to display the generated policy for this task. You may come back later to see the contents of the policy after the configuration is completed.
   – Click **Install File** to save the policy to the local z/OS system (if using z/OSMF Configuration Assistant) or transfer the policy to the z/OS system using FTP.
   – Select the **Mark task as complete** option to show the status of this task is changed to `Complete` in the list of setup tasks.

   Instructions display how to use the Eclipse Help System in a web browser panel. The browser can be used to print the instructions.

4. You can also view instructions for all the tasks by clicking **Select Action** → **View All Instructions** on Application Setup Tasks for Image SC30 panel. In this case, the panel lists the instructions for all individual tasks.

Installing the generated policy

When you are done with configuration, the policy file must be saved to z/OS system so that the z/OS Policy Agent can read it. There are two methods to install, as shown in Figure 4-18 on page 135:

- Save to disk (local z/OS file system)

  This method is supported only on z/OSMF Configuration Assistant. This option stores the policy file directly on the system on which z/OSMF is running. with the specified file name.

- FTP (to z/OS file system)

  For Configuration Assistant on Windows system, this is the only option.

  This method is also supported on z/OSMF Configuration Assistant to transfer the policy to another z/OS system.

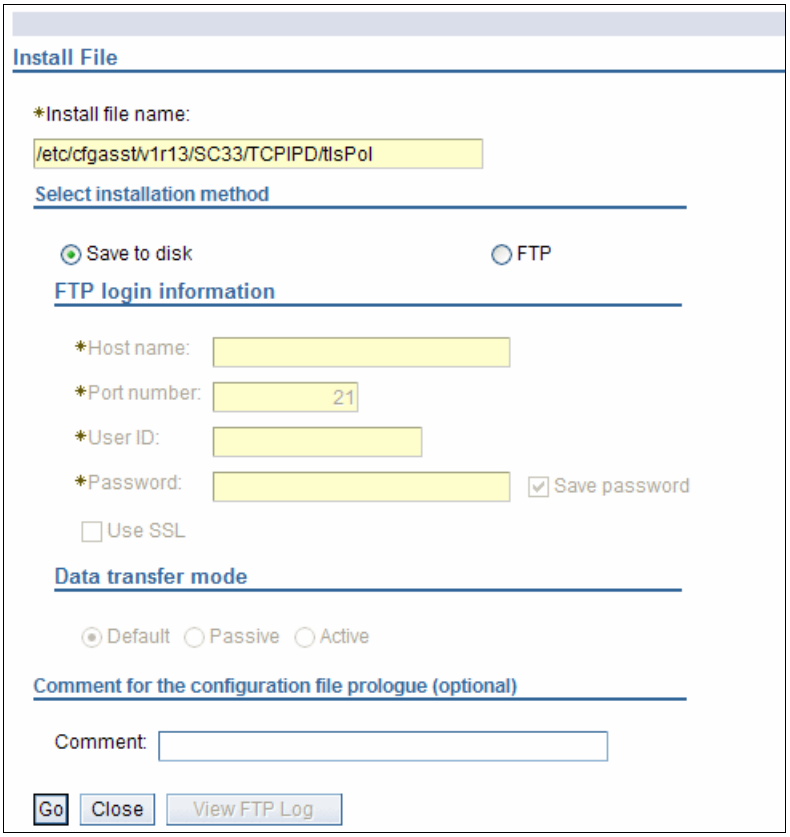


Figure 4-18 Installation options

After saving to the disk or transferring with FTP, you see comments added in the policy file, as shown in Example 4-19. The comments identify the backing store file (1).

Example 4-19 Location of backing store file and FTP History

```
##
## 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 13
## Backing Store = ATTLS_Telnet_V1R13_BackingStore
## FTP History:
## 2011-07-18 18:01:31 : Save To Disk 1
##
## End of Configuration Assistant information
```

Saving the changes to a current backing store

Select **Action** → **Save** to save changes to the current backing store you are working on.

Apply the policy to the policy agent

Apply the policy by specifying its stored path and filename to the policy agent configuration file and refreshing the policy agent.

4.4.3 Discovery of TCP/IP profile function

The discovery of TCP/IP profile function is a passive service that allows z/OSMF Configuration Assistant to discover IP addresses for each defined stack. Policy Agent retrieves information from the TCP/IP profile. z/OSMF Configuration Assistant can import this data and use it as configuration data.

This function is particularly useful when common configuration of multiple stacks, using reusable rules, is used. Common configuration allows an IPSec connectivity rule to be reused across multiple stacks. Rather than enter each stack's IP addresses manually for the local address *names* within the rule, you can discover them automatically by using this function. For more information about how to use common configuration and local address names, see 4.4.4, "Common configuration of multiple stacks" on page 139.

Preparing PAGENT for discovery of TCP/IP profile requests

To allow z/OSMF Configuration Assistant to discover local IP addresses, complete the following steps:

1. In the policy agent main configuration file, define a port and the TCP/IP stack name to which the Configuration Assistant will connect by coding the `ServicesConnection` statement. Optionally, configure secure connections from the profile data requestors. These steps are described in step 1 of "Preparing PAGENT for configuration file import requests" on page 157
2. Use the z/OS system security facility (for example, RACF) to authorize the import requestor user IDs that can access the requested profile information. Give READ access to SERVAUTH profile `EZB.PAGENT.sysname.image.ptype` to the user who can import the profile information. For `image`, specify the TCP/IP stack name specified on the `TcpImage` statement. For `ptype`, use CFGSERV or specify a wildcard value. In our environment, we gave permission to user CS03, as shown in Example 4-20.

Example 4-20 Permission to discover TCP/IP profile data

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
SETROPTS GENERIC(SERVAUTH)
RDEFINE SERVAUTH EZB.PAGENT.SC32.TCPIPC.* UACC(NONE)
PERMIT EZB.PAGENT.SC32.TCPIPC.* CLASS(SERVAUTH) ID(CSO3) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Instructions to discover local IP addresses

z/OSMF Configuration Assistant allows you to discover IP addresses for each stack defined in the policy. The discovered information includes the IP address, the interface name, and the interface type, such as OSA or DVIPA. Only strategic interface types are discovered:

- OSA-Express
- Hipersockets
- MPC Point-to-Point
- Dynamic XCF
- Static VIPA
- Dynamic VIPA (DVIPA)
- Loopback

To discover local IP addresses, use the following steps:

1. Connect to z/OSMF Configuration Assistant, and open the backing store file containing the policy definition that you would like to discover IP addresses for.
2. In the IPSec perspective navigation tree, select the stack name you for which you want to discover the IP addresses and then click the **Local Addresses** tab.
3. If the table in the Local Addresses tab is empty, you have not added any local interfaces manually. Click **Select Action** → **Discover** as shown in Figure 4-19

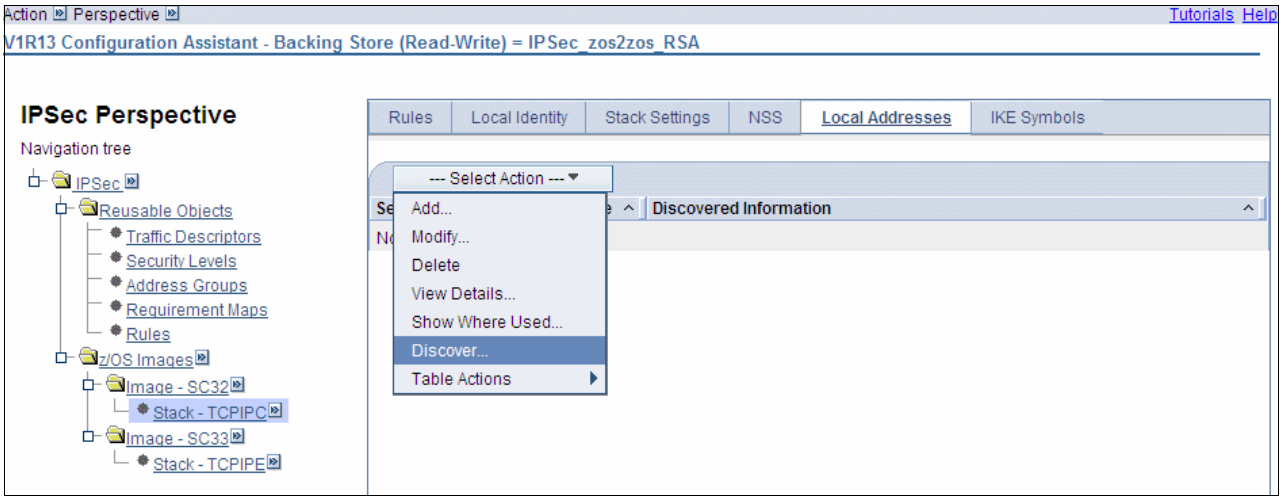


Figure 4-19 Discover local IP addresses

4. In the Discover Stack Local Addresses panel, the Image and Stack name are already selected as the discover function is invoked while you are editing a stack. Fill in the Host Connection information to connect to policy agent, and then click **Go** (Figure 4-20).

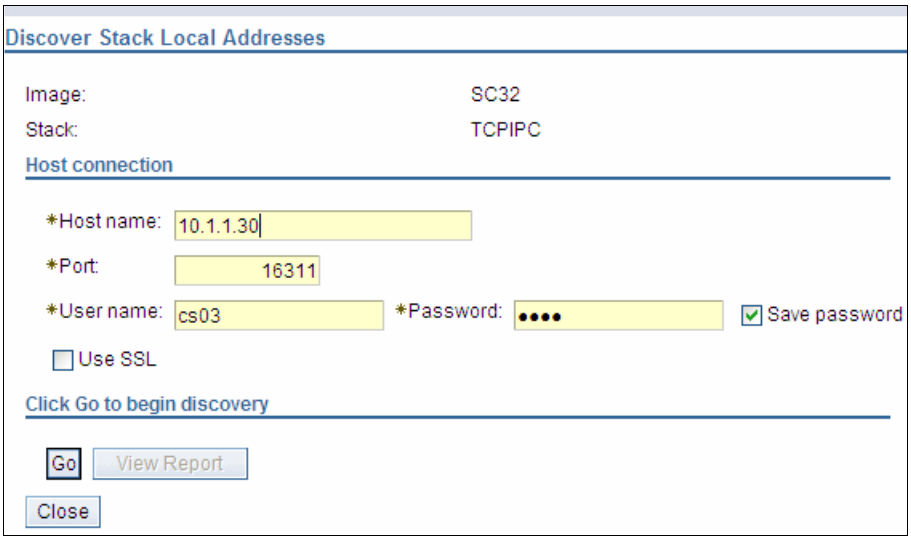


Figure 4-20 Discover Stack Local Addresses

5. When the discover local address process is complete, the information message shown in Figure 4-21 may be seen. Click **View Report** to see which interface names were not used and why.

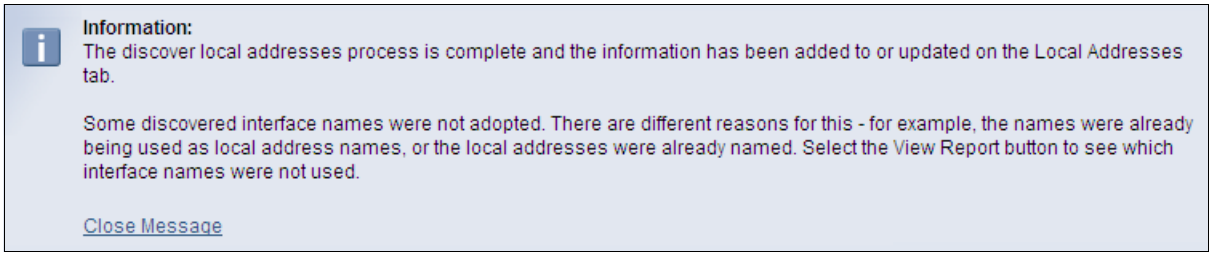


Figure 4-21 Information message for interface names

6. In our case, the report showed the reason (Figure 4-22) why some interface names were not added. The reason was because they were DVIPAs without an interface name. Click **Close** to exit the report and **Close** to exit the Discover Stack Local Addresses panel.

The following IP addresses did not already exist, and were added to the local addresses tab. The associated interface names were not used, because of the reasons shown in the table.

IP Address	Discovered Information	Reason
10.1.8.30/24	Type=Dynamic VIPA Define	This IP address does not have an interface name.
10.1.8.10	Type=Dynamic VIPA Backup, backup rank=90	This IP address does not have an interface name.
10.1.9.0/24	Type=Dynamic VIPA Range	This IP address does not have an interface name.

Figure 4-22 View report reason for not adding names

7. The Local Addresses tab looks like Figure 4-23 when it has been populated by the discover function. Click **Apply Changes**.

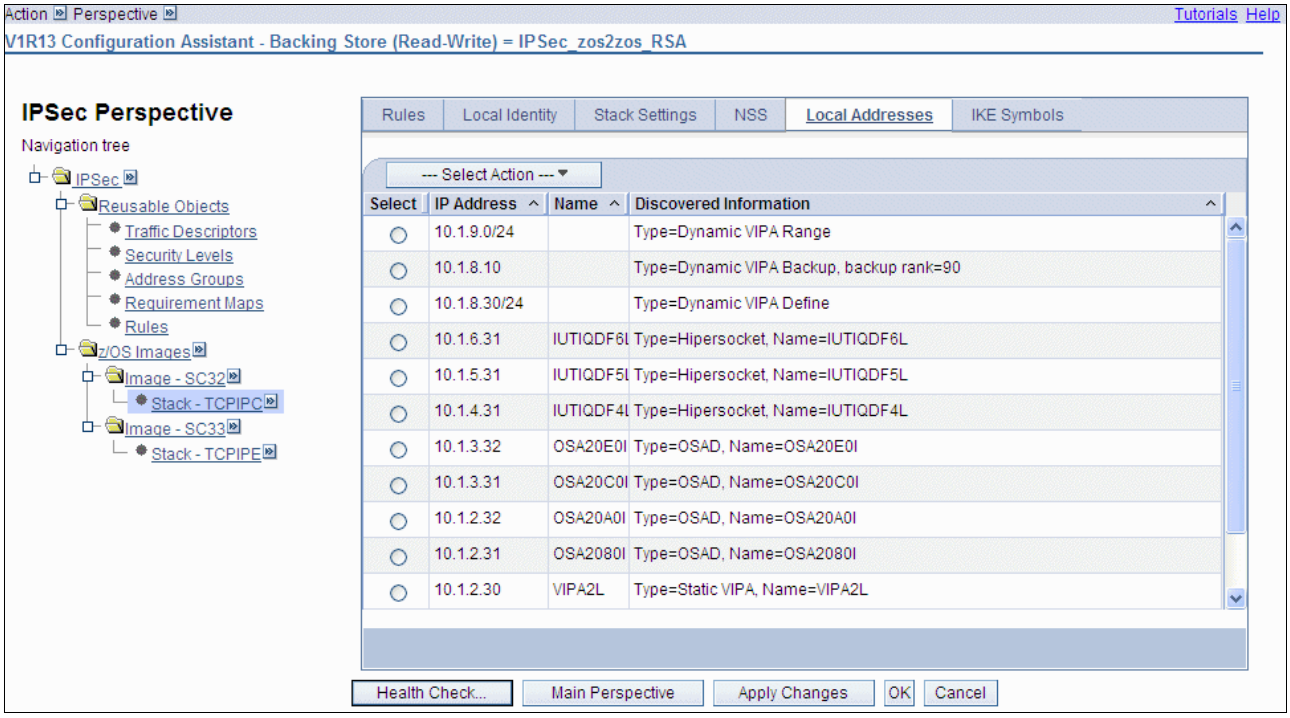


Figure 4-23 Populated local address table

Tip: Names assigned by discovery are available in all other stacks but without IP address values. When discovery is run on the other stacks, z/OSMF Configuration Assistant will update the IP address value for all address names that match discovered interface names.

4.4.4 Common configuration of multiple stacks

When using z/OSMF Configuration Assistant to configure IPSec rules, multiple stacks will likely require the exact same rule, with the only differences being the local IP address and the local IKE identity, which are stack-specific. To enable a reduction in the number of defined rules, and to make updating rules easier, use the common configuration function. Common configuration uses the following rules and names:

- Reusable rules

  A reusable connectivity rule is created a single time and assigned to multiple TCP/IP stacks. If a reusable rule needs to be updated, only that single rule needs to be modified in z/OSMF Configuration Assistant and the changes can be installed on all stacks. You can create reusable rules for both IP filtering and for dynamic IPSec tunnels.

- Local address names

  Local IP addresses are unique for each stack that might prohibit a single rule from being reusable across stacks. By configuring the local address as a name, rather than a specific IP address, the rule becomes reusable across stacks.

  Note: The same local address name used in the reusable rule must be configured for each stack that will use the reusable rule. The IP address that this name resolves to can be different for each stack.

- Local IKE identity names

  As with local IP addresses, local IKE identities are unique for each stack. A local IKE identity name can be configured to make the rule reusable.

  Note: The same local IKE identity name used in the reusable rule will need to be configured for each stack which will use the reusable rule. The local identity that this name resolves to will be different for each stack.

Creating new reusable rules

To show you how to create a new reusable rule, the following steps create a dynamic IPSec tunnel rule for use between two z/OS stacks and Windows. The scenario configured is based on the one shown in “IPSec between z/OS and Windows: RSA mode” on page 882. To show how to assign reusable rules to multiple stacks, we added image SC32 and stack TCPIPC to the configuration. Complete the following steps to create a new reusable rule:

1. Connect to z/OSMF Configuration Assistant, and open the backing store file containing the policy definition that you would like to create reusable rules for. If you are creating a new policy, add your required image names and stack names as described in “Create z/OS image and TCP/IP stack” on page 129.
2. For each stack that will share the reusable rule, configure the set of local IP addresses and assign names to each address. This step can be done manually from the IPSec perspective by selecting the stack name, clicking the Local Addresses tab and clicking **Add** to add an IP address and assign a name to it. However, the easier method is to use z/OSMF Configuration Assistant to discover the local IP addresses as described in 4.4.3, “Discovery of TCP/IP profile function” on page 136.

3. For each stack, configure a local IKE identity name. From the IPSec perspective, select the stack name, click the IKE Symbols tab and click **Select Action** → **Add**. In the IKE Identity panel, configure a name, and select the IKE identity type and identity as shown in Figure 4-24. We chose name IKE_zOS. We use an identity type of X.500 distinguished name and a local identity of `CN=SC32 RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us` for image SC32 and `CN=SC33 RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us for image SC33`. The names are derived from the subject's name in the personal certificate stored in RACF. Click **OK** and **Apply Changes**.

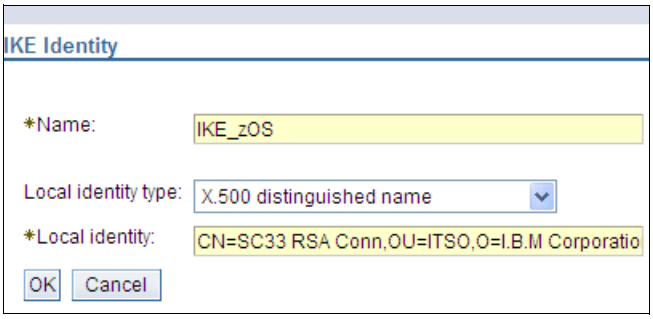


Figure 4-24 Local IKE identity name

4. From the IPSec perspective, click the **Rules** node in the navigation tree and click **Select Action** → **Add** as shown in Figure 4-25.

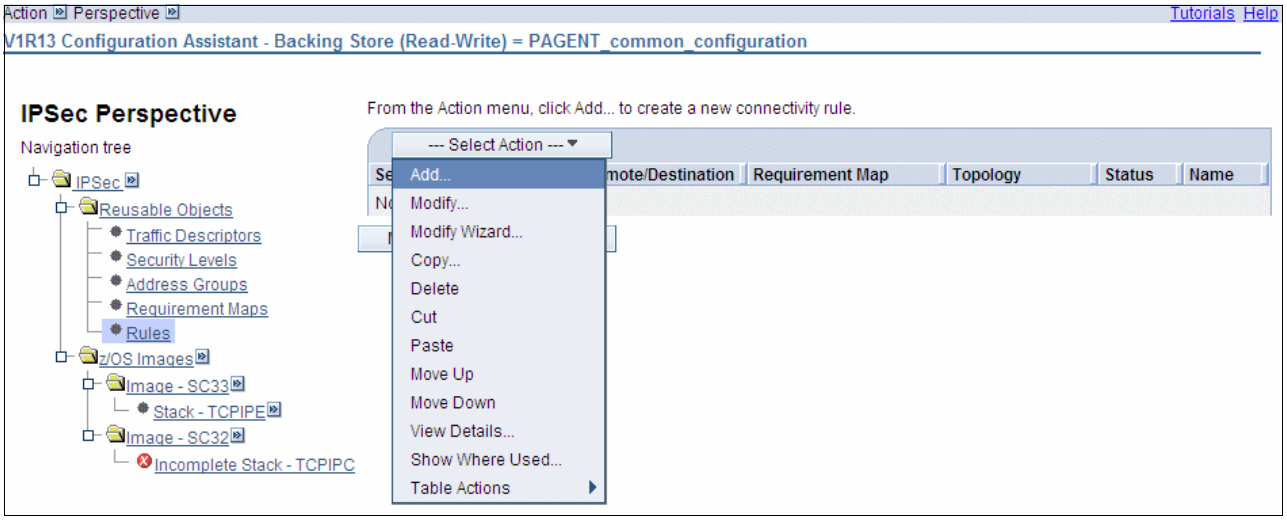


Figure 4-25 Add reusable rule

5. In the Welcome panel, select the **Typical** connectivity rule. Click **Next**.
6. In the Network Topology panel, select **This connectivity rule will contain a security level using IPSec tunnels**. Also, select **Host to Host**. Click **Next**.

7. In the Data Endpoints panel, we enter a Connectivity rule name of `All_Traffic_RSA`, and a Remote data endpoint of the Windows workstation IP address, 10.1.100.222. For the Local data endpoint, select Local IP address name and choose `VIPA1L` from the drop-down menu as shown in Figure 4-26. Click **Next**.

> **Note:** For this rule to be reusable in other stacks, the local IP address name VIPA1L will need to be defined with a valid local IP address in each stack's Local Addresses table. In our environment, VIPA1L was a static VIPA link name in each stack.
>
> If you do not have common link names across stacks, then for each stack add the local IP address to be used in this rule to your Local Addresses table, and assign the same symbolic name to it in each stack, for example `PRODVIPA`.

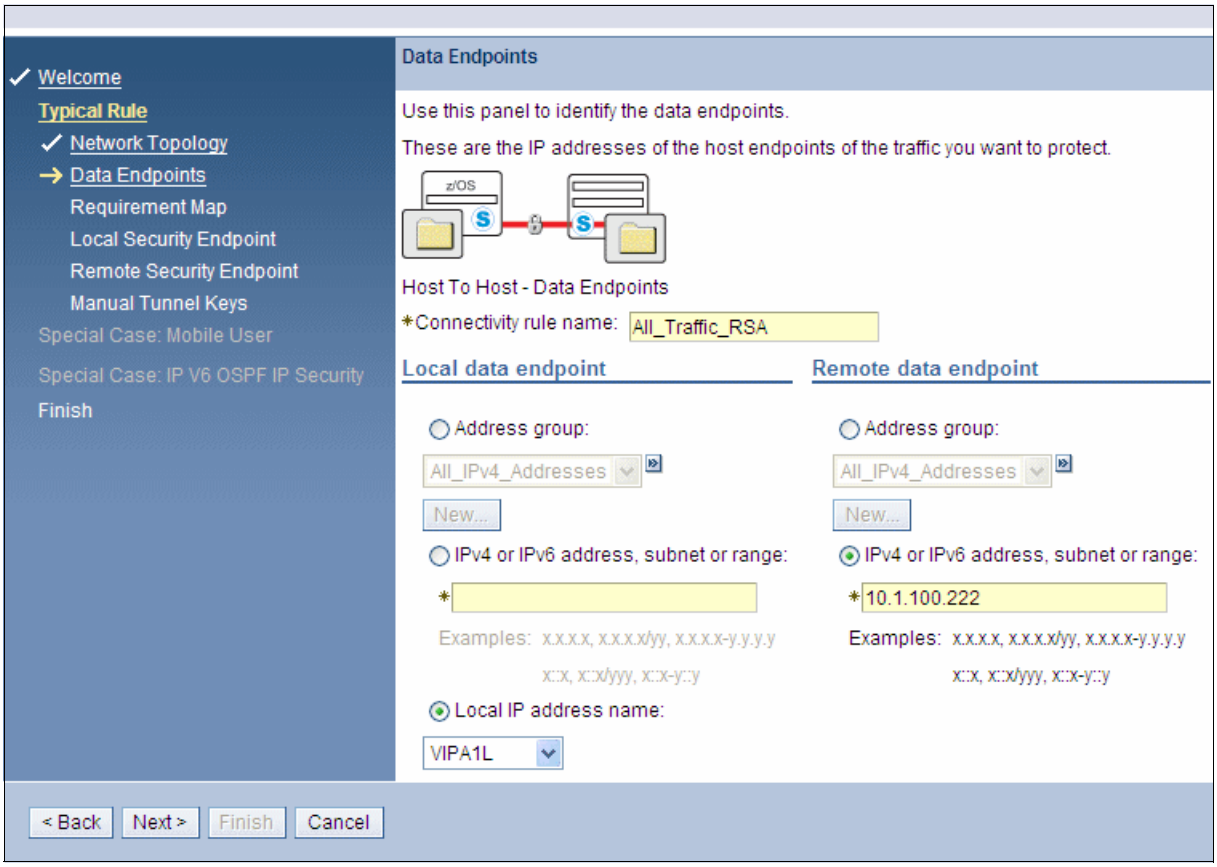


Figure 4-26 Data endpoint using local IP address name

8. In the Requirement Map panel, choose **Select an existing requirement map** and select `All_Traffic_PFS` from the drop-down menu, and then click **Next**.
9. In the Local Security Endpoint panel, select symbol and use IKE_zOS as the IKE identity name as shown in Figure 4-27. Click **Next**.

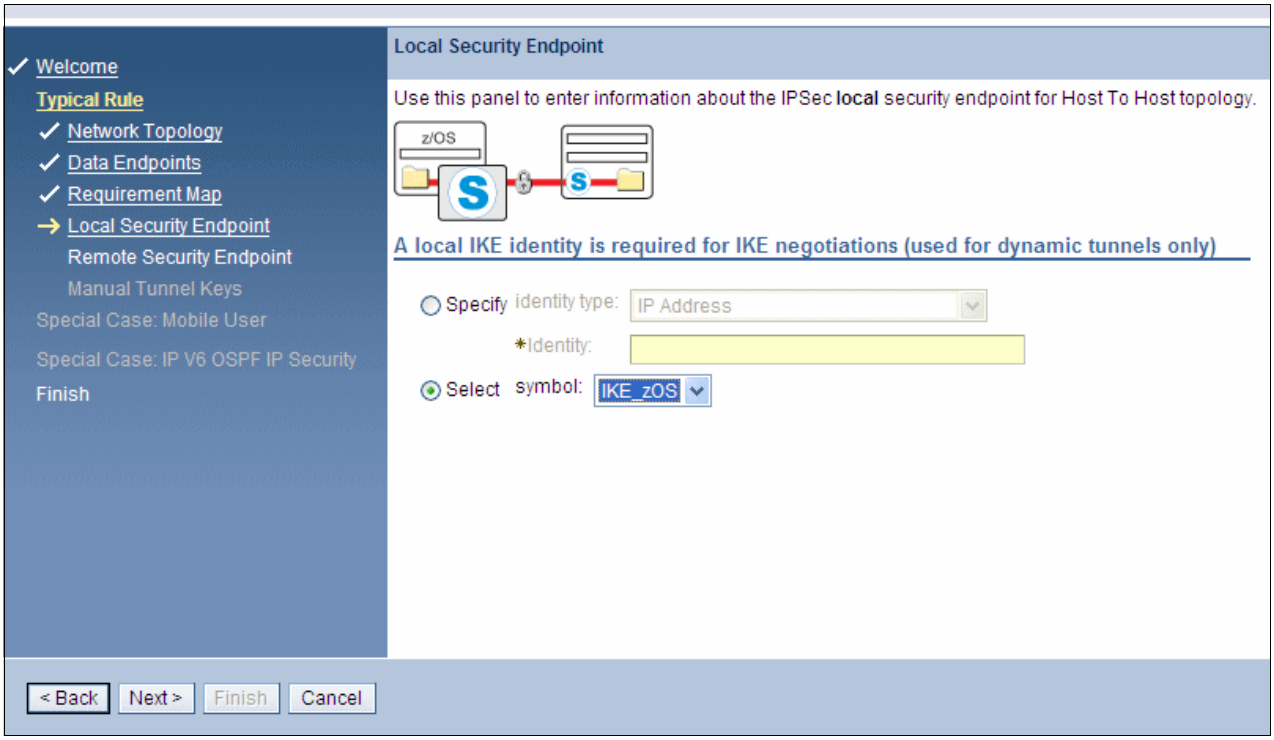


Figure 4-27 Local security endpoint IKE identity name

10. In the Remote Security Endpoint panel, we choose the remote identity type of X.500 distinguished name and the following remote identity, which is derived from the RACF certificate subject name:

```
CN=WINXP RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us
```

We select Digital signature as the authentication method (Figure 4-28), and click **Next**.

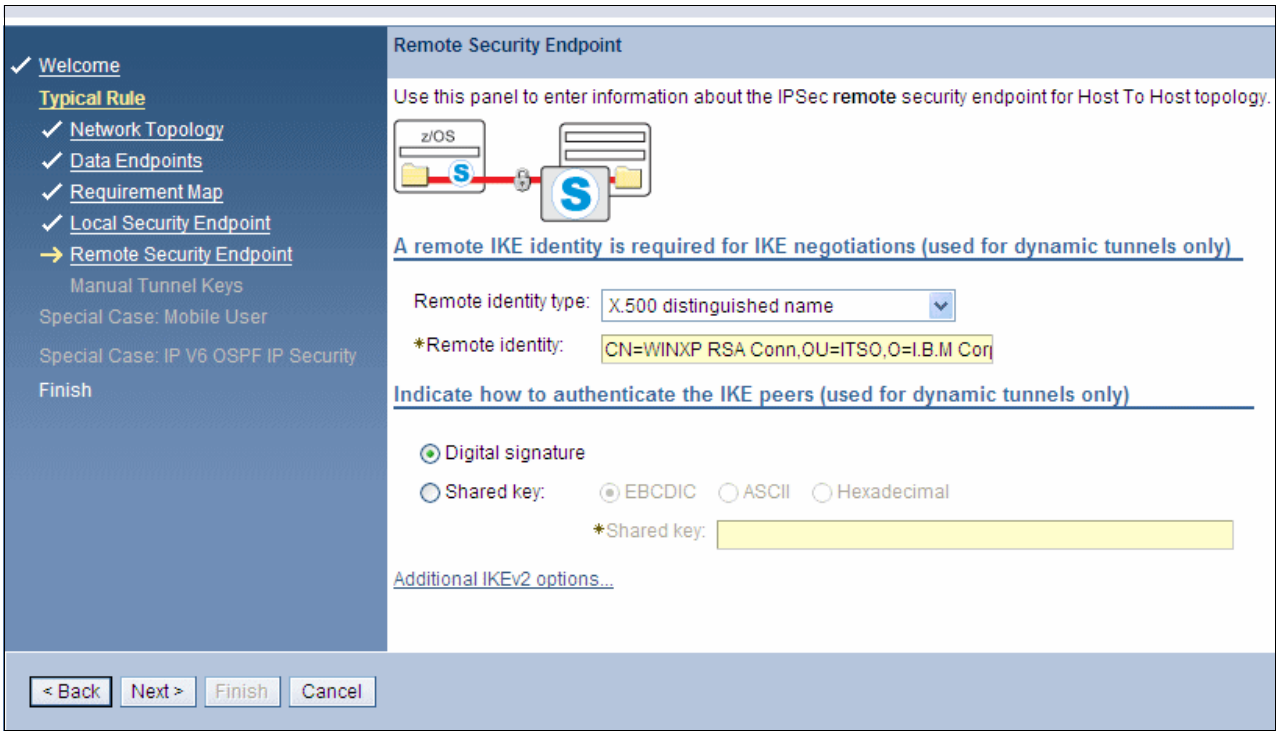


Figure 4-28 Remote security endpoint

11. Choose the required filter logging and click **Finish**. The new reusable rule is shown in Figure 4-29.

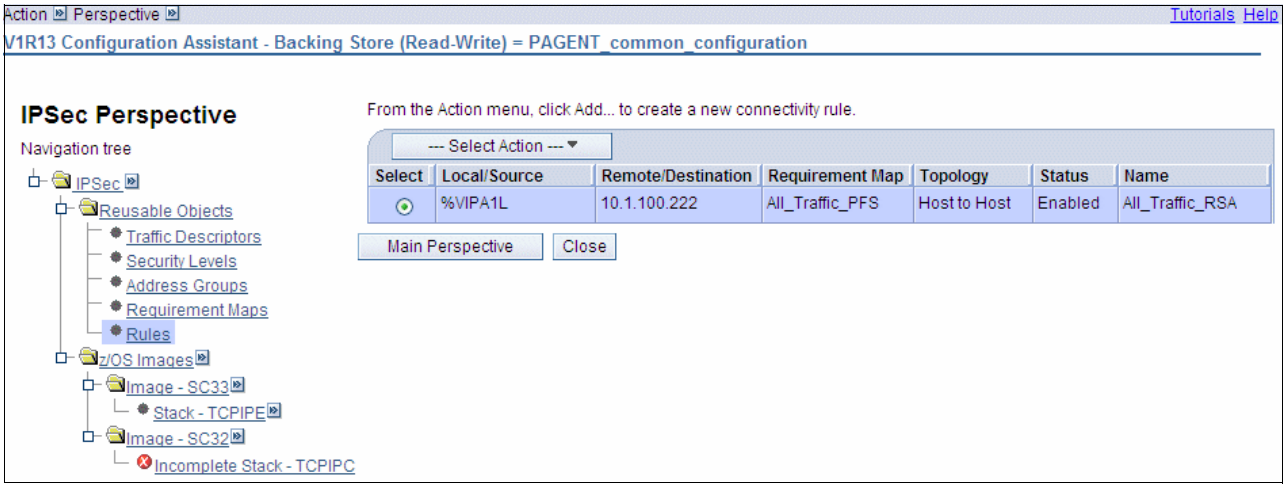


Figure 4-29 New reusable rule

We now have a reusable rule that can be assigned to any or all of our defined stacks.

Next, we show how to assign the rule to multiple stacks.

Assigning a reusable rule to a stack

To assign a reusable rule to one or more stacks complete the following steps:

1. In the IPSec perspective, select Stack-TCPIPE and click the **Rules** tab. Click **Select Action** → **Add** as shown in Figure 4-30.

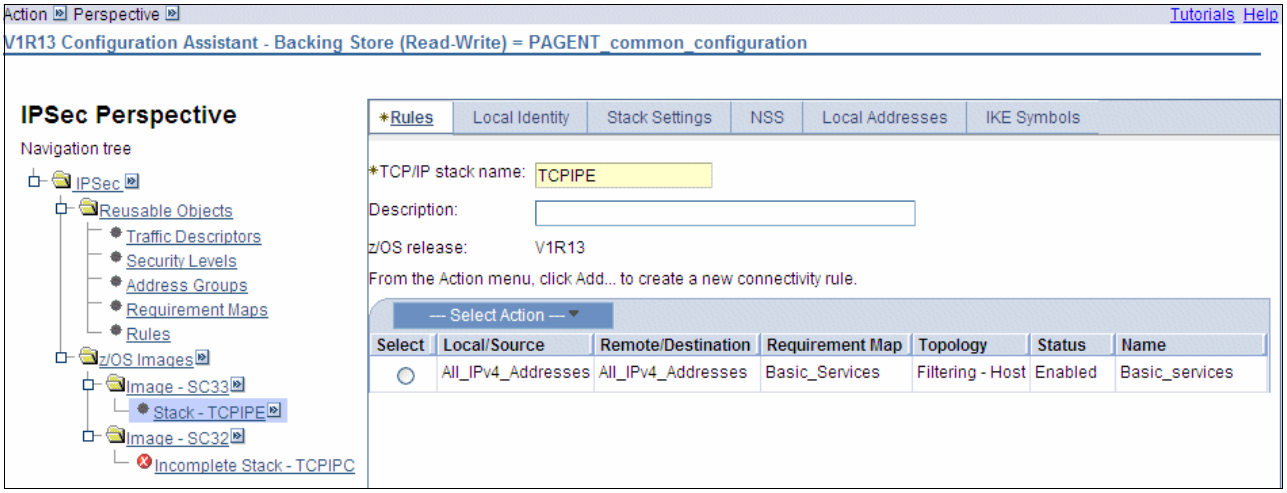


Figure 4-30 Add a new connectivity rule

2. In the Welcome panel, click **Reusable** rule and select **All_Traffic_RSA** from the drop-down menu (Figure 4-31). Click **Next**.

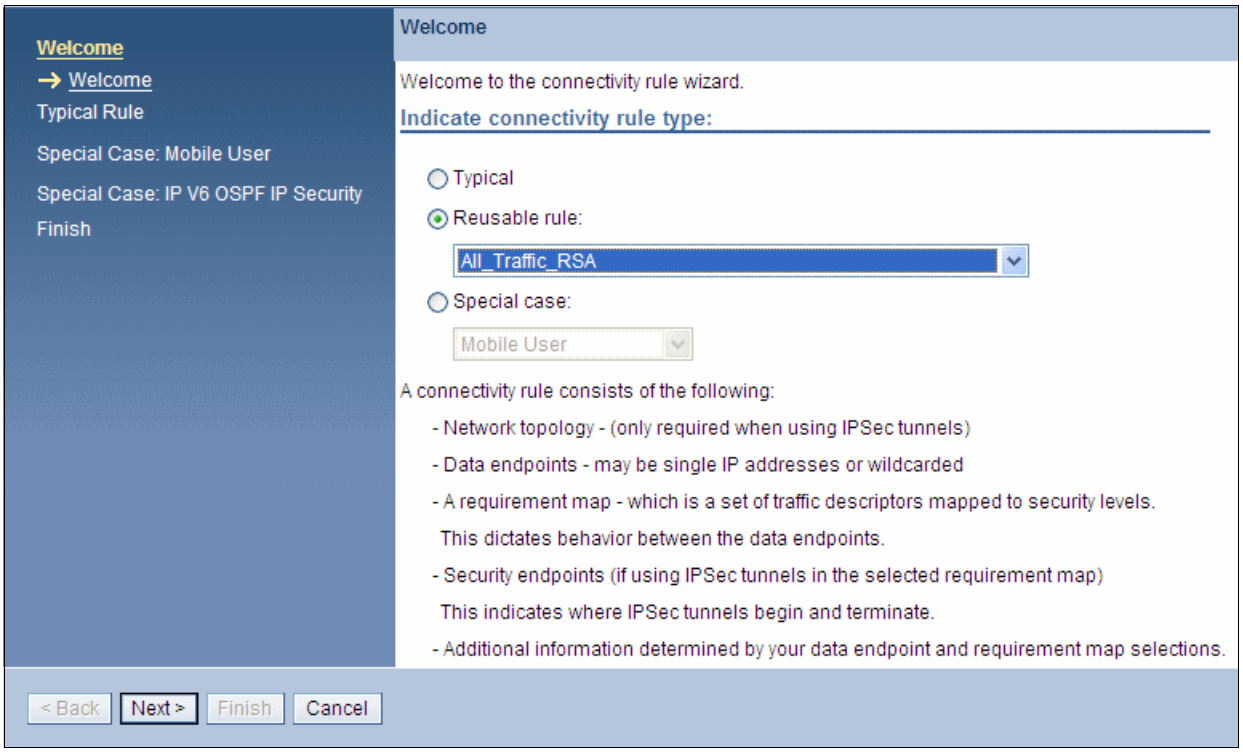


Figure 4-31 Indicate connectivity rule type

3. Review the Finish panel to ensure the information is correct. If you want to, at this point, you may change the connectivity rule name from the default of the reusable rule name. We let it default as shown in Figure 4-32. Click **Finish**.

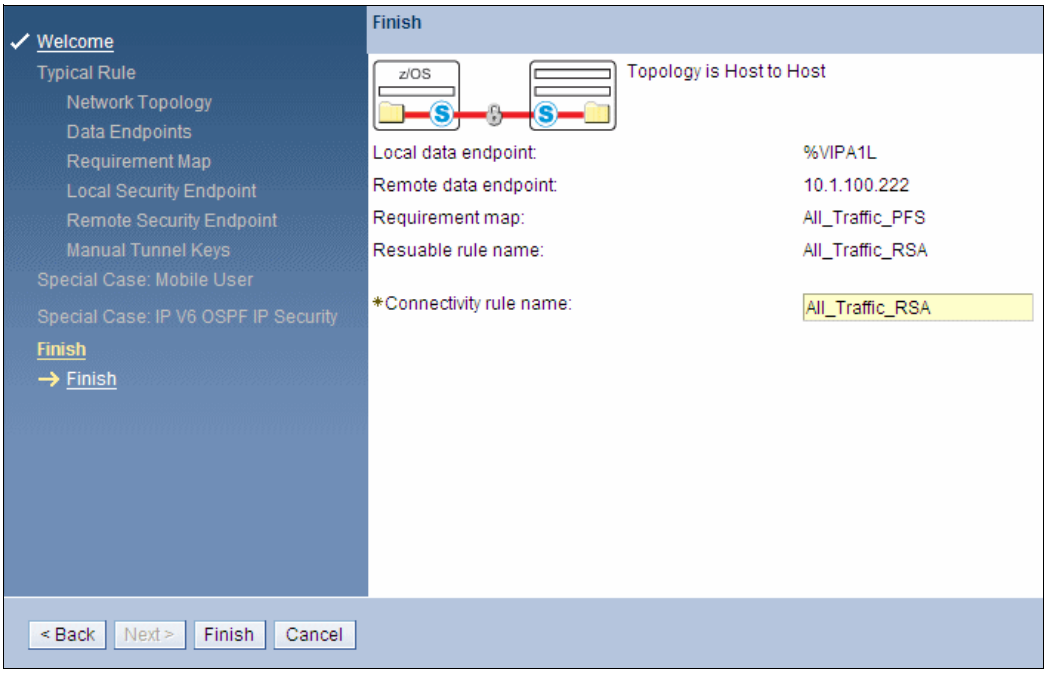


Figure 4-32 Finish assigning reusable rule

4. The reusable rule is now assigned to stack TCPIPE, as shown in Figure 4-33. Note the **(R)** character showing that this is a reusable rule. Click **Apply Changes**.

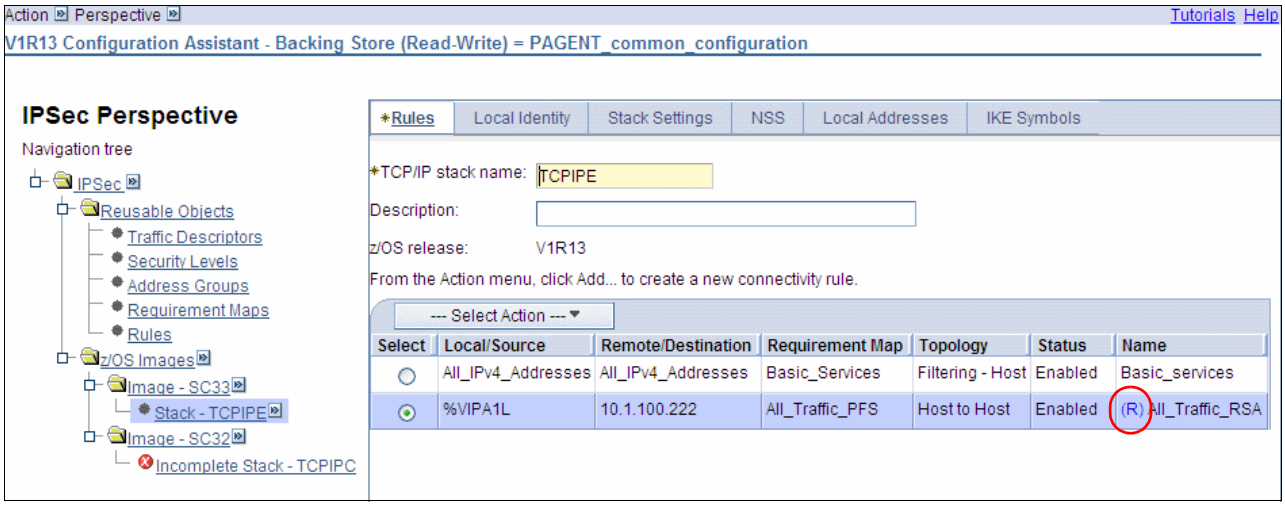


Figure 4-33 Reusable rule assigned to stack TCPIPE

5. Now, to assign the same reusable rule to stack TCPIPC, select **Incomplete Stack-TCPIPC**. Click **Yes** in response to the prompt to add a connectivity rule to the stack now. In the Welcome panel, click **Reusable** rule and choose **All_Traffic_RSA** from the drop-down menu. Click **Next**.
6. Review the Finish panel to ensure the information is correct. If you want to, at this point, you may change the connectivity rule name from the default of the reusable rule name. We let it default. Click **Finish**.
7. The reusable rule is now assigned to stack TCPIPC as shown in Figure 4-34. Note the **(R)** showing that this is a reusable rule. Click **Apply Changes**.

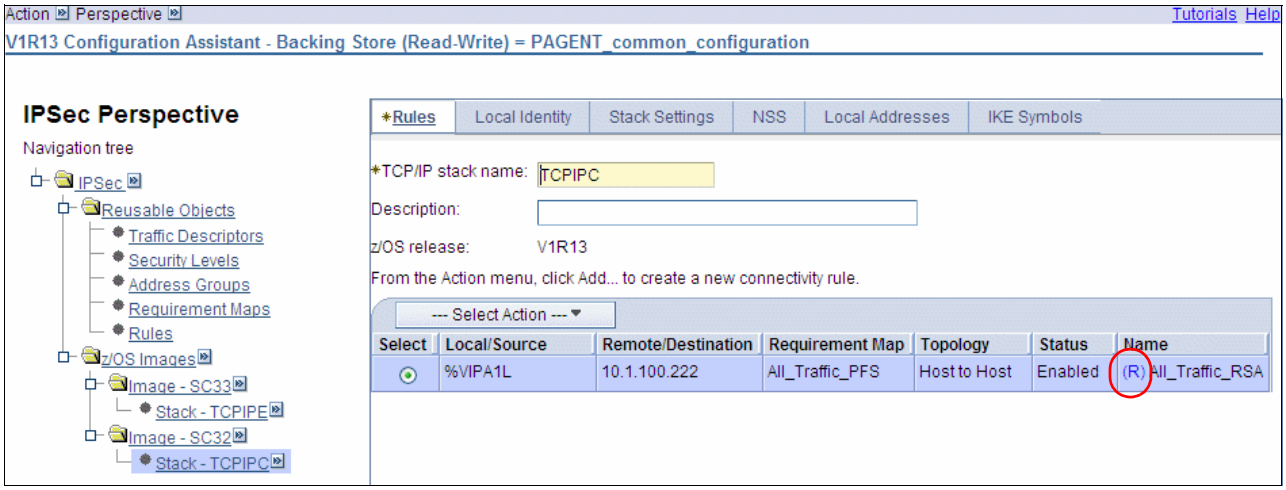


Figure 4-34 Reusable rule assigned to stack TCPIPC

Converting a stack specific rule to a reusable rule

To help you migrate from stack-specific rules to reusable rules, you may convert a stack rule to a reusable rule. You may also change a reusable rule into a stack-specific rule if you need to temporarily change a rule setting for one stack only, for example logging levels.

In our environment, we have a stack-specific rule for TCPIPE to allow basic services to flow without encryption. In the following steps, we show how to convert this rule to a reusable rule and how to assign it to multiple stacks.

1. In the IPSec perspective select Stack-TCPIPE and click the Rules tab. Select the Basic_Services rule and click **Select Action** → **Make Reusable**, as shown in Figure 4-35.

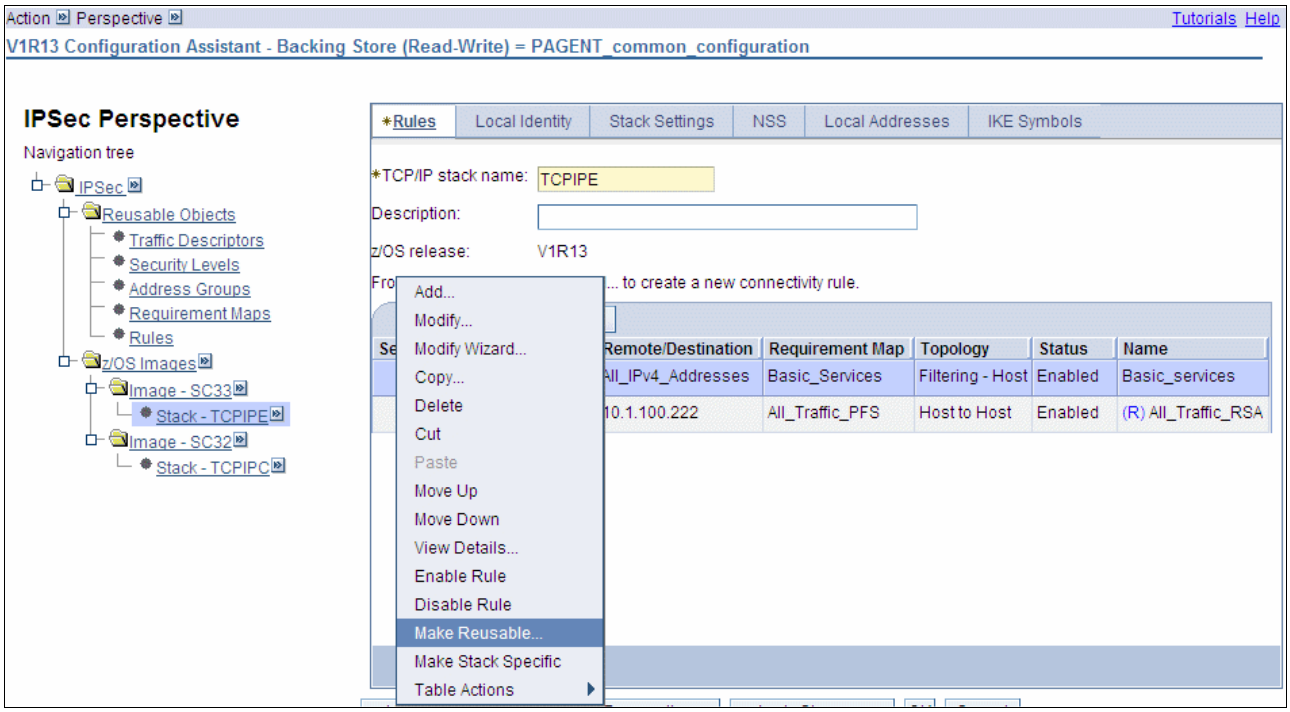


Figure 4-35 Make connectivity rule reusable

2. In the Converting Connectivity Rule to Reusable Rule panel, we could change the name of the reusable rule name from the default of the stack specific rule. We left it to default as shown in Figure 4-36. Click **OK**.

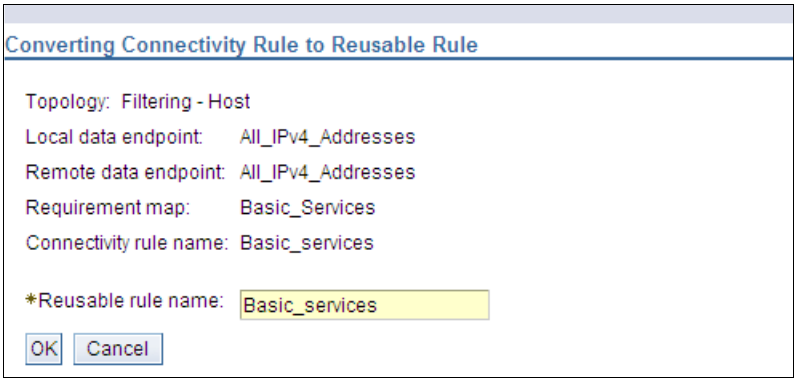


Figure 4-36 Converting connectivity rule to reusable

3. The Basic_services rule is now a reusable rule as shown by the (R) in the rule name in Figure 4-37. Click **Apply Changes**.

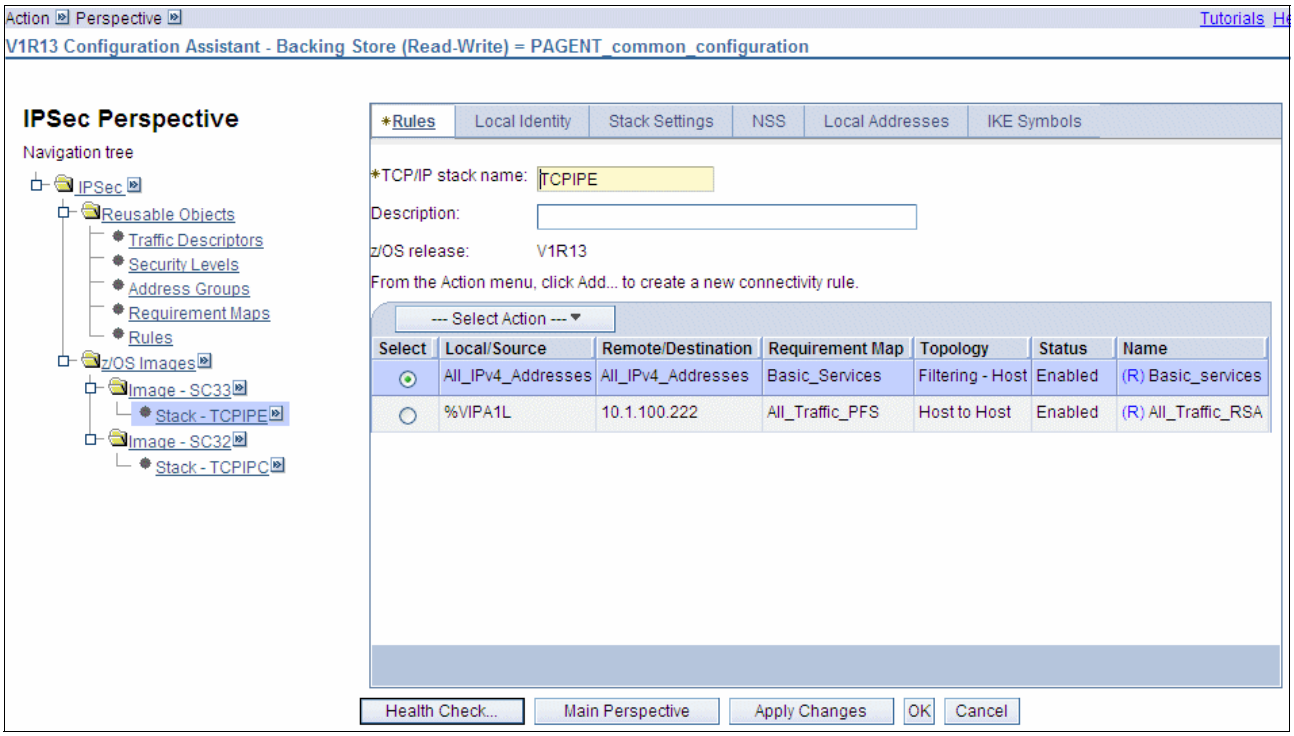


Figure 4-37 Basic_services rule is now reusable

4. We now assign this reusable rule to stack TCPIPC. From the IPSec perspective, select Stack-TCPIPC and click the Rules tab. Click **Select Action** → **Add**.

5. In the Welcome panel, select Reusable rule and choose Basic_services from the drop-down menu as shown in Figure 4-38. Click **Next**.

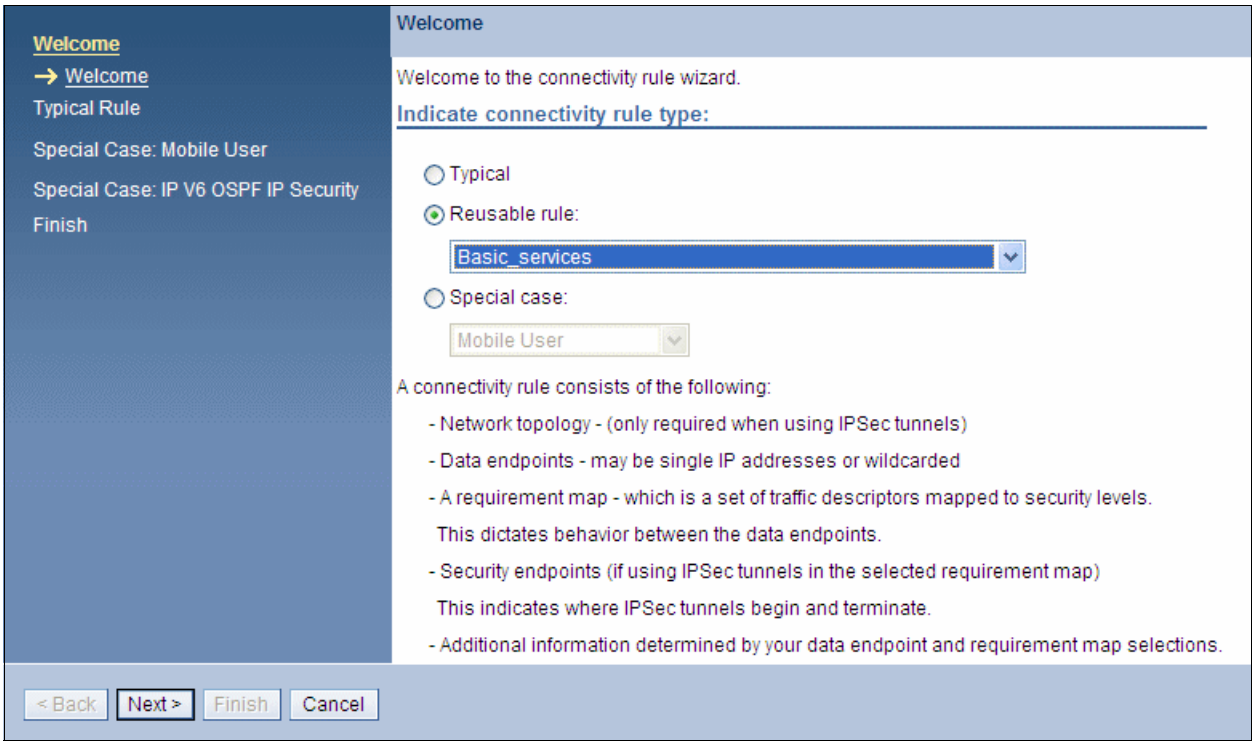


Figure 4-38 Choose Basic_services reusable rule

6. Review the Finish panel to ensure the information is correct. If you want to, at this point, you may change the connectivity rule name from the default of the reusable rule name. We let it default. Click **Finish**.
7. The Basic_services reusable rule is now assigned to stack TCPIPC as shown in Figure 4-39. Note the **(R)** showing that this is a reusable rule. Click **Apply Changes**.

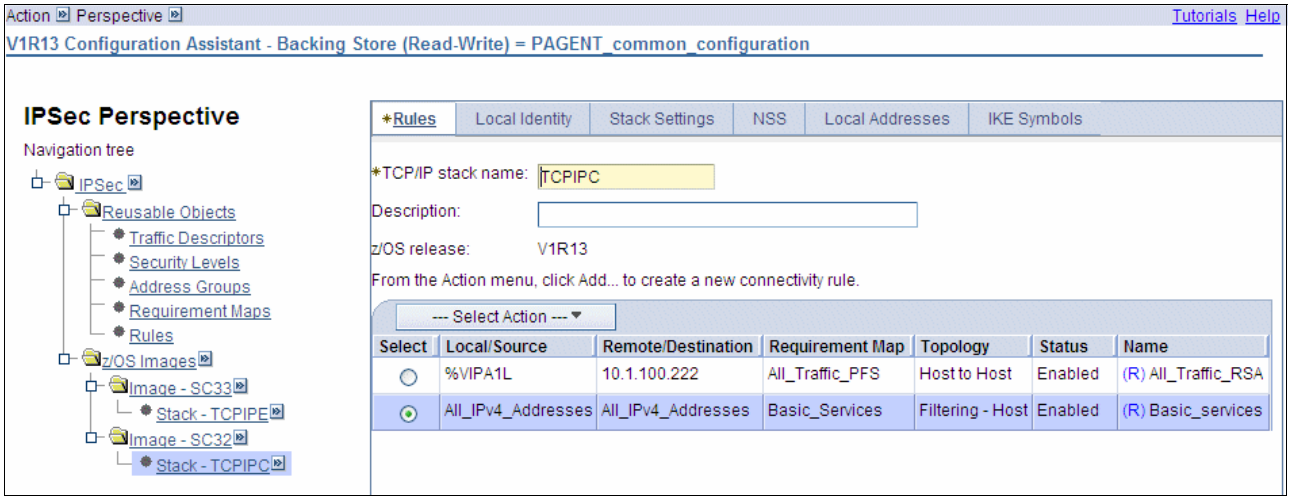


Figure 4-39 Basic_services reusable rule assigned to TCPIPC

Note: The order of the connectivity rules in a stack is important. When the policy is active and there is an attempt to send packets from or to the z/OS image, the first rule in the list is checked first. If there is a match with the definition of this first rule, it is used. If there is no match, the second rule is checked, and so on. Performance implications exist too. If possible, place your most used rules at the top of the list.

Inherent in the rules 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.

8. To move the Basic_services connectivity rule above the All_Traffic_RSA rule, select the Basic_services rule, click **Select Action** → **Move Up**.

4.5 Connection flooding

IDS provides protection against connection flooding across servers. This is not an attack type. The protection is always on; no configuration is needed. The number of servers under a potential connection flood is monitored on a per TCP/IP stack basis. If more than 25 servers are under a potential connection flood, no additional backlog-queue expansion is allowed. This prevents an unlimited number of servers from expanding to the 768 entry limit for a single server.

An individual server is considered under a potential connection flood when backlog-queue expansion is required to handle the incoming connection requests. A potential connection flood is detected is based on the initial size of the backlog queue. A small initial backlog queue (for example, 10 entries) is allowed to expand twice before the server is considered under flood. A server with a large initial backlog queue (for example, 500 entries) can expand once, up to a maximum of 768 entries, before it is considered under flood. A server with an initial backlog queue greater than or equal to 768 does not expand. A potential connection flood is detected when a connection request is received and the backlog queue is full.

Individual servers are allowed to expand up to the maximum value of 768 entries even if the server is considered under a potential connection-flood attack. A server's backlog expansion is only prevented if more than 25 servers are potentially under connection-flood attack.

While a server is being flooded, a high-water-mark count is kept of the number of backlog entries in use for half-open connections.

A server is no longer considered under a potential connection flood if either of these conditions is true:

- The number of entries in the server's backlog queue shrinks to half of the high-water-mark count.
- Forty percent or less of the server's backlog queue entries are in use for half-open connections.

The *ConnectionFlood* field in the Netstat ALL/–A report indicates whether the server is experiencing a potential connection-flood (Yes) or not (No).

The *ServersInConnFlood* field in the Netstat IDS/–k report shows the number of TCP servers under a potential connection flood.

4.6 Backup and migration considerations

This section has backup and migration considerations regarding the policy agent.

4.6.1 The backing store file

The Configuration Assistant keeps your configuration data in a file called the *backing store file*. You can manage separate sets of configuration information by keeping them in separate backing store files. Like all important data, you should back up all of your backing store files on a regular schedule.

You can create as many backing store files as needed. For example, you may want to have one backing store file for each data center's configuration information. Or you may want to have a backing store file for each technology across all of the data centers.

For the z/OSMF version of the Configuration Assistant, the backing store files are all stored on z/OS DASD and managed by the Configuration Assistant. If you migrate from a previous release of z/OSMF the Configuration Assistant automatically transfers the backing store files from the previous release.

The Configuration Assistant uses a simple file locking mechanism to ensure that two administrators are not operating on the same backing store file at the same time. When a backing store file is opened in the Configuration Assistant, a lock file is created that contains a lock ID value, and the date and time at which the lock file was created. For z/OSMF Configuration Assistant, the lock ID value is the z/OSMF logon user ID.

If you open a backing store file and the lock is already held, the panel shown in Figure 4-40 opens. From this dialog, you may choose to continue the session with the data in read-only mode, select a different backing store file, or cancel. The dialog also shows the contents of the lock file, so you can determine who has the backing store file in read-write mode, and when they obtained the lock.

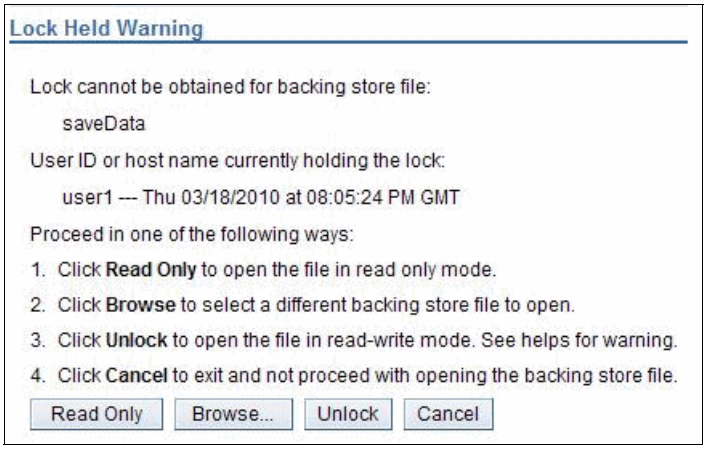


Figure 4-40 Lock held warning

Loss of configuration data: For the Windows version of the Configuration Assistant, you manage the backing store files and their locations. If you lose or destroy a backing store file, its configuration data will be lost. For the Windows version, the file is named `install_directory\files\saveData`. The `install_directory` is the directory and full path where the Configuration Assistant is installed.

The following instructions describe backing store management on z/OSMF Configuration Assistant. See Figure 4-41 on page 153.

- Creating a new backing store

  When you start the Configuration Assistant for the first time, the Configuration Assistant operates on the default backing store file. You may create a new backing store by selecting **Action** → **Open** → **Create a New Backing Store**. In this case, you are creating the configuration from scratch.

  You may also select **Action** → **Save As** to save the current backing store file, which you are working on as a new backing store file.

  Because all backing store files are stored on disk by the z/OSMF Configuration Assistant, you do not specify the file path to be stored; you only specify the file name.

- Saving to a current backing store

  Select **Action** → **Save** to save changes to the current backing store you are working on.

- Opening an existing backing store

  Select **Action** → **Open** to open an existing backing store file. A list of all existing backing store files, which z/OSMF manages, is displayed. Select the one you want to open.

- Deleting a backing store

  Select **Action** → **Delete** to delete the backing store files. A list of all existing backing store files, which z/OSMF manages, is displayed. Select the one you want to delete.

- Renaming a backing store

  There is no option to rename the file on z/OSMF Configuration Assistant, so open the existing backing store and save with new name with **Action** → **Save As** option. Then, delete the old backing store with **Action** → **Delete** option.

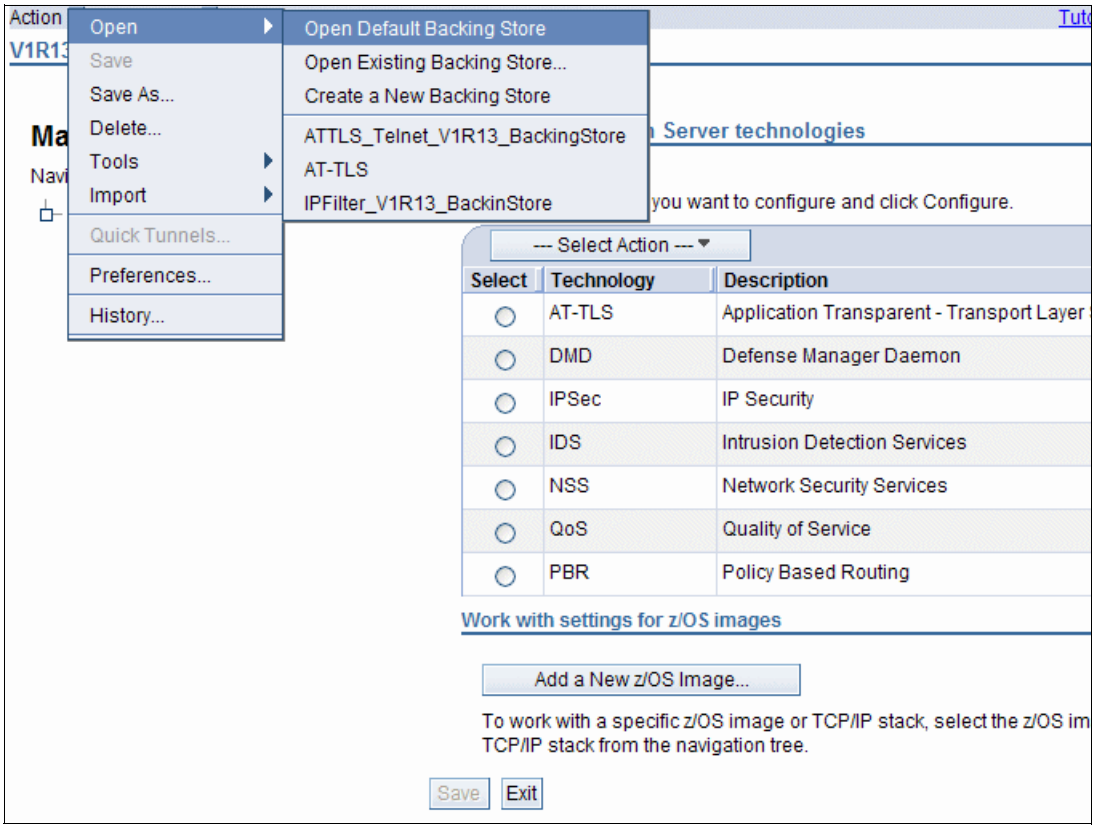


Figure 4-41 Backing store management

Backing store considerations: Import and copy

You might need to consider the use of multiple backing store files. For example, you might create a separate backing store file for the test configurations and production configurations. 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 Configuration Assistant. If the test with the merged production file is unsuccessful, you still have a backup of the production backing store. The Import function is described in 4.6.3, “Importing (merging) backing store files” on page 154.

Another way to accomplish the integration of a test policy into a production set of policies does not require the 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.

4.6.2 Migrating backing store files to z/OSMF Configuration Assistant

The Windows version of the Configuration Assistant allows you to store backing store files on your local drive, a LAN drive or on z/OS. Using the Configuration Assistant in the z/OSMF environment requires that existing backing store files be migrated into the z/OSMF environment. If you migrate from a previous release of z/OSMF the Configuration Assistant automatically transfers the backing store files from the previous release. If you need to migrate files from a previous release of the Windows version of the Configuration Assistant, complete the following steps. After the backing store files are transferred into z/OSMF, they will be known only by the file name. They will be placed in a directory managed by the Configuration Assistant and z/OSMF.

Instructions for migrating backing store files

Complete the following steps:

1. Ensure your backing store file is on z/OS DASD. If the backing store file is on your Windows local drive or LAN drive, use FTP to transfer the file to your z/OS system. If you do not know the location of your backing store file, click **File** → **Properties** from the Windows client to view its location.
2. Start z/OSMF Configuration Assistant.
3. Select **Action** → **Tools** → **Transfer Backing Store File to z/OSMF**. On the next panel that opens (Figure 4-42), specify the path and backing store file name that you have stored in the z/OS file system (in the previous step). Also specify the backing store file name to be used on z/OSMF. Click **Transfer**.

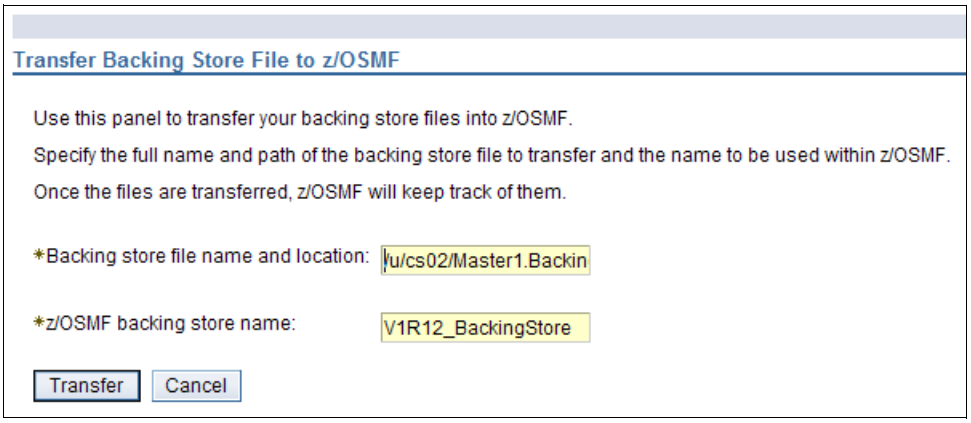


Figure 4-42 Transfer backing store

4. The result is show in the panel (Figure 4-43). Click **OK**.

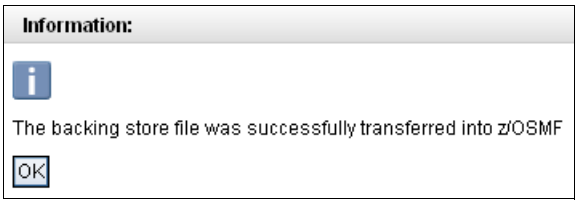


Figure 4-43 Backing store transfer complete

5. Now the transferred backing store file is managed by z/OSMF. You may open this backing store by selecting **Action** → **Tools** → **Open Existing Backing Store**, or you may merge it with other backing store file, as described in 4.6.3, “Importing (merging) backing store files” on page 154.

4.6.3 Importing (merging) backing store files

For various reasons, you might have multiple backing store files that contain separate policy configurations. For example, you might have an IDS administrator who keeps a version of the backing store file separate from the one containing QoS policies. During testing, you might have created separate AT-TLS policies for individual applications, such as IBM CICS® sockets, FTP, and TN3270. You might want to merge these separate backing store files.

The backing store import function reads an existing Configuration Assistant backing store file and combines the configuration data into the current configuration view. Reusable objects

and configuration entities such as images and stacks are adopted into the current view without change unless a name conflict occurs. In the case of a name conflict, decisions are made about whether the data can be combined at a lower level, or whether a rename of the import configuration object can allow the data to be adopted with only a name change.

Instructions for importing backing store files

Complete the following steps:

1. Start z/OSMF Configuration Assistant.
2. Open the backing store file that you want to import the data to (target backing store file) by selecting **Action** → **Open** → **Open Existing Backing Store**. Optionally, you may create a new backing store file by selecting **Action** → **Open** → **Create a New Backing Store**.
3. Select **Action** → **Import** → **Import Backing Store**. The panel, shown in Figure 4-44, opens. In the panel, all the backing store files that are managed on z/OSMF are listed. Select the backing store file that you want to import (source backing store file).

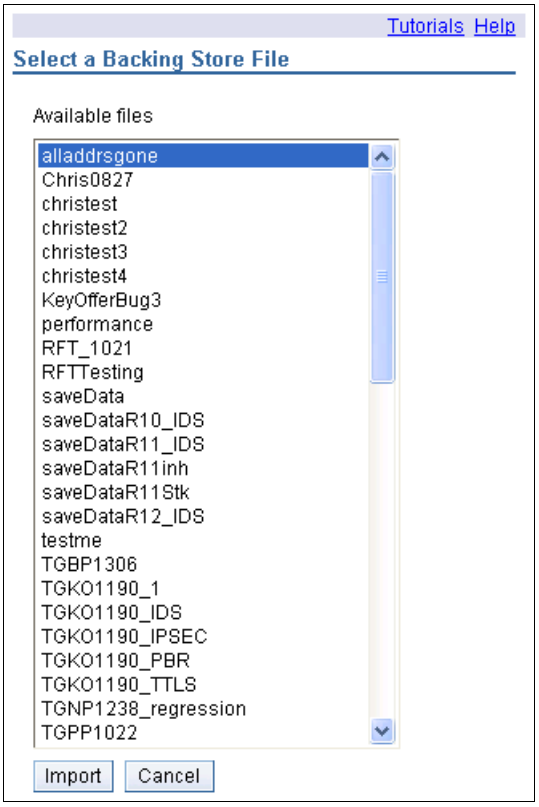


Figure 4-44 Import panel

4. When the import process completes, you receive the information message as shown in Figure 4-45, and the current configuration view contains all the combined data.

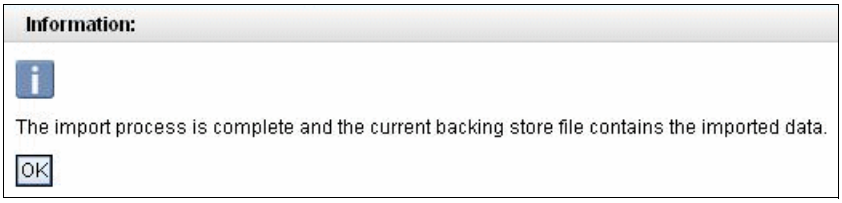


Figure 4-45 Backing store import complete

5. The administrator may review the combined data view and makes changes if necessary before saving the configuration. If the administrator does not save the combined configuration view, then nothing has been altered in the original configuration or that of the import configuration.
6. Save the merged backing store by selecting **Action** → **Save**. You may optionally leave a comment in the next panel.

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: This rule states that the image names and TCP/IP stack names should be consistent between files that are being merged. Consistency helps simplify the merging of backing store files. Backing store files are easy to merge if all policy administrators have agreed to name their MVS image the same within the 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 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, the same backing store files are merged as follows:

- Image SC33
  - TCPIPD
    - AT-TLS for TN3270
    - AT-TLS for FTP

To verify that the operation completes as expected, you can use the Import Activity Summary Report. It shows that many more of the reusable objects from the merged files are consolidated instead of repeated. With View Summary, you may view this information in another format.

4.6.4 Importing the policy file to Configuration Assistant

The Configuration Assistant produces configuration policy files used by the z/OS Communications Server Policy Agent. These files establish user policy for specific communications functions (such as IPSec, IDS, and others). With the Policy Data Import function, the Configuration Assistant reads policy data from an active Policy Agent and converts that data into Configuration Assistant objects.

There are two primary uses for the Policy Data Import function:

- The administrator has never used the Configuration Assistant but wants to initialize a Configuration Assistant configuration with data from existing policy files.
- The administrator uses and manages configuration data with the Configuration Assistant, but an emergency change was made to a policy file. The policy statement change must be incorporated into the primary backing store configuration data.

For both of these cases, the Configuration Assistant owns and manages the configuration data. The importing of policy data is performed only for the special cases noted previously. Making changes to the configuration by editing policy files is discouraged and should only be done in emergency situations.

Note: When creating or modifying policy file statements, do not add, modify, or delete policy statement names that contain a tilde character.

The Configuration Assistant sends the requests for import files and the policy agent needs to accept that request. Thus, the policy agent needs a `ServicesConnection` setup to accepting requests.

Preparing PAGENT for configuration file import requests

To allow Configuration Assistant to import the policy agent configuration files or the active configuration for any given TCP/IP stack, complete the following steps:

1. In the policy agent main configuration file, define a port and the TCP/IP stack name to which the Configuration Assistant will connect.

   The `ServicesConnection` statement in the main configuration file provides the port and TCP/IP stack name on which the policy agent listens for remote connections. The policy agent listens for services requestor connections on only one TCP/IP stack. In a single stack (INET) environment, the policy agent uses the active TCP/IP image to listen for services connection requests. In a multiple stack (CINET) environment, specify the name of the TCP/IP stack (**1**), or the default stack will be used. The default port value is 16311. Example 4-21 shows the configuration example for PAGENT.

   Example 4-21 PAGENT main configuration file /etc/pagent.sc33.conf

```
ServicesConnection
{
   Port                          16311
   ImageName                     TCPIPD 1
}
```

   Optionally, configure secure connections from the import requestors. By default, the `ServicesConnection` statement defines a basic unsecure connection. You can define a secure connection instead. When the secure connection is specified, you also have to define which TLS/SSL key ring to use. When you configure a secure connection, policy agent creates an AT-TLS policy for the service connection automatically. Thus, you also have to specify in Configuration Assistant that the connection is to be secured.

   You must enable the TTLS parameter on the TCPCONFIG statement in the TCP/IP profile for the generated AT-TLS policy to be effective.

   If ServicesConnection is configured, the policy agent will issue the messages shown in Example 4-22 to indicate that policy agent is ready to accept import requests.

   Example 4-22 Policy Agent message shows it is ready for accepting requests

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
...
EZD1576I PAGENT IS READY FOR SERVICES CONNECTION REQUESTS
```

   Example 4-23 on page 158 shows that TCPIPD stack is listening on port 16311 **3**.

Example 4-23 Policy Agent is listening on port 16311 of TCPIPD stack

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS V1R13 TCPIPD 609
USER ID  CONN     STATE
PAGENT   00000B75 LISTEN
  LOCAL SOCKET:   ::..16311  3
  FOREIGN SOCKET: ::..0
...
```

2. Use the z/OS system security facility (for example, RACF) to specify permissions for system access to policy technologies. Give READ access to SERVAUTH `EZB.PAGENT.sysname.tcpname.technologyname` to the user who can import the policy file. For `technologyname`, specify `IDS`, `IPSEC`, `ROUTING`, `TTLS`, or asterisk (`*`) to allow them all (2). In our environment, we specify the permission to CS02 user, as shown in Example 4-24.

Example 4-24 Permission to import policy

```
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST(SERVAUTH)
SETROPTS GENERIC(SERVAUTH)
RDEFINE SERVAUTH EZB.PAGENT.SC33.TCPIPD.* UACC(NONE)
PERMIT EZB.PAGENT.SC33.TCPIPD.* CLASS(SERVAUTH) ID(CS02) ACCESS(READ) 2
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Instructions to import policy to z/OSMF Configuration Assistant

The z/OSMF Configuration Assistant can request existing configuration files to be imported for further changes and additions. To use this service, the Configuration Assistant acts as a import requestor and depends on the configured policy agent to accept the policy agent configuration file import service. Complete the following steps:

1. Connect to z/OSMF Configuration Assistant, and open the backing store file that you want to import the policy to (target backing store file).
2. In the Main Perspective panel, click **File** → **Import** → **Import Policy Data**.
3. In the Import Policy Data panel (Figure 4-46 on page 159), specify the technology to import, host connection information (on which policy agent accepts the request), and the user name and password that has security authorization. Click **Go**.

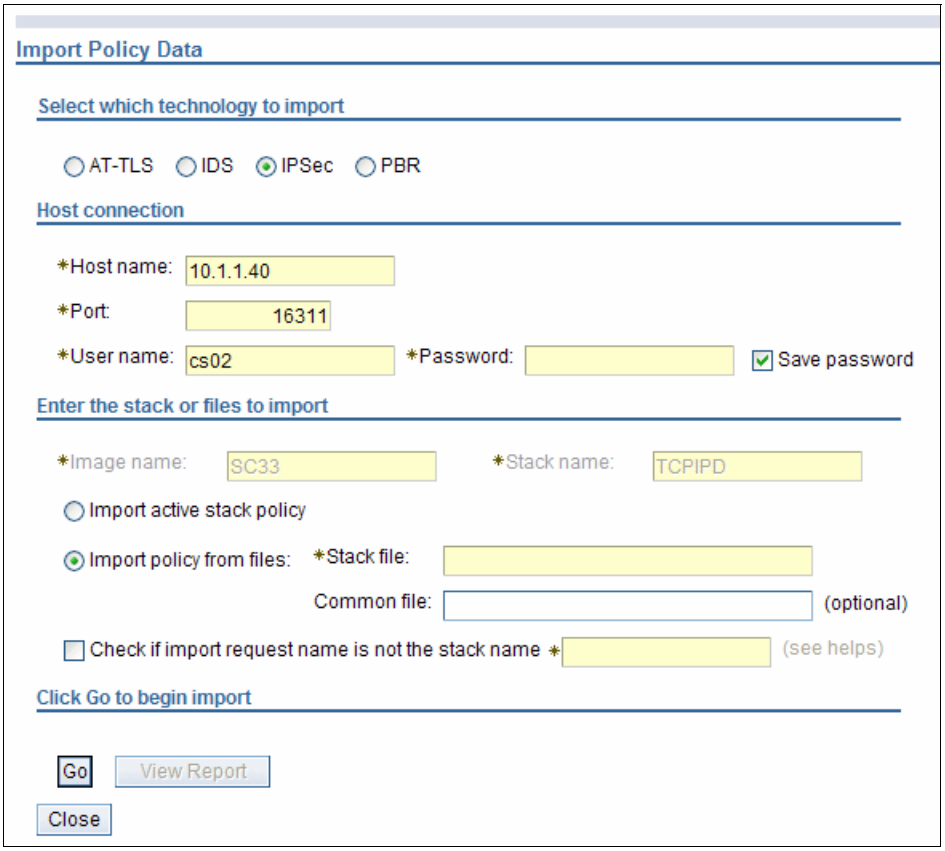


Figure 4-46 Import Policy data panel

4. Save the imported data on the backing store file by selecting **Action** → **Save**.

4.7 Additional information

For additional information about PAGENT, see the following resources:

- *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.

5

Central Policy Server

In this chapter, we explain how a policy agent can become a Central Policy Server 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.

We discuss the following topics in this chapter.

Section	Topic
5.1, “Background” on page 162	Introduces the policy agent
5.2, “Basic concepts” on page 164	Basic concepts behind a centralized or distributed policy service
5.3, “Configuring distributed (centralized) policy services” on page 166	Implementation tasks for configuring the policy services at the server and the client nodes
5.4, “Activating and verifying the policy services environment” on page 188	Pertinent commands and messages used to verify correct operation of policy services
5.5, “Diagnosing the centralized policy services environment” on page 193	Pertinent displays and manuals that are helpful in diagnosing policy services problems
5.6, “Configuring the Central Policy Server without SSL Security” on page 195	Tasks to eliminate SSL security on the connections between the Central Policy Server and a client
5.7, “Additional information” on page 197	References to additional information to help you configure policy agent and Central Policy Services

5.1 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 certain 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 4, “Policy agent” on page 103, policy-based networking is a way of accomplishing a set of network goals through the use of policy definitions. For example, one network goal might 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 5-1, 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
- Intrusion detection system (IDS)
- IPSec
- Routing or policy-based routing (PBR)

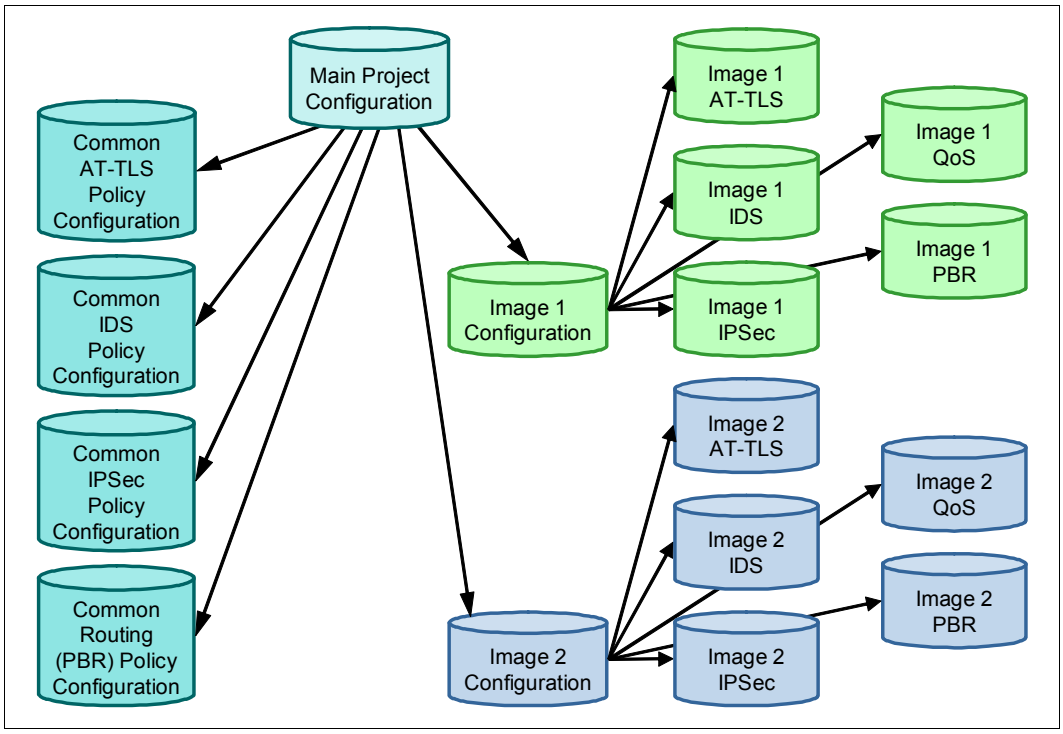


Figure 5-1 Hierarchy of PAGENT configuration files

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 policy types (AT-TLS, IPSec, IDS, QoS, and Routing or 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.

5.2 Basic concepts

The problem being solved by centralized policy servers is primarily one of policy management. The policy agent configuration shown in Figure 5-1 on page 163 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 Configuration Assistant needs connectivity to the policy server only if no local policies are defined on the policy clients. Figure 5-2 shows an overview of the distributed (also known as centralized) policy services solution.

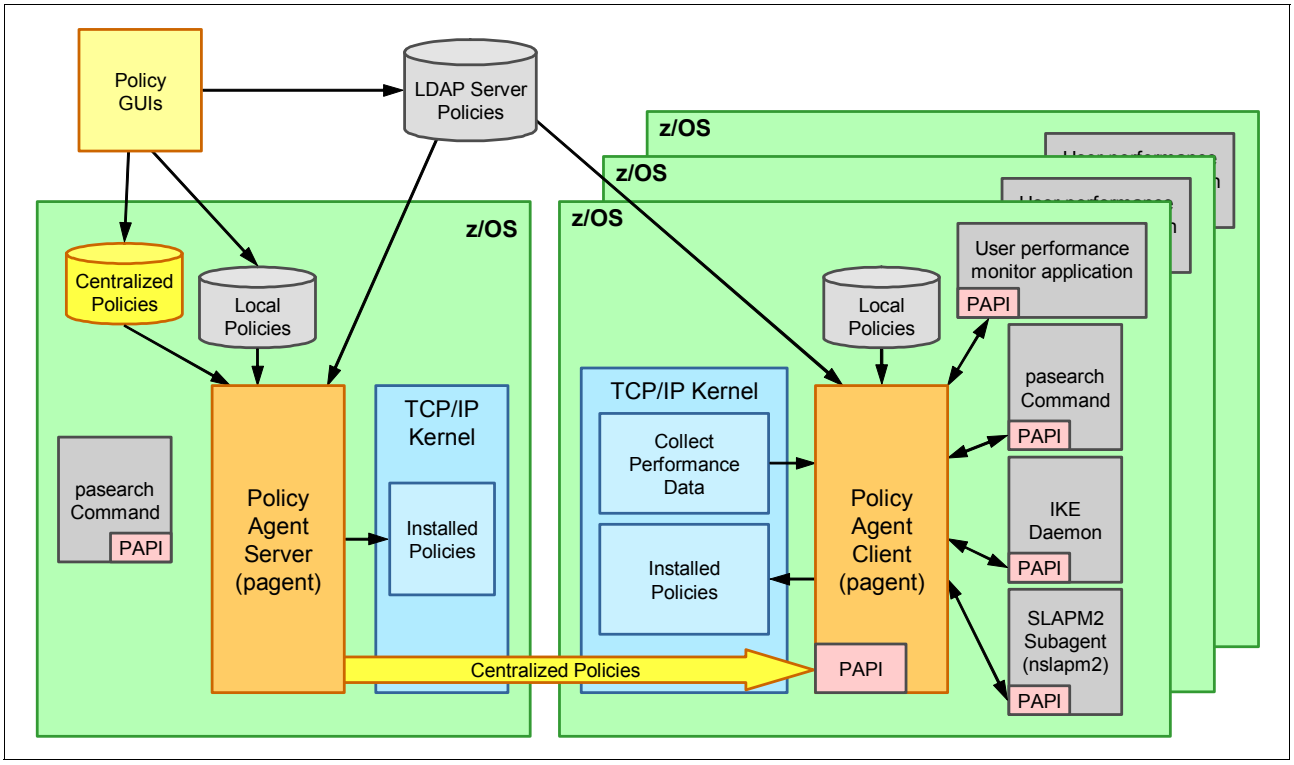


Figure 5-2 The Centralized Policy Services solution

On the right side of Figure 5-2 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 you see the policy agent 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 connections between the policy server and its clients can be optionally secured with SSL/TLS. The 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 Configuration Assistant must have a connection to only 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
- 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 you can implement secure, long-running connections between the policy clients and the policy server. If an SSL/TLS connection is desirable between the server and the client, the policy server negotiates the connection with the help of local AT-TLS policies. However, you cannot use AT-TLS policies on the policy client. If the AT-TLS policies reside on the policy server, the policy client will need to connect to the policy server to obtain the policies that secure that connection. For this reason, the policy clients are configured as SSL clients, using local definitions in the image configuration files.

5.3 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.

Tip: The tasks, examples, and references in this chapter are based on the configuration shown in Figure 5-3.

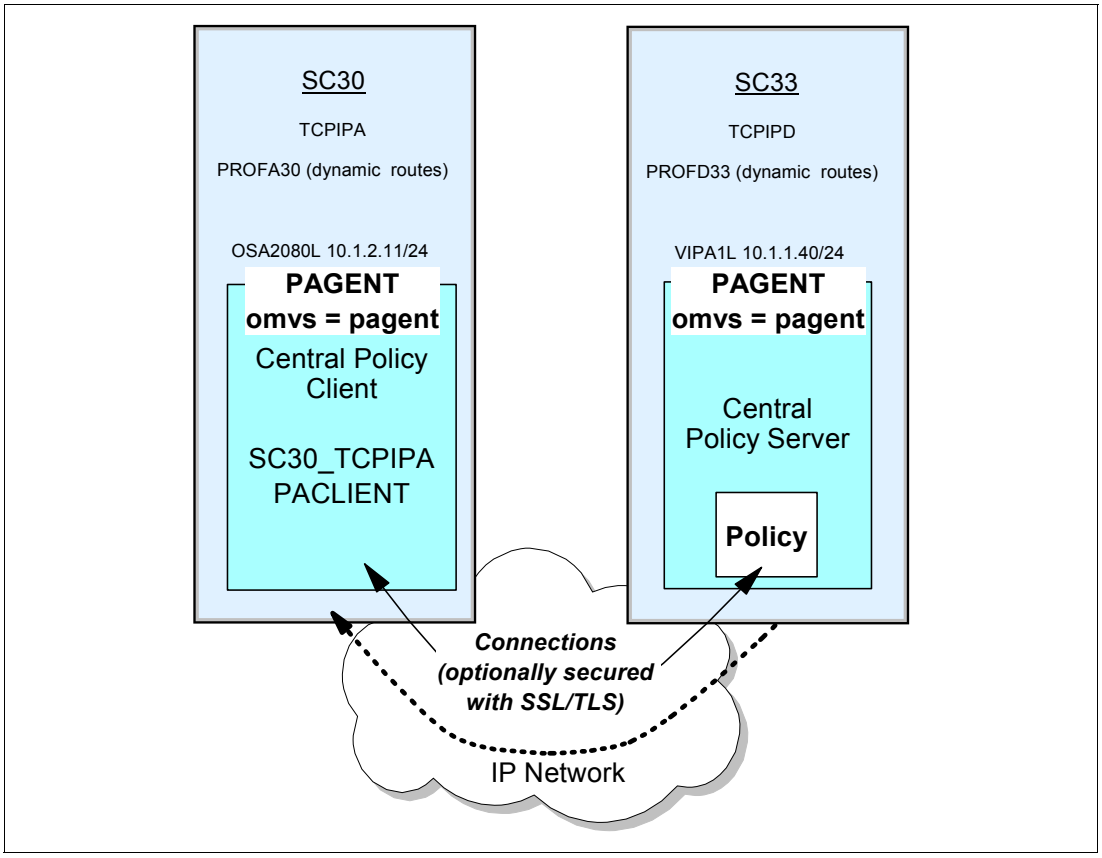


Figure 5-3 Relationship between central policy server and policy client

In Figure 5-3, the example server has an z/OS system ID of `SC33` and the policy client has an z/OS system ID of `SC30`. The name of the policy agent procedure at each z/OS 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 that is associated with the policy client is `PACLIENT`. This information is required to establish the RACF environment for the policy server and policy client relationship.

Although optional, we implemented SSL security in our test scenario. However, in 5.6, “Configuring the Central Policy Server without SSL Security” on page 195, we show that the Central Policy Server operations are still functional if the client-server establishes connections without SSL.

5.3.1 Configuring the base environment with SSL

The base environment for distributed policy services (centralized policy services) has up to four prerequisites:

- An existing policy agent (required)
- A set of policies we want to distribute to clients (required)
- A RACF environment that authenticates the user (required):
  - 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
- A server certificate, either a private USER certificate or a SITE certificate with the appropriate key ring authorizations (optional)

First, a working, existing policy agent is needed. See Chapter 4, “Policy agent” on page 103 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, see other chapters in this book.

Figure 5-4 illustrates the types of policies that a central policy server can distribute to its clients. Where possible, both common and image-specific policy configurations are distributed to clients. Note that there is no Common Policy category for the QoS policy type.

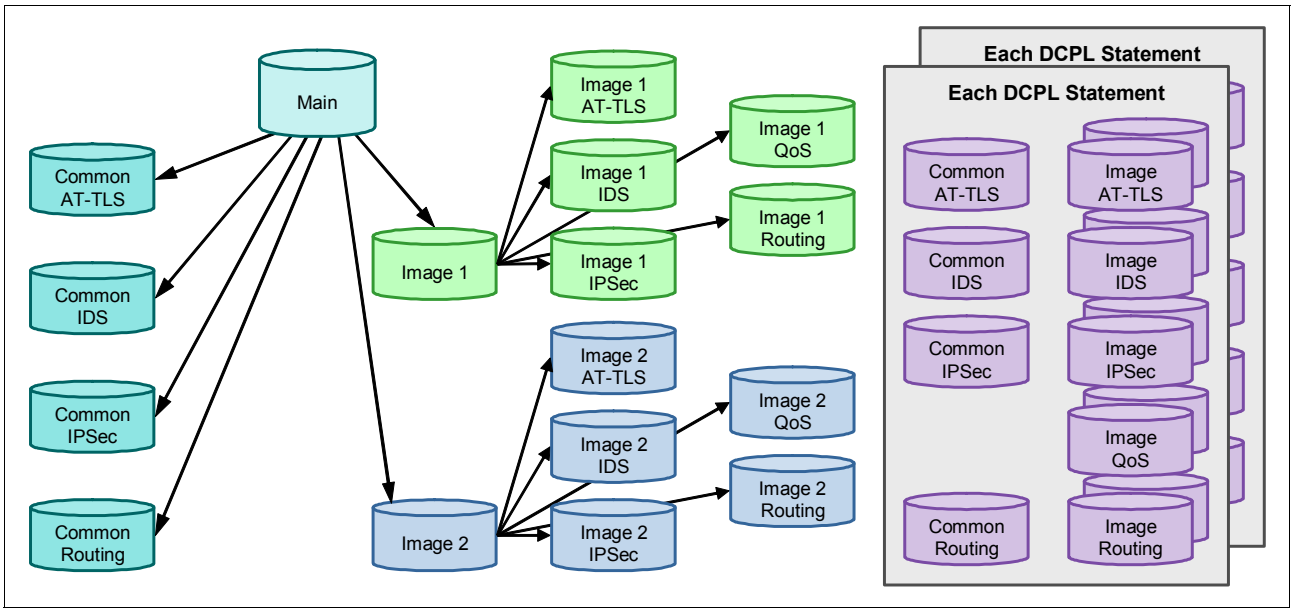


Figure 5-4 DynamicConfigPolicyLoad (DCPL): What central policy server distributes to policy clients

Each policy client needs a RACF identity for our policy clients. In our test environment, we did not use a client SSL certificate for client authentication. Instead, we used a user ID and either a password or a pass ticket. We explain how to establish a user ID and password in “RACF user ID and password for the policy client” on page 170. To learn how to set up the pass ticket environment, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

We also chose to secure the connections between the policy server and its clients using a key ring and certificate environment to support SSL. With this secured connection, a server certificate is required. The server certificate can be either a user certificate or a site

certificate. We already had a site certificate available that was tested with both the TN3270 server and the FTP server. Thus, we used the same certificate and associated PAGENT with that certificate.

Tip: If your enterprise needs more stringent security controls, you might not want to employ the shared *site* certificate for the policy server. Instead, you might want to create a new *user* certificate that is distinct from the other server certificates. If your enterprise does not need stringent controls, you might not decide to implement SSL connections between clients and server.

For more information about certificate management, see Chapter 3, “Certificate management in z/OS” on page 39. You can read about how we created the site certificate and the certificate authority certificate in 17.3.2, “Configuration of FTP native TLS security” on page 726. 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 if you choose to implement SSL/TLS. The certificate definitions needed for the policy server are described in “Certificate management for central policy services” on page 168.

Certificate management for central policy services

Example 5-1 shows the commands 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) to put the new commands (6** 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 5-1 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(20010-09-11))
KEYUSAGE(CERTSIGN) WITHLABEL(‘CS1A ITSO CA1')
SETROPTS RACLIST(FACILITY) REFRESH
RACDCERT CERTAUTH LIST

4
RACDCERT SITE GENCERT  SUBJECTSDN(CN('ITSO.IBM.COM') OU('ITSO CS1A Shared Site')
C('US')) WITHLABEL('CS1A ITSO SharedSite1') SIGNWITH(CERTAUTH LABEL(‘CS1A ITSO
CA1’)
SETROPTS RACLIST(FACILITY) REFRESH
RACDCERT SITE LIST

5
RACDCERT ID(TCPIP) CONNECT(CERTAUTH LABEL('CS1A ITSO CA1’)RING(SharedRing1)
USAGE(CERTAUTH)
```

```
RACDCERT ID(TCPIP) CONNECT(SITE LABEL('CS1A 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
```

Example 5-1 on page 168 takes advantage of many of the steps that 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.

The following list is a review of the steps in Example 5-1 on page 168:

1. It is unnecessary to perform this step again. It was executed in 17.3.2, “Configuration of FTP native TLS security” on page 726 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 executed in 17.3.2, “Configuration of FTP native TLS security” on page 726 when we created the shared key ring for our installation.

3. It is unnecessary to perform this step again. It was executed in 17.3.2, “Configuration of FTP native TLS security” on page 726 when we created the certificate authority certificate.

4. It is unnecessary to perform this step again. It was executed in 17.3.2, “Configuration of FTP native TLS security” on page 726 when we created the server SITE certificate.

5. It is unnecessary to perform this step again. It was executed in 17.3.2, “Configuration of FTP native TLS security” on page 726 when we connected the two certificates (CA and server SITE) to the shared key ring 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:

- 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 170.

- 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 z/OS

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 in Example 5-6 on page 174.

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. Example 5-2 shows the JCL job stream that we used to create this user ID.

Example 5-2 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 can have its own unique user ID and password, or you can 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.

Tip: If you want to use passticket authentication instead of a password, see *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 5-3.

Example 5-3 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 z/OS system name of the z/OS node. In this case, it is the system name of the z/OS 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 that 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 5-4 in case we needed this granularity later.

Example 5-4 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

Example 5-5 shows the permissions that we granted to the policy client user ID. Although not required, it does provide additional security in the z/OS UNIX environment.

Example 5-5 BPX.DAEMON authorization for policy client user ID

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(PACLIENT) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

5.3.2 Configuring the policy server

We configured the policy server environment for the simple scenario shown in Figure 5-5.

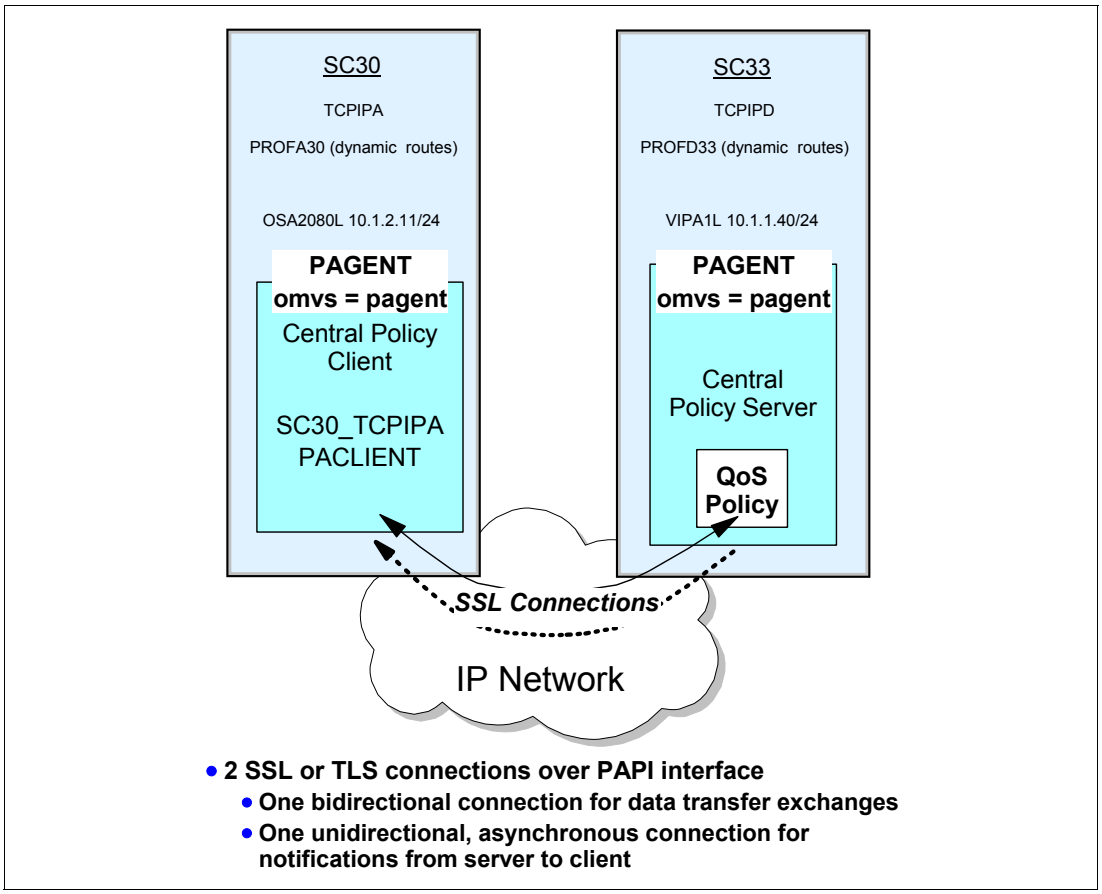


Figure 5-5 QoS policy maintained at central policy server and sent to policy client

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.

Tip: 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.

Example 5-6 shows the configuration of the policy server file that resides on z/OS System ID of SC33. The main configuration file for PAGENT on the LPAR contains image statement for all SC33 TCPIP stacks (a). TCPIPD (b) is to be the centralized (or distributed) policy server.

Example 5-6 pagent.sc33.conf at SC33

```
# LogLevel Statement
  Loglevel 15

# TcpImage and PEPInstance Statements (synonyms)
  TcpImage TCPIPA /etc/pagent.sc33.tcpipa.conf  FLUSH PURGE 600     a
  TcpImage TCPIPB /etc/pagent.sc33.tcpipb.conf  FLUSH PURGE 600     a
  TcpImage TCPIPC /etc/pagent.sc33.tcpipc.conf  FLUSH PURGE 600     a
  TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600     b
  TcpImage TCPIPE /etc/pagent.sc33.tcpipe.conf  FLUSH PURGE 600     a
#
# ClientConnection Statement
#   The Policy Agent acting as a policy server uses the ClientConnection
#   statement to specify the listening port.  The Policy Agent acting as
#   policy client uses this connection to retrieve remote policies.
      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/pagent.sc33.tcpipd.qos.polsvr.policy     g
        }
#     }
```

When we first set up the PAGENT environment for TCPIPD at SC33, we copied the sample for the main policy file from `/usr/lpp/tcpip/samples/pagent.conf` into a file named `/etc/`*`pagent.sc33.conf`* on SC33. 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 *pagent.sc33.conf* was already successfully tested at this point and was being used for all TCP/IP images on SC33. Recall that the main configuration file is coded manually: the Configuration Assistant is not employed to create this file.

We then 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 5-6.

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). The DynamicConfigPolicyLoad statements, d and e, identify the clients that are authorized to

retrieve policies from the server, 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 Examples Example 5-3 on page 172 and Example 5-4 on page 172. 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. The 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>`.

For more detailed information about variable string, see “Creating variable strings for the DynamicConfigPolicyLoad statement” on page 197.

Now to resume with our explanation on policy server configuration. 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 allowed the client to retrieve the QoS policy (f) that is loaded from the location `etc/pagent.sc33.tcpipd.qos.polsvr.policy` g.

The QoS policy file named `/etc/pagent.sc33.tcpipd.qos.polsvr.policy` was created earlier with the Configuration Assistant and is shown in Example 5-7.

Example 5-7 Excerpts from pagent.sc33.tcpipd.qos.polsvr.policy

```
## QoS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
.....................................................................Lines deleted
#####################################################################
# PolicyRule statements
#####################################################################

policyRule 0~1
{
```

```
   PolicyRulePriority        65000
   SourcePortRange           16310
   DestinationPortRange      1024-65535
   ProtocolNumberRange       6
   PolicyActionReference     action~1
}
.....................................................................Lines deleted
#########################################################################
# PolicyAction Statements
#########################################################################

PolicyAction action~1
{
   PolicyScope               DataTraffic
   OutgoingTOS               11100000
}
.....................................................................Lines deleted
```

The QoS policy in Example 5-7 on page 175 is loaded into both TCPIPD on SC33 and TCPIPA on SC30. The Central Policy Server treats this policy as a local policy for TCPIPD (SC33): it is a distributed policy for TCPIPA (SC30).

Configure the AT-TLS policy rules for the central policy server

We use SSL between the client and server. The server needs a policy to establish the SSL environment. The client does not need a policy: the client implicitly requests SSL services through a definition in the policy client configuration file.

We have working AT-TLS rules for several environments. Therefore, we have already enabled AT-TLS in the TCPIPD stack on z/OS SC33. See Chapter 12, “Application Transparent Transport Layer Security” on page 551 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 Configuration Assistant. We initialize the GUI from our workstation, and then in the Main Perspective panel, 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.

To configure policy rules, complete the following steps:

1. In the z/OSMF Configuration Assistant, click **Action** → **Open** → **Create a New Backing Store**.
2. Enter the name of the new backing store (using the Host file location as the repository as defined in the preferences of the Configuration Assistant) and click **OK**.

   After creating the backing store file on the mainframe, the lock on the existing file with which you opened the Configuration Assistant is released, and you are now working on the new backing store file. You are returned to the Main Perspective panel of the z/OSMF Configuration Assistant.

Tip: In our testing, 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 could later import into the production backing store file. We discuss the import function in 4.6.2, “Migrating backing store files to z/OSMF Configuration Assistant” on page 153.

You can also test the AT-TLS connection by copying the existing backing store file into a backup file and then adding the new definitions into the production backing store. With this method, you do not need to perform the import function. For more information, see “Backing store considerations: Import and copy” on page 153.

3. On the Main Perspective panel, select **Add a New z/OS image**. We name the image `SC33`, which was the z/OS system name, and then click **OK**.

Tip: Merging definitions is easier if the z/OS 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. For more details, see the discussion of this topic in Chapter 4, “Policy agent” on page 103.

4. Next, you are prompted to add a TCP/IP stack. Click **Yes**. Type in the TCP/IP stack name (TCPIPD in our environment), then click **OK**. Then select **AT-TLS** from the z/OS Communication Server technologies section, and click **Select Action** → **Enable**. The status column then shows `Incomplete` (Figure 5-6) because you need to configure the policy. Click the **Select Action** → **Configure** to start the configuration.

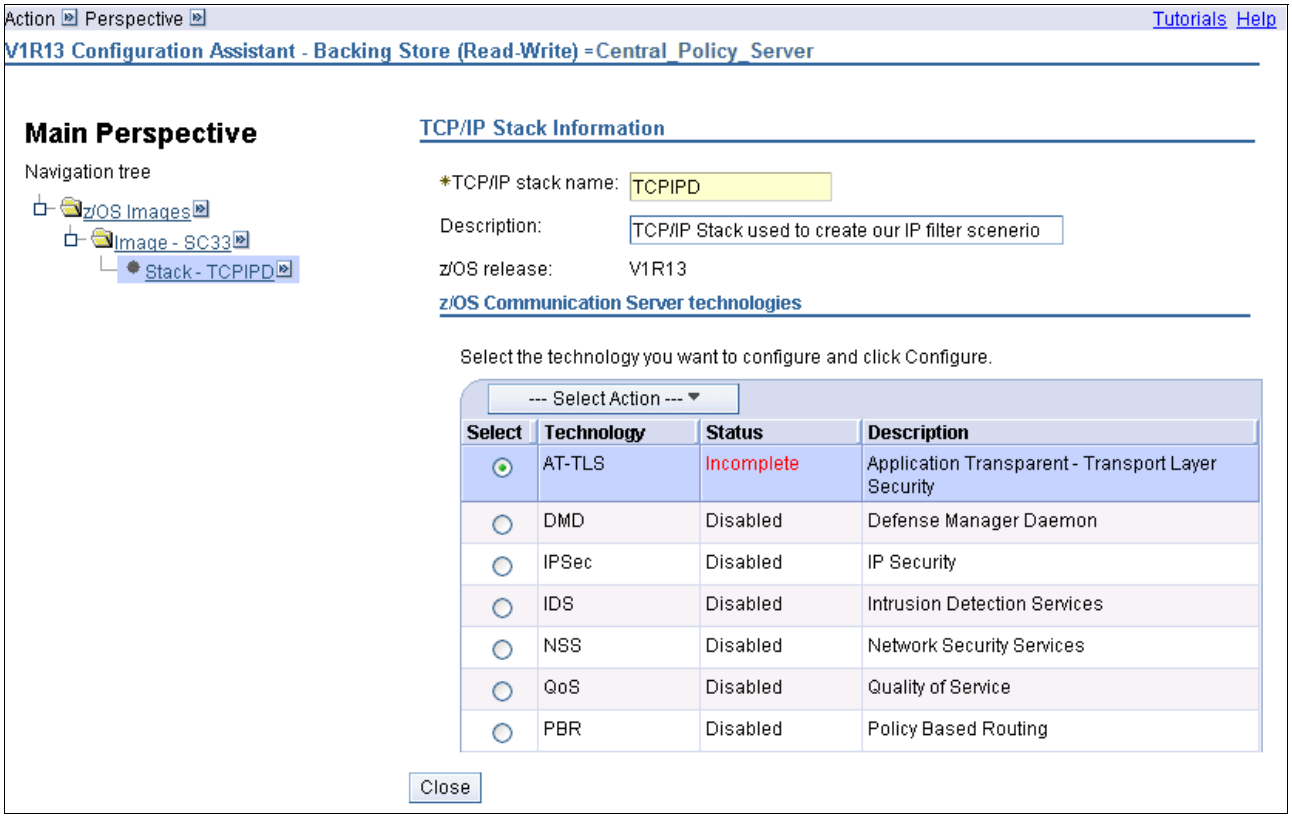


Figure 5-6 Enabling AT-TLS

Tip: If you view the existing rules on this panel, you can see an existing, disabled rule called Default_Central_PolicySvr. You can select this rule and then click **Modify** as an alternative to creating one from scratch as we will demonstrate in the following steps.

5. Click **Select Action** → **Add** to see the New Connectivity Rule Welcome panel. Click **Next**. In this example, we create a rule named `CentralPolicyRule` and select **Address Group** - **all_IPv4_addresses**, both for local and remote endpoints, as shown in Figure 5-7. Click **Next**.

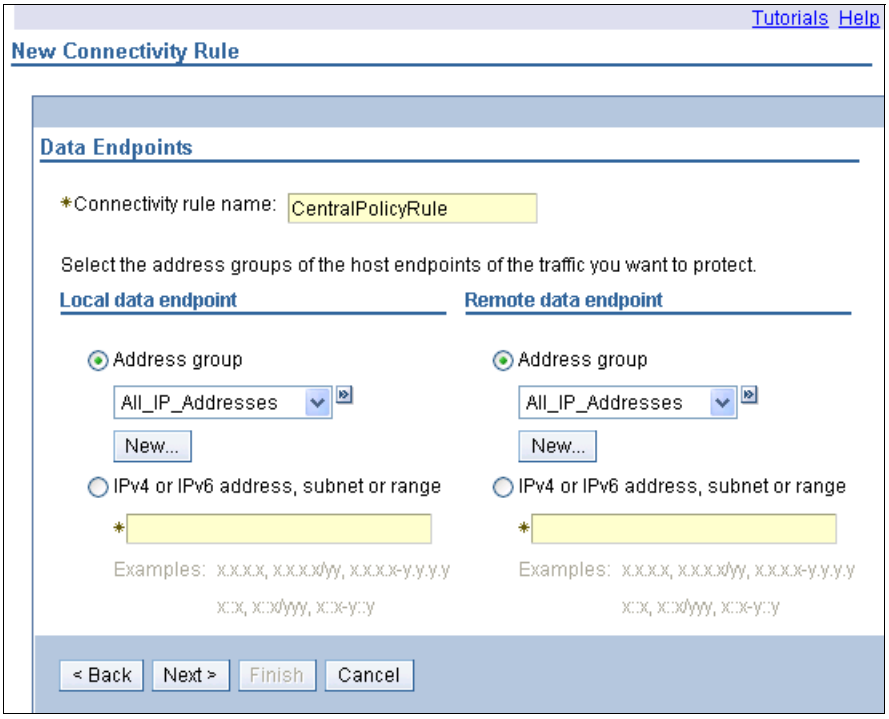


Figure 5-7 New Connectivity Rule - Data Endpoints panel

6. Select Requirement Map panel opens, as shown in Figure 5-8. Keep **Create a new requirement map** selected. Enter the new requirement map name (we use `CentralPolicyServerReq`). From the Traffic Descriptor pull- down menu, select **Centralized_Policy_Server**. For security level, we chose **AT-TLS_Silver**.

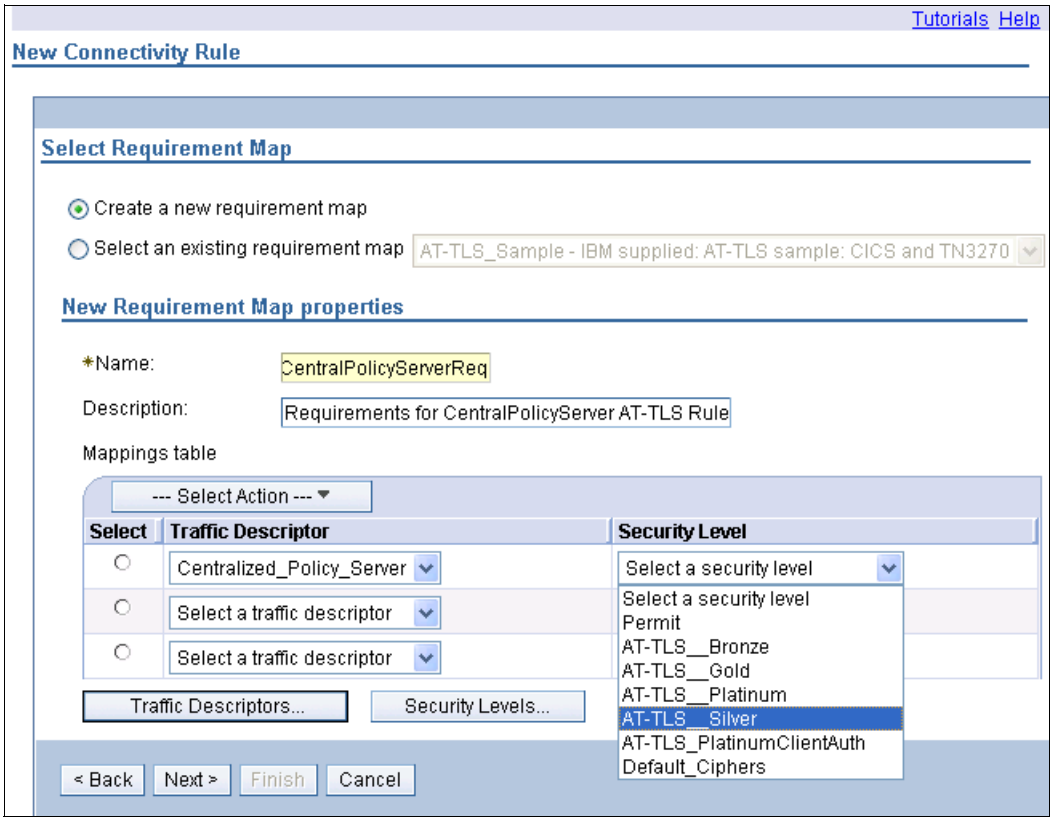


Figure 5-8 New Connectivity Rule - Select Requirement Map panel

7. Select **Action** → **View Details** to examine the traffic descriptor made available by the z/OSMF Configuration Assistant (Figure 5-9). Click **Close**.

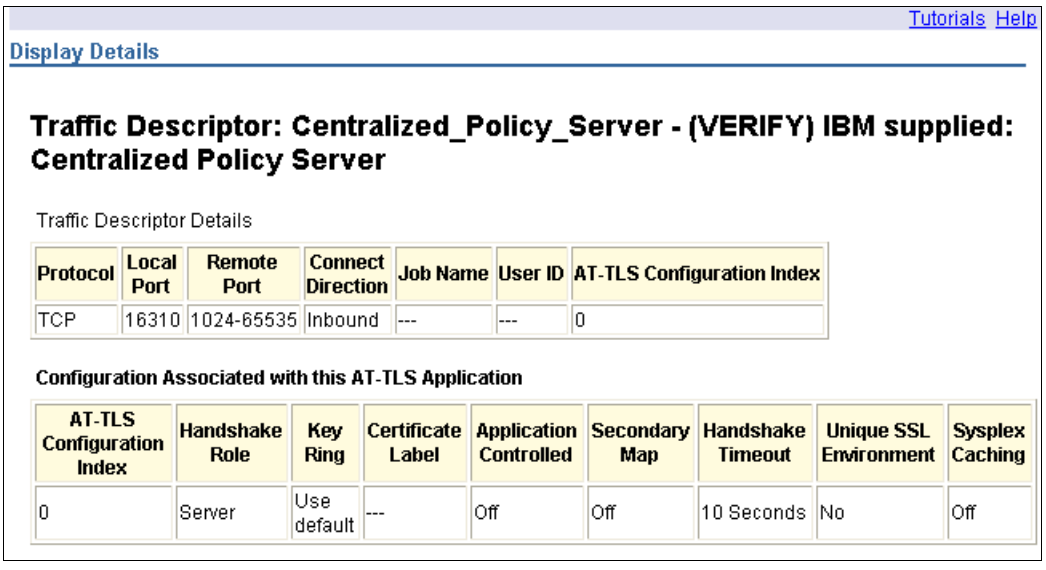


Figure 5-9 Traffic descriptor details using Configuration Assistant help

8. Click **Next** to move to the Advanced settings panel. You do not need to change anything on this panel. Click **Finish**.
9. You are returned to the AT-TLS Perspective main panel for the TCPIPD stack. The newly created rule, `CentralPolicyRule`, and the requirement map, `CentralPolicyServerReq`, can be seen in Figure 5-10.

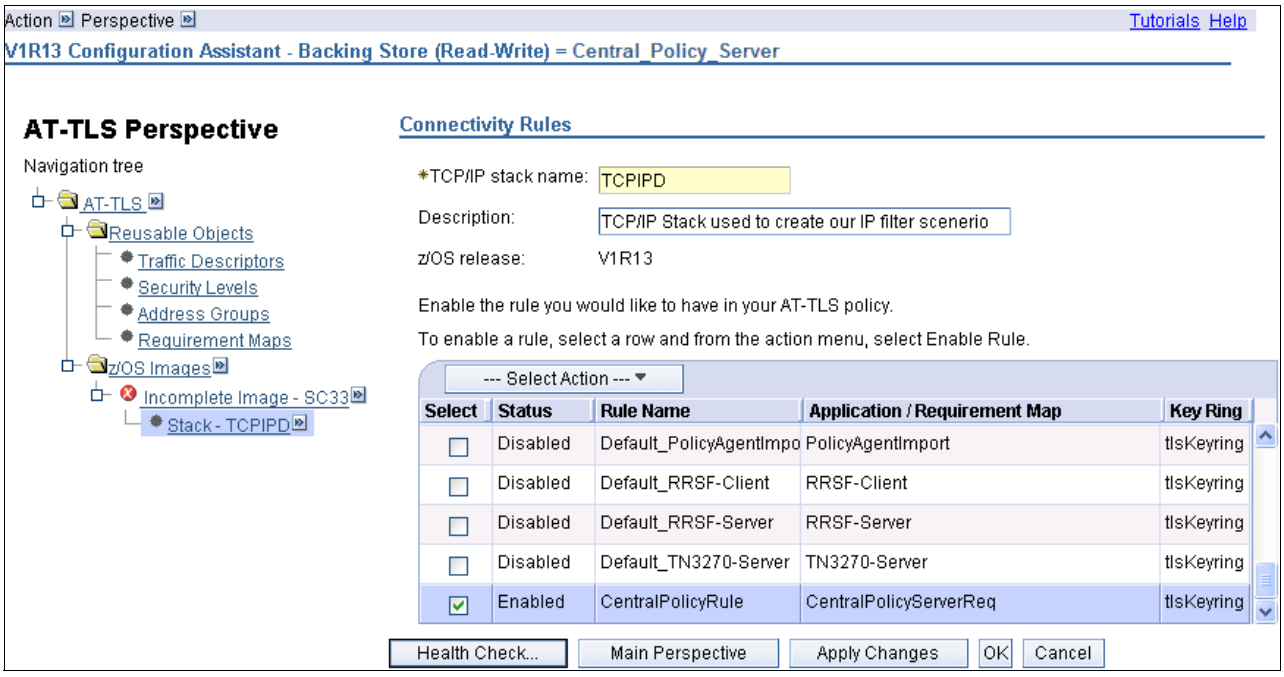


Figure 5-10 AT-TLS Perspective panel

10. Optional: While your newly created rule is selected as in Figure 5-10, you can click **Select Action** → **Modify**, select the Advanced tab, and then click **Advanced settings**. The Advanced Connectivity Rules panel opens, as shown in Figure 5-11. From here you can click **Select Action** → **Settings** for the selected traffic descriptor so that you can activate or alter tracing and other advanced parameters. These functions can be helpful when directed to enable tracing when directed by IBM Remote Technical Support.

Figure 5-11 Advanced Connectivity Rule Settings panel

As shown in Figure 5-10, the status of the new policy is now `Enabled`. In this panel, you can click **Health Check** and view the results. With no errors in the policy, it should work at the mainframe central policy server site. Click **Close**.

11. Note that in Figure 5-10 on page 180 our z/OS image displays as `Incomplete`. To correct this, click **Apply Changes** first and then select the z/OS image in the Navigation tree. You should be prompted with the panel shown in Figure 5-12. It is doubtful that you would want to use the default key ring name, so the Configuration Assistant is prompting you to confirm the setting before leaving the TCPIP stack configuration environment. Select **Settings** tab, and we enter the user ID (TCPIP) and ring name for our environment and then click **OK**.

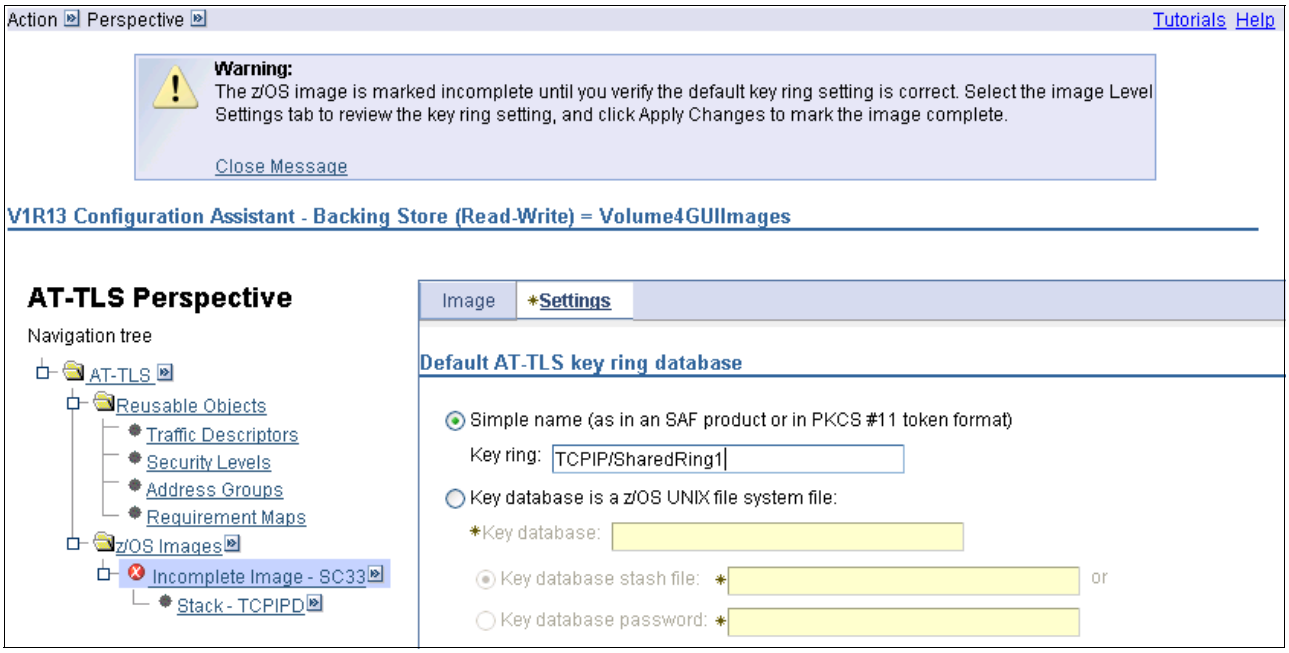


Figure 5-12 Required AT-TLS Image Level Settings

12. The `Incomplete` status for the z/OS image clears, signalling that the z/OS image definition is now complete. Click the small symbol to the right of z/OS image name, and select **Install Configuration Files**.
13. On the List of Configuration Files for Stack TCPIPD panel (Figure 5-13), select the policy and click **Select Action** → **Install**. Select the policy that you want to transfer, and click **Select Action** → **Install**.

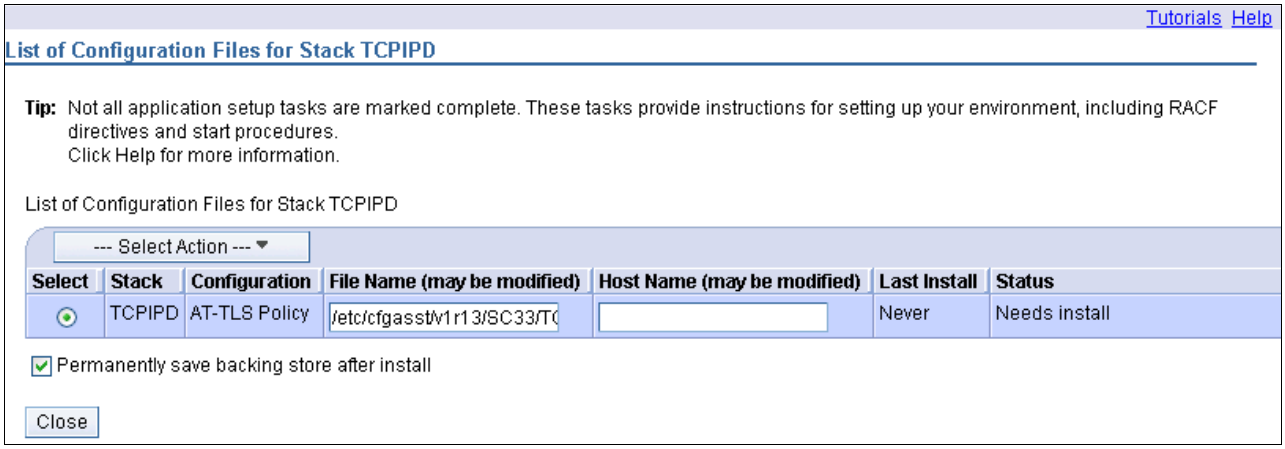


Figure 5-13 List of Configuration Files panel

14. At the next panel (Figure 5-14), we save the policy in the z/OS local file system by selecting **Save to disk**. Another option is to use FTP to transfer the policy to another z/OS; enter the IP address, port number, user ID and password in that case. Click **Go**.

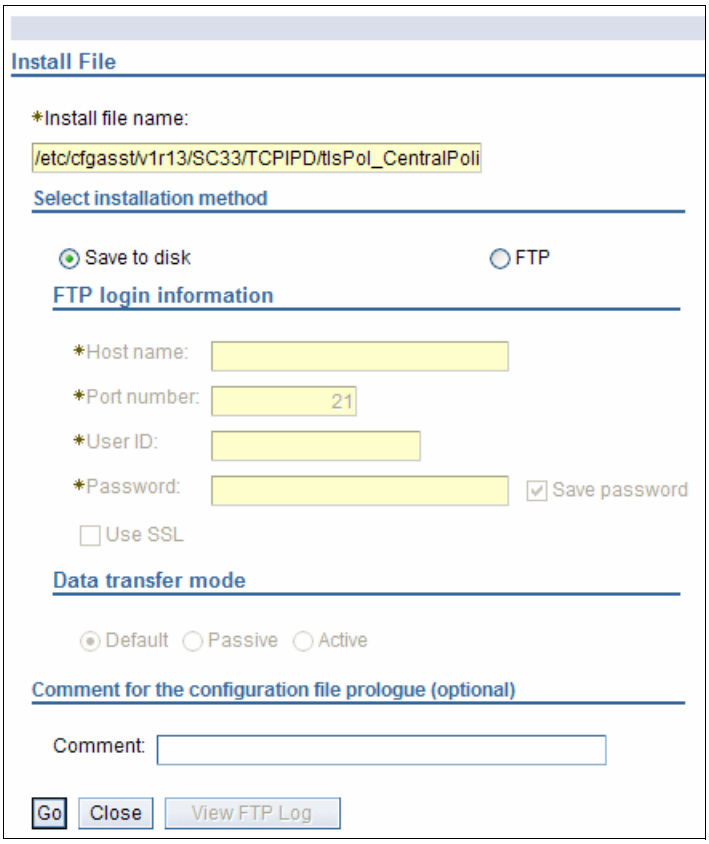


Figure 5-14 Install Files to Remote Host panel

Again, the backing store is updated again before the actual FTP of your policies begins.

After testing and verification, as explained in 5.4, “Activating and verifying the policy services environment” on page 188, you can 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 110 and execute the following RACF command:

```
PERMIT EZB.INITSTACK.SC33.TCPIPD CLASS(SERVAUTH) ID(PAGENT) ACCESS(READ)
```

5.3.3 Configuring the policy client

Now that the centralized policy server is configured and the AT-TLS policy file for the SSL connection between client and server is built, you can build the PAGENT configuration files for the centralized policy client.

In our scenario, the TCPIPA stack at SC30 is our client. At least two configuration files and two configuration statements for the centralized policy client are required:

- The main PAGENT configuration file, which contains the `ServerConnection` statement
- The `TcpImage` configuration file, which contains the `PolicyServer` statement

Both files are edited manually without the help of the z/OSMF Configuration Assistant. Example 5-8 shows the configuration of the ServerConnection statement in the main PAGENT configuration file for the policy client. We copy the sample pagent configuration file from `/usr/lpp/tcpip/samples/pagent.conf` and modified it for our purposes.

Example 5-8 Main configuration file on SC30 (pagent.sc30.conf) for policy client

```
# LogLevel Statement
  LogLevel 255

# TcpImage and PEPInstance Statements (synonyms)
  TcpImage TCPIPA /etc/pagent.sc30.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 the main PAGENT configuration file statements. The main configuration file points to a TcpImage file called `/etc/pagent.sc30.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 through TCP to this address. Instead of an IP address, you might 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 key ring at the client side of the connection. It must be prefaced with the user ID that owns the SSL key ring 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 can also contain the client certificate if SSL Client Authentication has been enabled for the connection. 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.

Example 5-9 shows our customizing of the client's `TcpImage` file identified by 1 in Example 5-8 on page 183. We copy the sample image file from `/usr/lpp/tcpip/samples/pagent.conf` and omit all statements except for `PolicyServer`.

Example 5-9 /etc/pagent.sc30.tcpipa.conf on SC30

```
# #  ******* Being used as a Central Policy Client (TCPIPD=Server) ***
#  IMAGE FILE FOR TCPIPA ********************************************
# 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 170. The user ID does not need to be unique. Multiple policy clients can 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 can take place with a password or with a passticket. First implement the Policy Server and Policy Client relationship with the simple password method. After you have the implementation functions as you want, 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 170. In our example, we used the Password authentication method using the password assigned to the policy client user ID in "RACF user ID and password for the policy client" on page 170. 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. See *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 SC30 and the TCP stack name is TCPIP, for example, the policy ClientName becomes SC30_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 valid values are IDS, IPSec, QoS, Routing, and TTLS.

Tip: If you have older PAGENT configuration files with traffic regulation (TR) policies defined under QoS, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776 for information about considerations for TR and IDS policies.

5.3.4 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. Figure 5-15 shows how a subset of the definitions relate to each other.

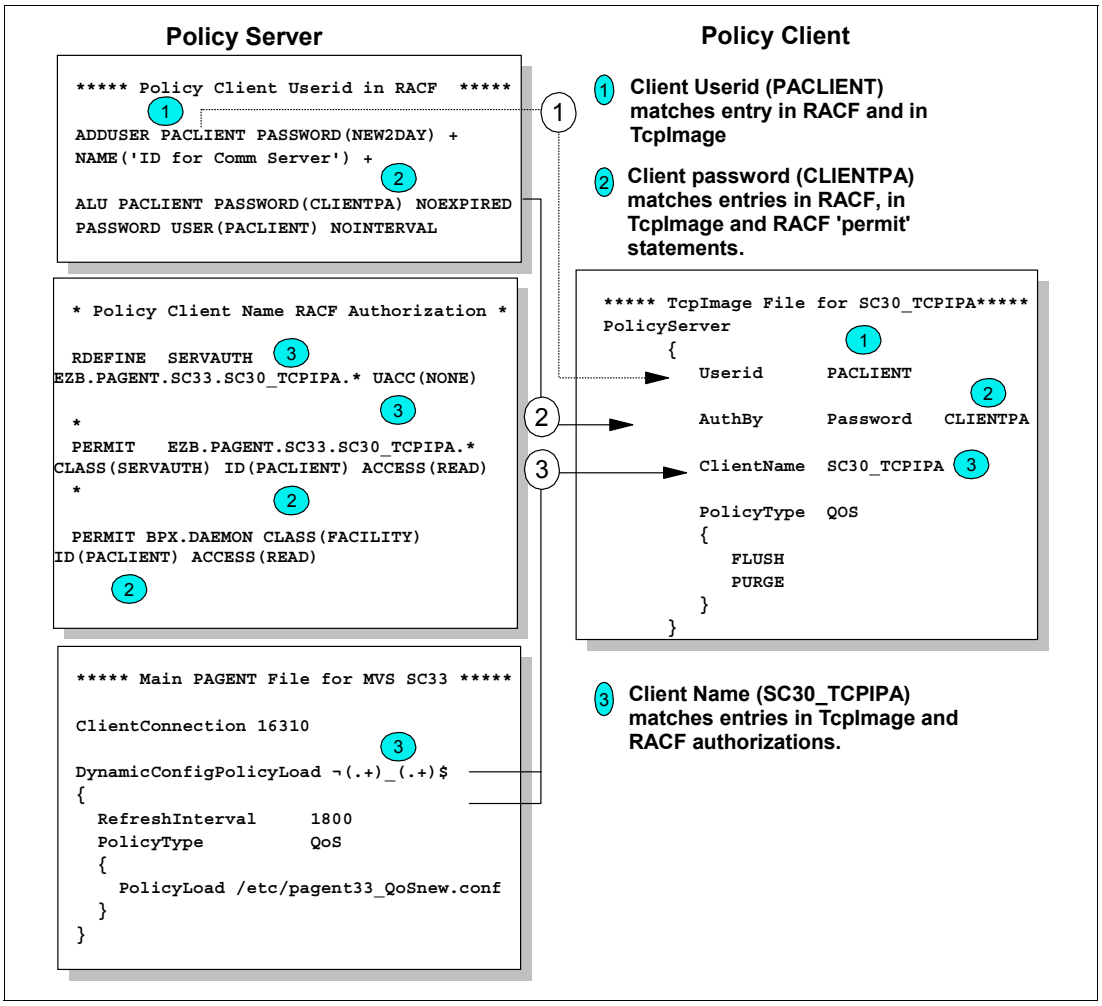


Figure 5-15 Correlation of Client User ID, Client Password, Client Name at Server and Client

Figure 5-16 shows the definitions for the SSL processing.

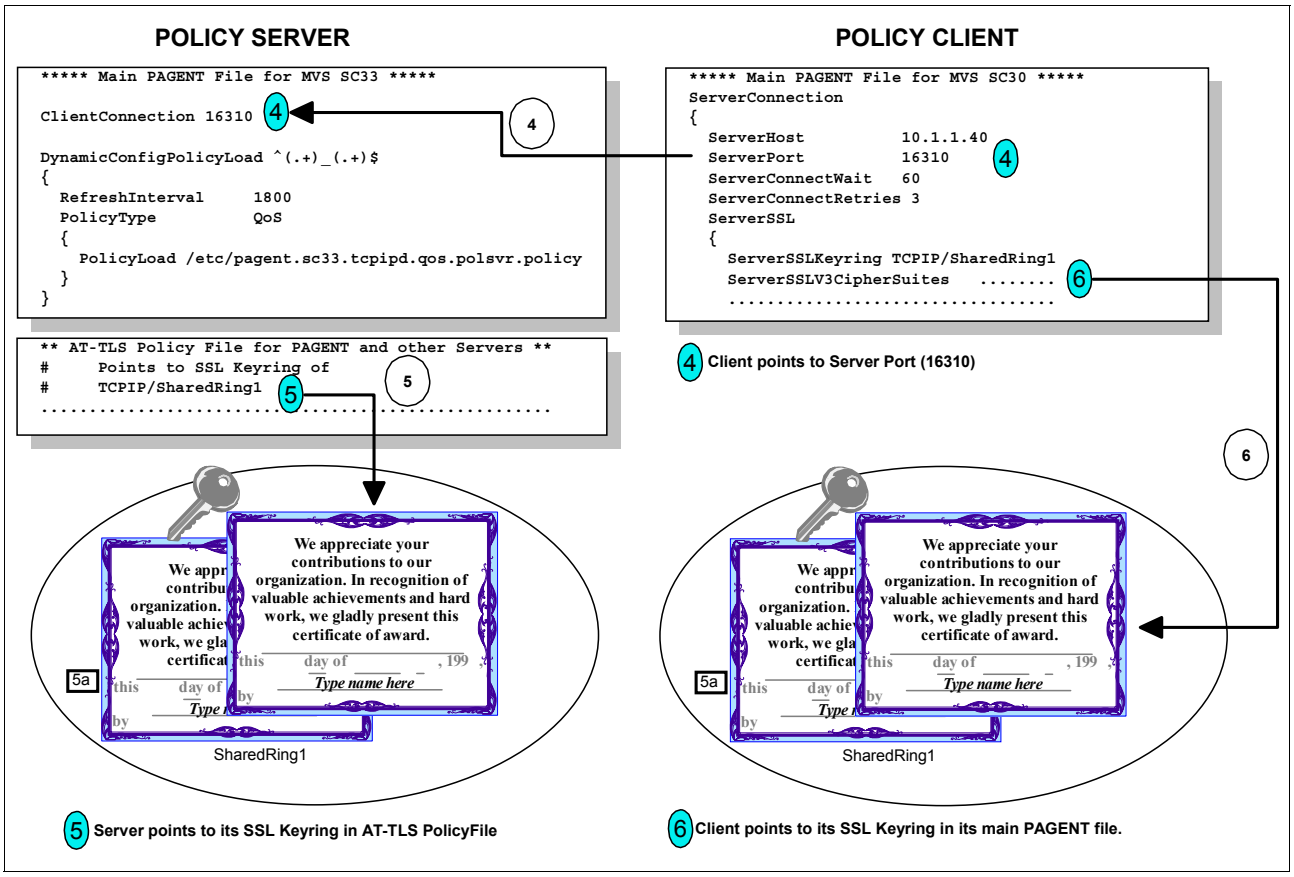


Figure 5-16 Shared key ring for the SSL connection between server and client

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, (4) in Figure 5-16. The client requests an SSL connection implicitly; that is, there is no need for an AT-TLS policy at the client side to establish a secure connection with the Central Policy Server. The coding labeled as (6) identifies the client's key ring.

Tip: Our implementation of the key ring at this installation uses a shared key ring. 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 key ring, 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.

5.4 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 5-10.

Example 5-10 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 5-11 reveals that PAGENT is bound to the Policy Server port that we had identified in our configuration file: 16310.

Example 5-11 D TCPIP,TCPIPD,N,CONN output

```
PAGENT   00000022 LISTEN
  LOCAL SOCKET:   ::..16310    b
  FOREIGN SOCKET: ::..0
```

In Example 5-12, 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 5-12 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
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCPIPA
EZZ8781I PAGENT CONNECTED TO POLICY SERVER FOR TCPIPA : PRIMARY AT
10.1.1.40
EZZ8790I PAGENT REMOTE POLICY PROCESSING COMPLETE FOR TCPIPA : QOS
EZD1589I PAGENT HAS INSTALLED ALL REMOTE POLICIES FOR TCPIPA
```

Tip: 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 5-13, shows that the SSL or TLS connections between Policy Server and Policy Client are indeed established.

Example 5-13 D TCPIP,TCPIPD,N,CONN output

```
D TCPIP,TCPIPD,N,CONN
...
PAGENT   00000277 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1029
PAGENT   00000279 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1030
```

We then verified from the z/OS console at the server (z/OS system ID of SC33) that the connections established were indeed based on an AT-TLS policy. We issued the **netstat ttls** command. The output from this command is shown in Example 5-14.

Example 5-14 TLS connections between the policy server and the policy client

```
D TCPIP,TCPIPD,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPD 169
TTLSGRPACTION                          GROUP ID           CONNS
GACT1~CENTRALIZED_POLICY_SERVER        00000003               2
1 OF 1 RECORDS DISPLAYED
```

We displayed the connection at the server, filtering on the address space name of PAGENT, and obtained the results shown in Example 5-15.

Example 5-15 D TCPIP,TCPIPD,N,CONN,CLIENT=PAGENT output

```
D TCPIP,TCPIPD,N,CONN,CLIENT=PAGENT
EZD0101I NETSTAT CS V1R13 TCPIPD 190
USER ID  CONN     STATE
PAGENT   00000277 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1029
PAGENT   00000022 LISTEN
  LOCAL SOCKET:   ::..16310
  FOREIGN SOCKET: ::..0
PAGENT   00000279 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1030
END OF THE REPORT
```

We looked at the basic TTLS information by issuing the **netstat TTLS** command with the connection identifier shown in Figure 5-16 on page 187.

Example 5-16 D TCPIP,TCPIPD,N,TTLS,CONN=90DE output

```
D TCPIP,TCPIPD,N,TTLS,CONN=277
EZD0101I NETSTAT CS V1R13 TCPIPD 196
CONNID: 00000277
  JOBNAME:      PAGENT
  LOCALSOCKET:  ::FFFF:10.1.1.40..16310
  REMOTESOCKET: ::FFFF:10.1.2.10..1029
  SECLEVEL:     TLS VERSION 1 a
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA   b
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
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
D TCPIP,TCPIPD,N,TTLS,CONN=279
EZD0101I NETSTAT CS V1R13 TCPIPD 203
CONNID: 00000279
  JOBNAME:      PAGENT
  LOCALSOCKET:  ::FFFF:10.1.1.40..16310
  REMOTESOCKET: ::FFFF:10.1.2.10..1030
  SECLEVEL:     TLS VERSION 1 a
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA   b
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
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 version 1. 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 5-17.

Example 5-17 D TCPIP,TCPIPD,N,TTLS,CONN=x,DETAIL output

```
D TCPIP,TCPIPD,N,TTLS,CONN=277,DETAIL
EZD0101I NETSTAT CS V1R13 TCPIPD 208
CONNID: 00000277
  JOBNAME:      PAGENT
  LOCALSOCKET:  ::FFFF:10.1.1.40..16310
  REMOTESOCKET: ::FFFF:10.1.2.10..1029
  SECLEVEL:     TLS VERSION 1
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

```
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
TTLSRULE: CENTRALPOLICYRULE~1
  PRIORITY:       255
  LOCALADDR:      ALL
  LOCALPORT:      16310
  REMOTEADDR:     ALL
  REMOTEPORTFROM: 1024              REMOTEPORTTO: 65535
  DIRECTION:      INBOUND
  TTLSGRPACTION:  GACT1~CENTRALIZED_POLICY_SERVER
    GROUPID:                    00000003
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
    FIPS140:                    OFF
  TTLSENVACTION:  EACT1~CENTRALIZED_POLICY_SERVER
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1    d
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    TLSV1.1:                    ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    TRUNCATEDHMAC:              OFF
    CLIENTMAXSSLFRAGMENT:       OFF
    SERVERMAXSSLFRAGMENT:       OFF
    CLIENTHANDSHAKESNI:         OFF
    SERVERHANDSHAKESNI:         OFF
    CLIENTAUTHTYPE:             REQUIRED    e
    CERTVALIDATIONMODE:         ANY
  TTLSCONNACTION: CACT1~CENTRALIZED_POLICY_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
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SECONDARYMAP:               OFF
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

In Example 5-17 on page 190 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 authenticated with user ID and password (or passticket) and additional RACF controls have been put in place for this authentication.

Next we went to the UNIX shell on the server system and issued the **pasearch** command with the **-C** option as shown in Example 5-18.

Example 5-18 Display of local stacks and policy clients at policy server node

```
CS02 @ SC33:/u/cs02>pasearch -C

TCP/IP pasearch CS V1R13
  Date:                 07/19/2011        Time:  10:29:22

  Policy Agent image names with policies configured:
    TCPIPD      1
    SC30_TCPIPA 2
```

The example output in Example 5-18 shows that PAGENT in z/OS SC33 is managing the policies for the 1 TCPIPD local stack and for the policy client named 2 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 5-19, 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 User ID of PACLIENT 3. The QoS policy object that applied to this client is `/etc/pagent.sc33.tcpipd.qos.polsvr.policy` 4.

Example 5-19 Verifying at the server that a specific client has retrieved policy

```
CS02 @ SC33:/u/cs02>pasearch -c -p SC30_TCPIPA    2

TCP/IP pasearch CS V1R13                 Image Name: SC30_TCPIPA
  Date:                 07/19/2011       Time:  10:30:48
  PAPI Version:         10               DLL Version:  10

Qos Policy Object:
  ConfigLocation:       Client    1      LDAPServer:        False
  ImageFileName:        /etc/pagent.sc33.tcpipd.qos.polsvr.policy    4
  ClientUserid:         PACLIENT    3
  PolicyClientAddr      ::ffff:10.1.2.10
  PolicyClientPort:     1029
  ConnectTime:          Tue Jul 19 10:16:29 2011
  ApplyFlush:           True             PolicyFlush:       True
  ApplyPurge:           True             PurgePolicies:     True
  AtomicParse:          False            DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   True             ModifyOnIDChange:  True
  Configured:           True             UpdateInterval:    1800
  PerfColEnabled:       False
  InstanceId:           1311084989
  LastPolicyChanged:    Tue Jul 19 10:16:29 2011
```

The **pasearch -c** command at the client showed that the policy location is Remote. Example 5-20 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 5-20 Display of all policies at TCPIPA

```
CS02 @ SC30:/u/cs02>pasearch -c -p TCPIPA

TCP/IP pasearch CS V1R13                  Image Name: TCPIPA
  Date:                 07/19/2011        Time:  10:32:25
  PAPI Version:         10                DLL Version:  10

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
  ClientSSLActive:      True    g
  ConnectTime:          Tue Jul 19 10:16:29 2011
  ApplyFlush:           True              PolicyFlush:       True
  ApplyPurge:           True              PurgePolicies:     True
  AtomicParse:          False             DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   False             ModifyOnIDChange:  True
  Configured:           True              UpdateInterval:    1800
  PerfColEnabled:       False
  InstanceId:           1311084989
  LastPolicyChanged:    Tue Jul 19 10:16:29 2011
...
```

In Example 5-20, note that the QoS policy had been retrieved from a Remote (a) location.

The lines labeled as b, c, d, e, and f display the QoS Policy Objects being used to communicate with the central policy server. The line labeled as g shows that SSL is active for this client connection.

5.5 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 z/OS system consoles must be investigated. First, look up the message in the appropriate IP Messages manual. Then, 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 might need to temporarily raise that logging level to 511 to obtain more granular messages.

Tip: You can edit the `pagent.conf` file to change the loglevel. Or simply issue the **`MODIFY PAGENT,LOGLEVEL,LEVEL=n`** command from the z/OS console. Remember to reset the loglevel to 15 after you diagnose the problem or collect 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.

Alternatively, if they are accurate, you find messages similar to those shown in Example 5-21.

Example 5-21 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 5-22 at the client or the server when the intended policy is not loaded.

Example 5-22 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 information shown in Example 5-23 was collected with PAGENT loglevel of 15.

Example 5-23 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.sc33.tcpipd.qos.polsvr.policy', errno
EDC5129I No such file or directory.

OBJERR :009: ...plfm_get_file_time: Can not get FILE handle for information:
'/etc/pagent.sc33.tcpipd.qos.polsvr.policy'
```

The log entries in Example 5-23 provide the following information:

- The DCPL (DynamicConfigPolicyLoad) statement as coded at SC33 in the pagent.sc33.conf file does not match the client image name of SC30_TCPIPA.
- The 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 QoS file (`pagent.sc33.tcpipd.qos.polsvr.policy`) in the /etc directory. Therefore, 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.

Important: Policy agent requires the use of US Code Page 1047 for the proper character representation in the policy files.

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. 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.

5.6 Configuring the Central Policy Server without SSL Security

To eliminate SSL security on the connections between the Central Policy Server and one or more of its clients, you need to remove the SSL statements from the main PAGENT configuration file for any client that does not require additional security. Example 5-24 shows the SSL statements in the main PAGENT configuration file at TCPIPA on SC30 (`/etc/pagent.sc30.conf`) commented out.

Example 5-24 Implementation definitions for connections not secured by SSL/TLS

```
  ServerConnection
  {
    ServerHost                    10.1.1.40
    ServerPort                    16310
    ServerConnectWait             60
    ServerConnectRetries          3
#   ServerSSL
#   {
#     ServerSSLKeyring           TCPIP/SharedRing1
#     ServerSSLV3CipherSuites    TLS_RSA_WITH_3DES_EDE_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DHE_RSA_WITH_AES_256_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DHE_DSS_WITH_AES_256_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
#     ServerSSLV3CipherSuites    TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
#   }
  }
```

Example 5-24 shows that we have commented out the entire ServerSSL block in the main client PAGENT configuration file. However, in our scenario, this was not enough to eliminate an SSL connection request to the Central Policy Server. In the policy agent log in UNIX System Services at the client node, we found the messages shown in Example 5-25.

Example 5-25 Messages in PAGENT log at z/OS SC30 (Client node)

```
OBJERR :006: pclient_connect_to_server: unable to connect to policy server at
10.1.1.40, port 16310, rc = Read from Policy Agent failed
```

The OBJERR message indicates that the policy could not be read from the policy server because the client failed to connect. Indeed, a **pasearch** command at the client node for TCPIPA showed that there were no policies loaded. Therefore, we must investigate the PAGENT configuration for TCPIPD at SC33.

Tip: You might find that you must also alter the AT-TLS policy at the Central Policy Server if your AT-TLS policy requires a secure connection. Our definitions at the server platform showed that we were requiring SSL for any connection from and to any client IP address to the listening server port at 16310. The AT-TLS policy either needed to be eliminated or changed to require AT-TLS only with certain Central Policy clients but not with our policy client. We decided to alter the AT-TLS policy at the Server, as we explain next.

We removed the AT-TLS policy for Central Policy Server and refreshed the PAGENT procedure at z/OS SC33. We displayed the client-server connections in Example 5-26 to illustrate that, though connections exist, they are not supported by AT-TLS.

Example 5-26 Connections between client and server not secured by SSL/TLS

```
D TCPIP,TCPIPD,N,CONN
.....................................................................Lines deleted
PAGENT   00000022 LISTEN
  LOCAL SOCKET:   ::..16310
  FOREIGN SOCKET: ::..0
PAGENT   0000022F ESTBLSH    a
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1027
PAGENT   00000231 ESTBLSH    b
  LOCAL SOCKET:   ::FFFF:10.1.1.40..16310
  FOREIGN SOCKET: ::FFFF:10.1.2.10..1028
.....................................................................Lines deleted

D TCPIP,TCPIPA,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPA 433
TTLSGRPACTION                          GROUP ID           CONNS
0 OF 0 RECORDS DISPLAYED    c
END OF THE REPORT
D TCPIP,TCPIPA,N,TTLS
```

The **D TCPIP,TCPIPA,N,CONN** command at the TCPIPA stack on z/OS SC30 provides the connection IDs **a** and **b** for our sessions between the policy client and the policy server. The subsequent command (**D TCPIP,TCPIPA,N,TTLS**) shows that the two connections were not established with SSL/TLS security **c**.

The log at the policy client also shows that policy agent is able to connect to the policy server as shown in Example 5-27.

Example 5-27 Log from Central Policy Client

```
INFO   :006: pclient_connect_to_server: SSL is not configured
INFO   :006: pclient_connect_to_server: Associate with TCP/IP image name =
'TCPIPA'
EVENT  :006: pclient_connect_to_server: connected to policy server '10.1.1.40'
(10.1.1.40), port 16310
LOG    :006: pclient_connect_to_server: EZZ8781I PAGENT CONNECTED TO POLICY SERVER
FOR TCPIPA : PRIMARY AT 10.1.1.40
```

5.7 Additional information

Additional information is available from the following sources.

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. See *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for more information about this topic. However, Table 5-1 lists and explains the meaning of various regular expressions to provide insight into how our string was constructed.

Table 5-1 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, particularly the symbolic expressions in the DynamicConfigPolicyLoad statement, requires 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 might 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 5-1 for help, or see *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.
- Use 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, illustrated in Example 5-19 on page 192, lists which files are loaded. The syslog daemon log determines whether parsing failed.

Related documentation

For additional information about central policy server, see the following resources:

- *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

6

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).

We discuss the following topics in this chapter.

Section	Topic
6.1, “Quality of service (QoS) definition” on page 200	Basic concepts of QoS
6.2, “Configuring QoS in the z/OS Communications Server” on page 205	Describes what is needed to configure QoS in a z/OS environment
6.3, “QoS implementation” on page 207	Describes how to include the QoS definitions in the policy agent configuration
6.4, “Verifying and diagnosing the QoS implementation” on page 216	This section describes how to verify and diagnose the QoS policies implementation

6.1 Quality of service (QoS) definition

QoS is not a new concept even in a TCP/IP context. The original 1981 standard for the IP protocol, RFC 791, contained a field named Type of Service (TOS). Within this field were eight 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:

- IP networks were popular because of their simplicity (otherwise we might still be running SNA networks).
- 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, however, QoS has become of interest again. IP routers and hosts are starting to check the ToS field of IP packets and handle these packets according to QoS rules. Also, 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 to 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 within specific 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, because of 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 can 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 and 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 System z 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.

6.1.1 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 6-1). 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 (in this 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.

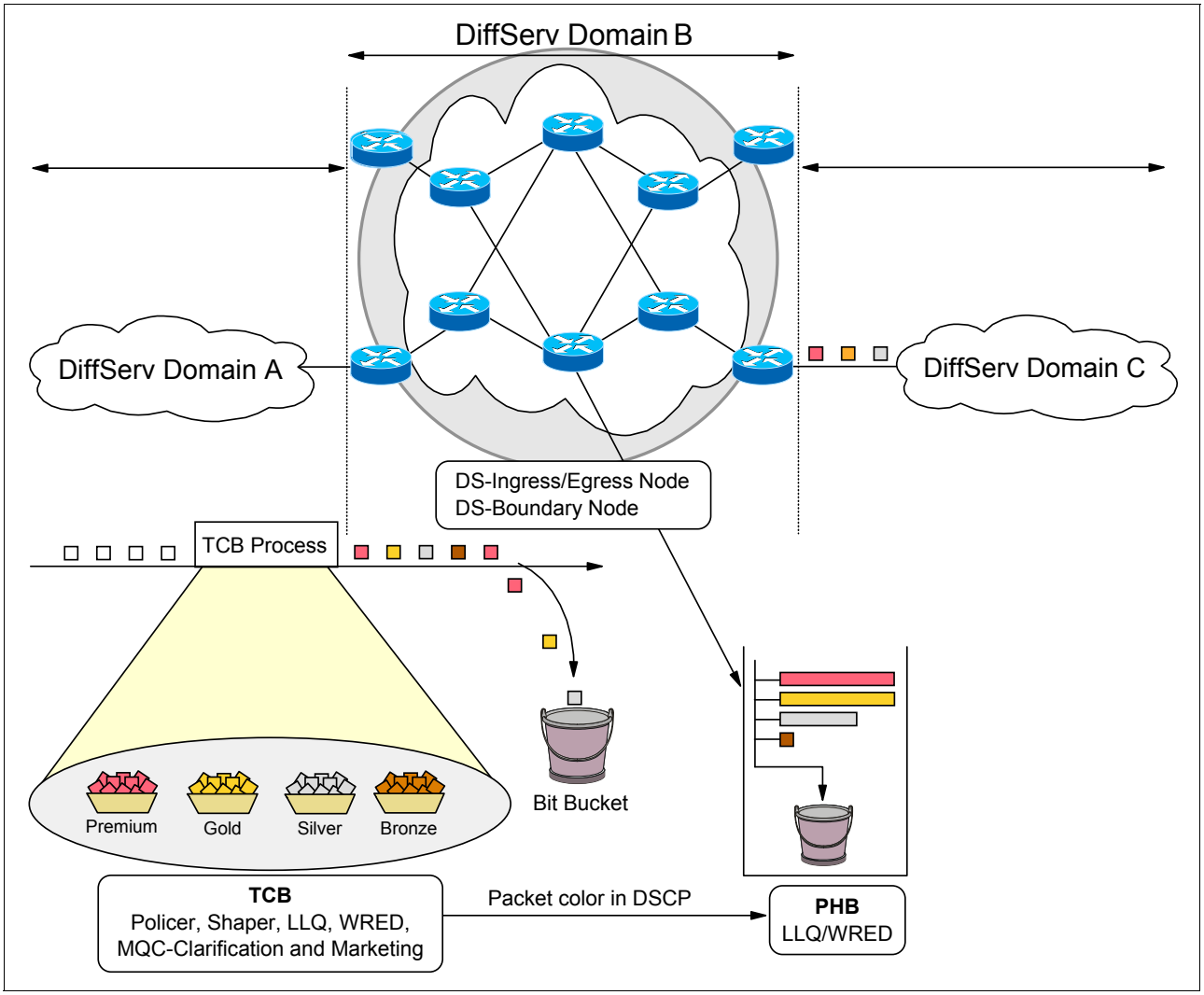


Figure 6-1 DiffServ end-to-end architecture

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 with 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 applications in DS-capable network devices, the IP packets are modified in a subset of the IP TOS. A small bit pattern in each IP packet, called the *DS field*, 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 6-2 shows the structure of the defined DS field.

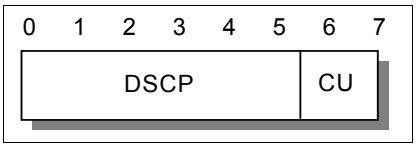


Figure 6-2 DS field

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 6-1.

Table 6-1 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

6.1.2 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 6-3 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.

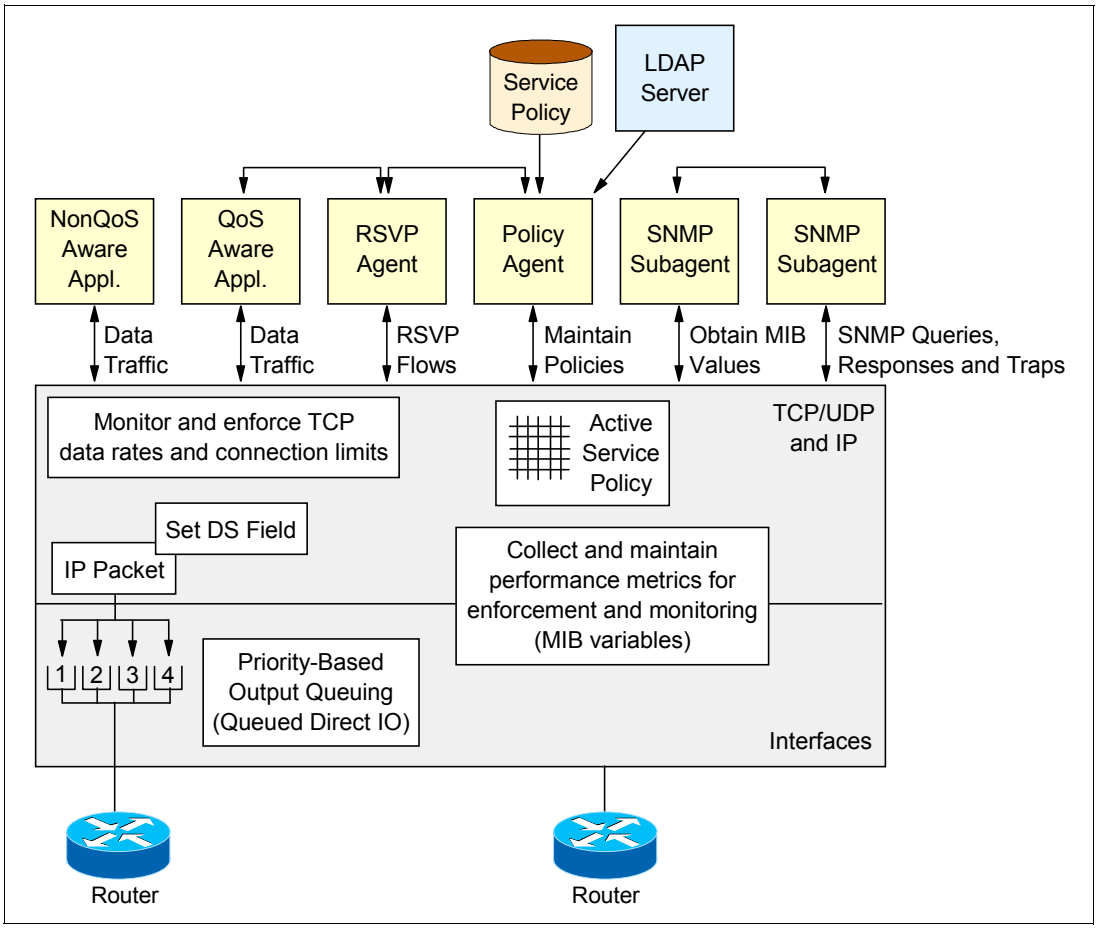


Figure 6-3 z/OS Communications Server quality of service components

Tip: Without a cooperative framework of host-based components and QoS mechanisms within the network (*and* the necessary inter organizational 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.

6.1.3 PAGENT QoS policies

When you first implement QoS policies, 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

Tip: Sysplex Distributor policies are discussed in *IBM z/OS Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7998.

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, and 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 can be referred to by several policy rules.

Differentiated Services policies

Policies to be implemented can be configured using 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.

6.1.4 Migrating TR QoS policies to intrusion detection services policy function

z/OS Communications Server no longer supports the traffic regulation (TR) policy as part of the QoS policy type. The TR policy function is still available but only as part of the intrusion detection services (IDS) policy type. if your QoS configuration policies have PolicyAction statements that specify PoliceScope TR parameter, these statements must be converted to an IDS policy statement.

Tip: This change is only for the TR policy configuration. The TR policy functions themselves remain unaffected and continue to run.

To do the conversion, complete the following steps:

1. Convert QoS policies that specify the PolicyScope TR parameter on the PolicyAction statement to IDS policies. The new IDS policies must be configured in an existing or new IDS configuration file.
   a. All PolicyRule statements that reference any such PolicyAction statement must be converted to an IDSRule statement with the ConditionType TR parameter.
   b. The PolicyAction statements must be converted to IDSAction statements with the ActionType TR parameter.
2. If you are not using IDS policies prior to the conversion, specify the IDSConfig statement to point to the new image-specific IDS configuration file.
3. Refresh the policies by issuing the **`MODIFY procname,UPDATE`** command, or restart the policy agent. If you are using UNIX files and you previously started the policy agent using the **`-i`** startup option, no action is necessary. The new policies are refreshed as soon as the configuration files are saved.

To help with the conversion, see *z/OS Migration*, GA22-7499, which provides conversion tables to migrate from QoS TR policies to IDS TR policies.

6.2 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 Communications Server 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.

In the z/OS Communications Server environment, QoS is configured using a set of configuration statements and parameters coded into a flat file, which is parsed by the policy agent to establish the QoS policy for each TCP/IP stack. To create these policy agent files, two options are available:

- Using the Configuration Assistant

  IBM provides a configuration GUI that you can use to generate the policy agent files. The tool provides four types of reusable objects to help create the QoS policies:

  – Traffic descriptors define the local application by describing the TCP traffic with ports or identifying the application using its job name.
  – Priority levels define the level of network priority or type of service (ToS).
  – Traffic shaping levels define settings to enforce specific traffic thresholds.
  – Requirement maps map traffic descriptors to priority levels and traffic shaping levels.

  After the policies are updated, PAGENT installs policies in one or more z/OS Communications Server stacks, replacing existing policies or updating them as necessary.

  In our examples in this book, we used the z/OSMF Configuration Assistant to implement our QoS policies. For further information about the Configuration Assistant, see 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

- Manual configuration

  You can create the QoS policy configuration files manually by coding all of the required statements in a file. This option is not recommended because of the amount of configuration options provided by QoS policy statements that could be necessary to create QoS policies on a per-stack basis. For details about the QoS policy statements, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

6.2.1 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 that are associated with the policy rule are executed.

6.2.2 Differentiated Services rule

The most common QoS deployment uses rules to map outbound traffic from particular applications into subclasses. Example 6-1 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 6-1 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
}
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, and the use of several DS-only attributes.

The following statements apply to Example 6-1:

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.

6.2.3 For additional information

For additional information about these topics, see the following resources:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Migration*, GA22-7499

6.3 QoS implementation

These sections describe our QoS implementation activities.

6.3.1 Using the Configuration Assistant to configure QoS

The Configuration Assistant enables centralized configuration of QoS policies for z/OS. The Configuration Assistant helps a network administrator produce a flat QoS policy file.

The Configuration Assistant is a tool for network administrators. Therefore, before you begin, read Chapter 4, “Policy agent” on page 103. 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.

In this example, we create a QoS policy for all FTP outgoing traffic in LPAR SC33, stack TCPIPD. We use z/OSMF Configuration Assistant to demonstrate this example.

Tip: Configuration Assistant help is available through the **Help** hyperlink.

You can implement QoS policies with the following steps:

1. Start the Configuration Assistant.
2. If this is a new backing store file, follow the steps (in 4.4.2, “General configuration steps using the Configuration Assistant” on page 128) to create a z/OS image to represent the z/OS System. Under this z/OS image add a TCP/IP stack as described in Chapter 4.4, “Configuration Assistant for z/OS Communications Server” on page 127. If your z/OS image already has an active policy agent running, you can also import the active configuration using the policy agent configuration file import services. You can then implement the QoS Services in the latest version of policy agent configuration in use.
3. On the Main Perspective panel, select the stack on which you want to enable QoS, TCPIPD, and, in the z/OS Communication Services Technologies table, select **QoS**.
4. Click **Select Action** → **Enable**. The status of the QoS settings changes from `disable` to `incomplete`. Next, click **Select Action** → **Configure**.

5. You may be prompted to add a connectivity rule. Click **Yes**. If you selected **No** or were not prompted, the QoS Perspective panel is displayed, as shown in Figure 6-4. In that case, click **Select Action** → **Add** to create a new connectivity rule.

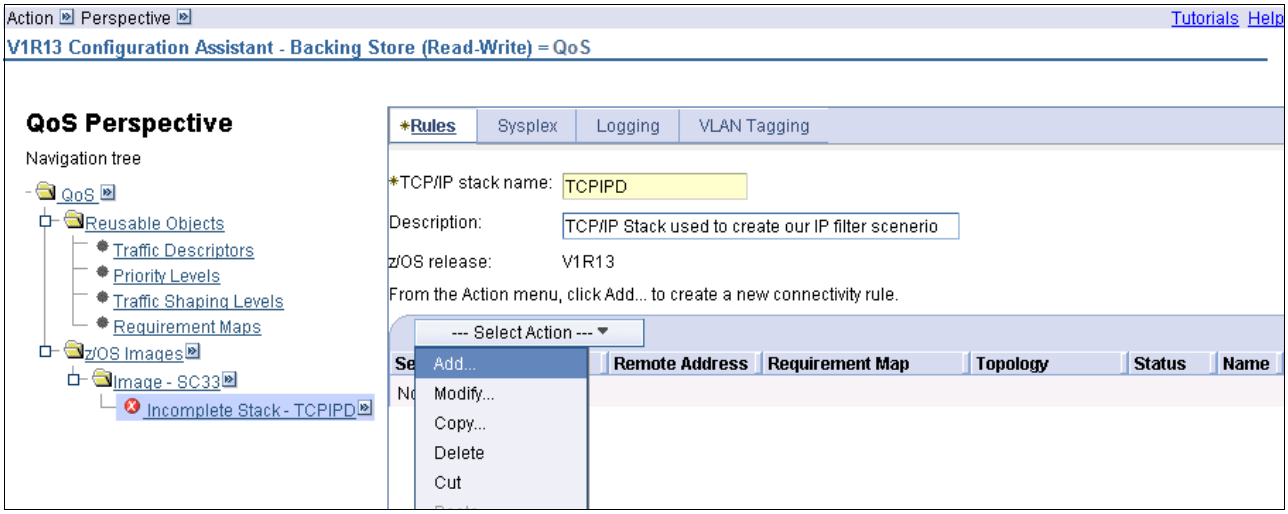


Figure 6-4 QoS Perspective panel: Add a new connectivity rule

6. The New Connectivity Rule panel is shown in Figure 6-5. We use a connectivity rule name of `PriorityControl`, enter our local z/OS IP address, and set the rule to apply to all remote IP addresses. Click **Next.**

Tip: Specifying the QoS policy rules is the most critical task and typically is 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.
- Optionally, you can specify that rules apply only during validity periods.

In association with the new connectivity rules, you must create the QoS requirement maps and define their validity ranges, which we discuss in the following sections.

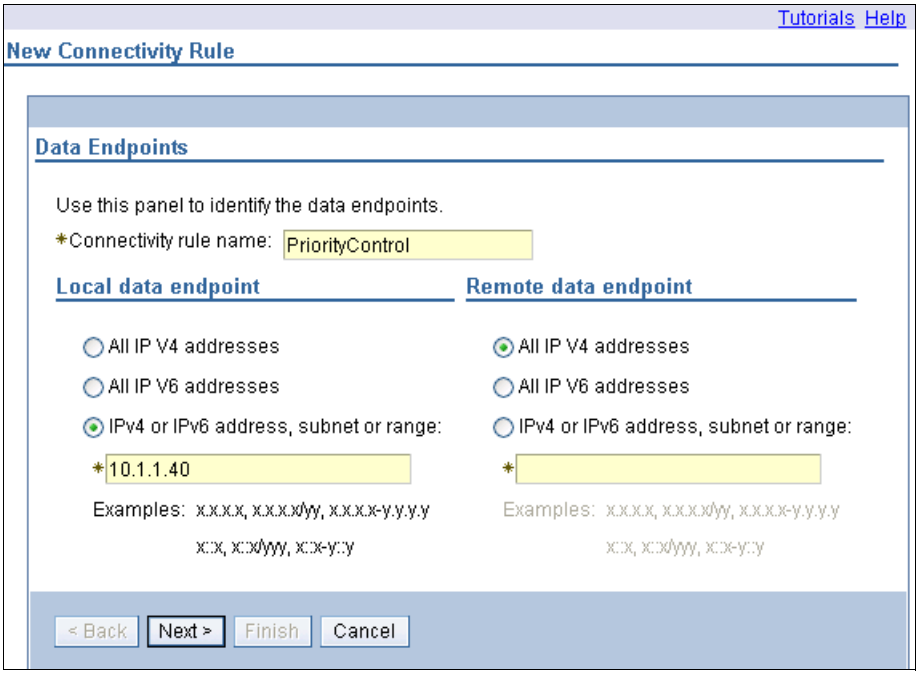


Figure 6-5 Connectivity rule entry panel

7. A Requirement Map panel is displayed. There are currently four IBM supplied default control type options as shown in Figure 6-6. If any of those matches your requirement, you can choose it.

   To see the types of traffic that are governed by this policy, click **Select Action** → **View Details**. In this example, our goal is to create a new policy to apply only to FTP client and server traffic. Click **Select Action** →**Add** to create our own requirement map.

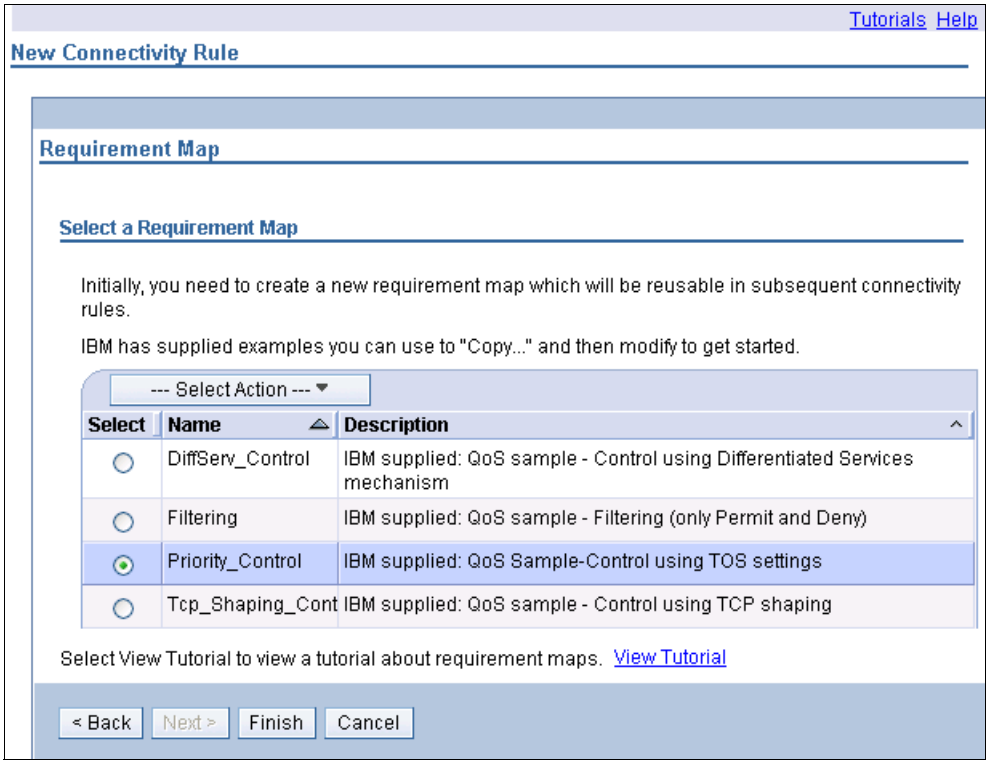


Figure 6-6 Requirement maps, including supplied defaults

8. New Requirement Map panel opens, as shown in Figure 6-7 on page 211. Enter the name of the requirement map (we use `FTP_Priority_Control` in this example). Select **FTP-Client** in the Traffic descriptor field and click **Add**. Do the same for **FTP-Server.** We leave the priority level for both FTP-Client and FTP-Server as Batch_1. Optionally, in the **New Requirement Map** panel, select **Set Effective Times** to set the periods of time on which the requirement map is active. After you finish setting the effective times, then in the Effective Times panel, click **OK**.

Tip: 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:

- *Priority levels* are objects that characterize 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 that characterize ways to control the levels of traffic. Traffic shaping levels can control TCP connections, TCP throughput, and Token Bucket traffic policing.

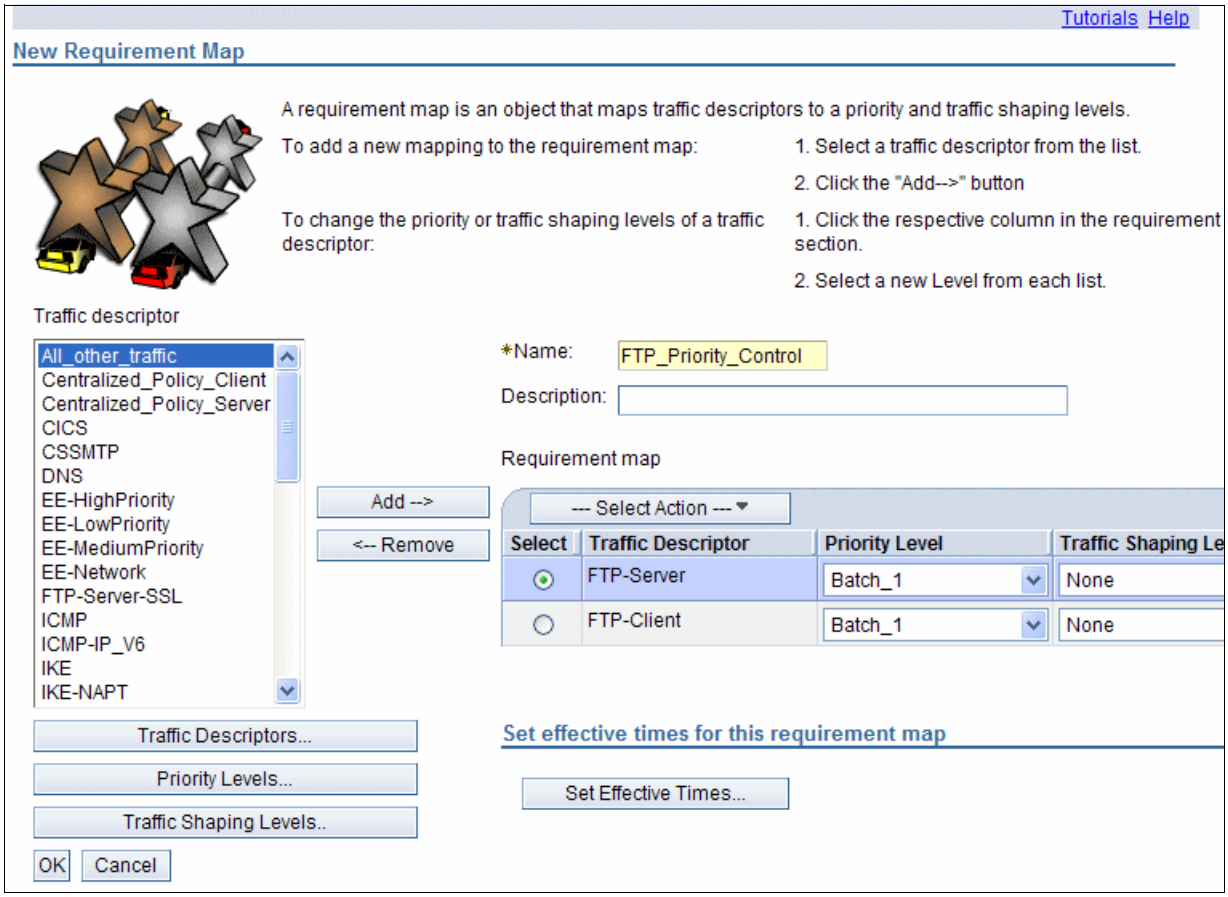


Figure 6-7 Creating a requirement map for FTP client and server

Tip: Be aware that these requirement map objects are reusable. As such, you can include them in multiple QoS connectivity rules. Generally, use only one requirement map between any two data endpoints. Therefore, a requirement map must contain all security requirements for all IP traffic between two data endpoints

9. Now you are back to the panel shown in Figure 6-6 on page 210, except that you have your newly created requirement map, `FTP_Priority_Control`, added to the list. Click **Finish** to view the QoS Perspective for your stack, as shown in Figure 6-8.

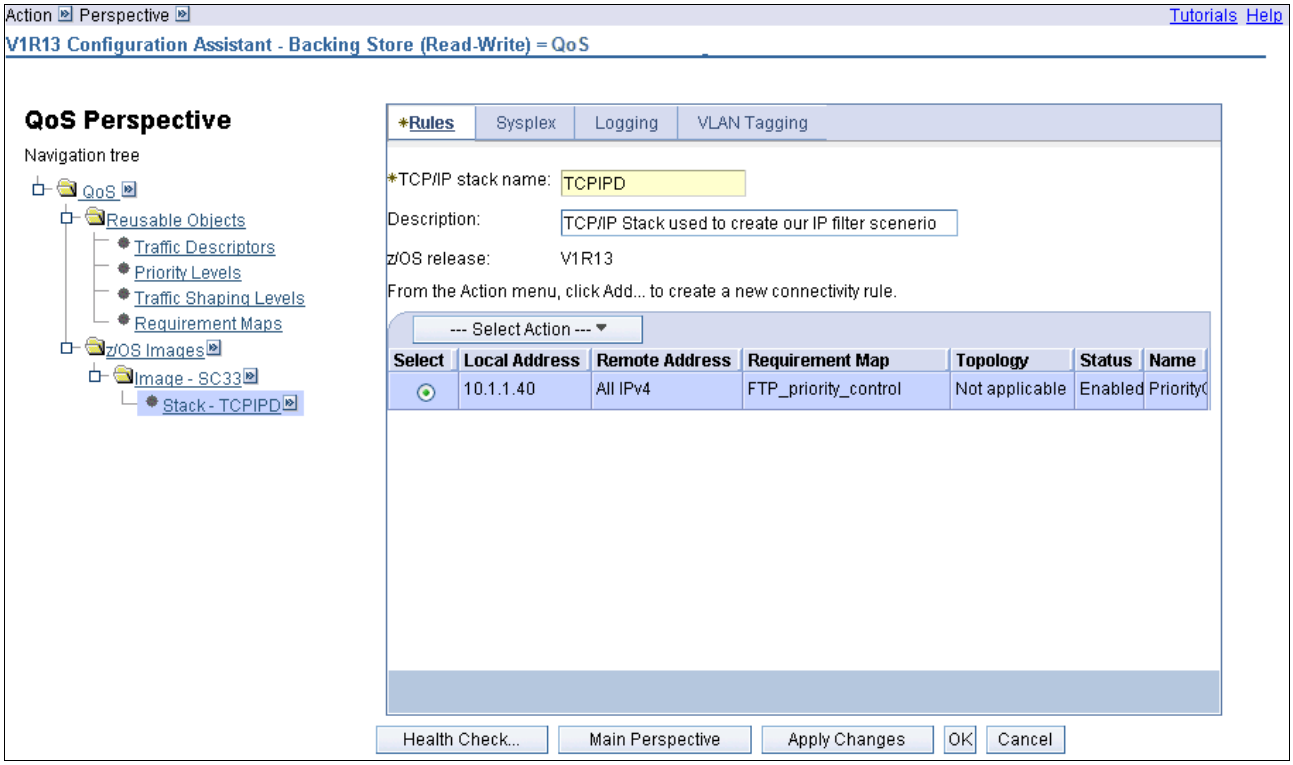


Figure 6-8 QoS perspective, showing newly created connectivity rule

Connectivity rule priorities

A connectivity rule object refers to one requirement object that 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 that are associated with the requirement map object are executed. Only one connectivity rule and associated actions can be applied to a particular packet.

With this information in mind, it is important to define a priority for the rule that is applied. In the QoS Perspective panel, the **Select Action** → **Move Up** and **Select Action** → **Move Down** options can be used to change the position of the Connection Rule to the desired priority, if two or more rules are defined. The first connectivity rule with a true condition is executed.

If you are using QoS in a sysplex environment, you may click the Sysplex tab (shown in Figure 6-8). The default is to not use QoS for sysplex distributor, so we left this tab unchanged.

If the stack is participating in a sysplex, complete the following steps:

1. In the QoS Perspective panel, click the **Sysplex** tab.
2. Select the **This Stack will use QoS for Sysplex Distributor** option.
3. Indicate the Sysplex Distributor Role for this stack, either as a Sysplex Target or a Sysplex Distributor.

   If you select **This Stack will be a Sysplex target**, click **Advanced Sysplex Target Settings** to indicate if this stack is defined as an *equal* or a *secondary target*, or if it should be the *first choice target* (which requires the XCF interface information for this stack). Click **OK**.
4. In the New Connectivity Rule: Finish panel, select **Next**. In the New Connectivity Rule: Finish panel, click **Advanced** to modify the QoS sysplex distributor settings. After you complete the modification, click **Finish**.

   Tip: The actual setup of Sysplex Distributor for high availability and workload balancing is discussed in detail in *IBM z/OS Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7998.
5. You may also change logging settings and configure VLAN priorities from this view. In our case, we click the Logging tab and enable performance logging to a file `/tmp/QoS_pfm.log`. We then click the VLAN Tagging tab and activate QDIO priority tagging as shown in Figure 6-9. We select **Enable for all interfaces**.
6. Click **OK**. The policy is configured for QoS.

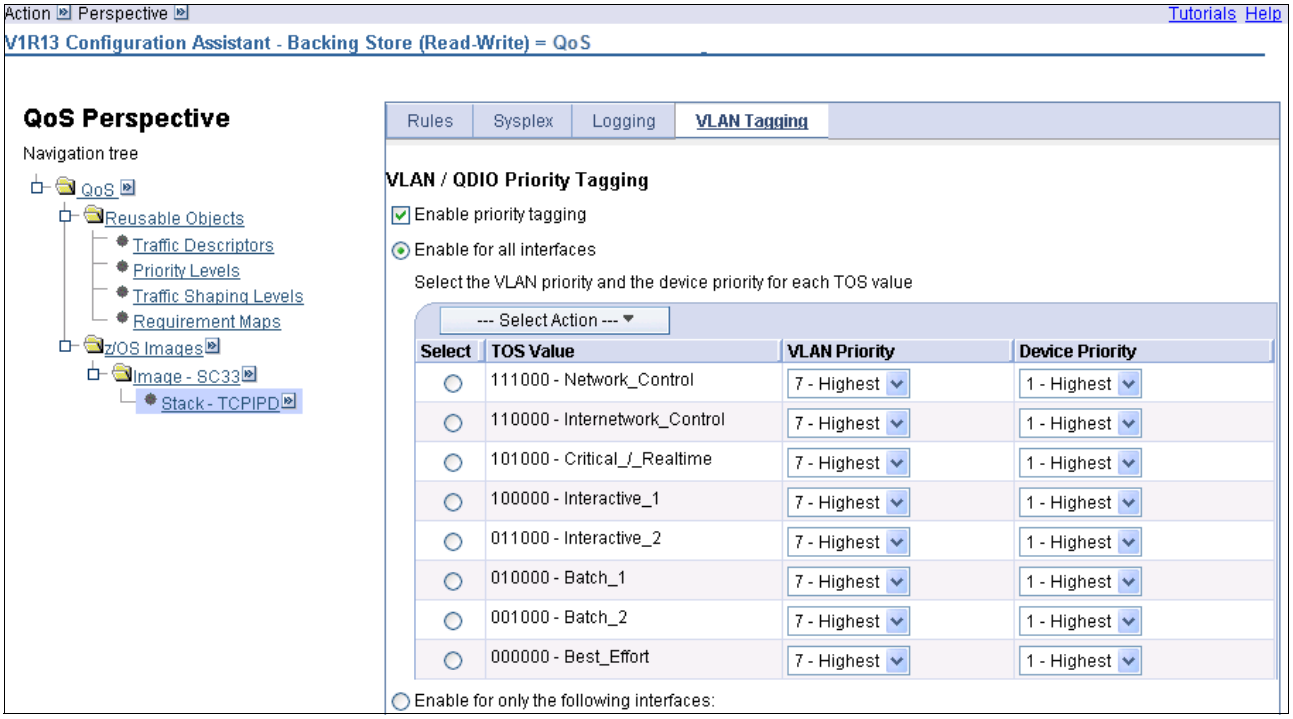


Figure 6-9 VLAN tagging enabled

Tip: The panel shown in Figure 6-9 is used to set specific parameters for virtual LAN (VLAN) and device priority tagging:

- With VLAN priority tagging you can 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.
- With device priority tagging you can correlate the TOS byte in outgoing IP packets to a priority for interfaces that use OSA-Express configured in QDIO mode.

Save the policy to the z/OS image

After you define the QoS policy, save the policy to the z/OS image as follows:

1. From the z/OSMF Configuration Assistant Navigation tree, click the small symbol to the right of your image name, and then click **Install Configuration Files**.

   Tip: Click **Health Check** from the Connectivity Rules tab before you install the configuration file to the z/OS image. The Health Check reports on errors in the policy. We found it helpful in our testing.

2. If you are asked to apply changes, click **Apply Changes**.
3. List of Configuration Files for Stack TCPIPD panel is displayed, as shown in Figure 6-10. Select the policy and click **Select Action** → **Show Configuration File**, which is how you can view the policy you are about to send to the host. Click **Select Action** → **Install**.

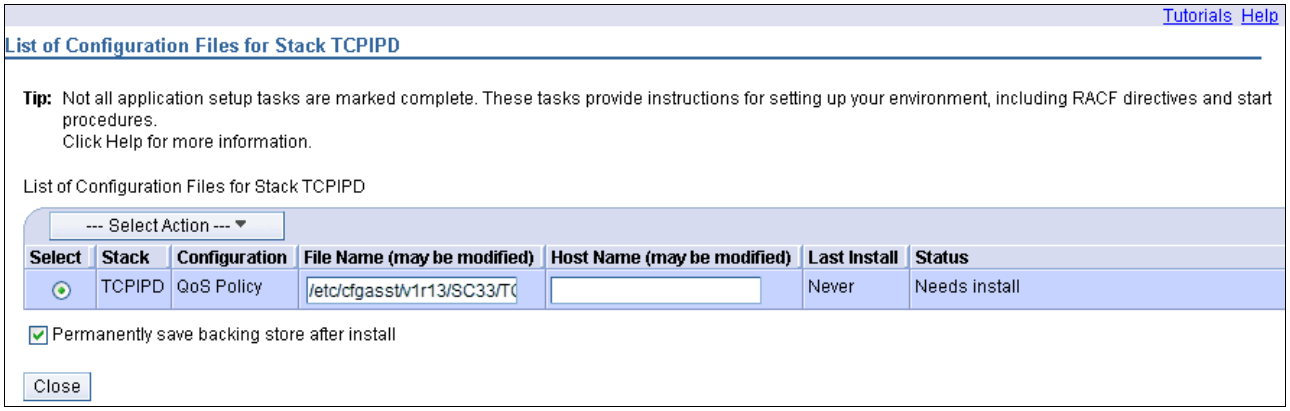


Figure 6-10 Using FTP to copy the QoS policy configuration file to the mainframe

4. The panel in Figure 6-11 opens. We enter the name of the location to where we want to store the policy file, and select **Save to disk** option to save the policy to the local z/OS file system. Another option is to use FTP to transfer the file to another z/OS image. Enter the IP address or host name, port number, user ID, and password in that case. Click **Go**. The save/transfer result is displayed on the bottom of the same panel. Click **Close**.

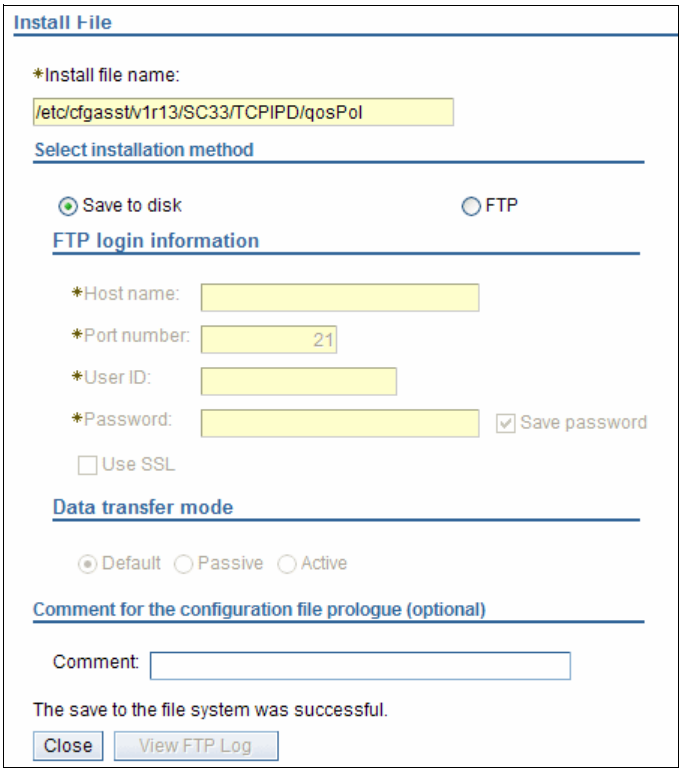


Figure 6-11 Information for install file

5. Now you are done with the Configuration Assistant. Save the backing store and close the Configuration Assistant.

6.3.2 Including QoS in the policy agent configuration

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 4, “Policy agent” on page 103.

1. After setting up the PAGENT, you must define the QoSConfig statement to specify the path of the policy file that contains stack-specific QoS policy statements to PAGENT. Example 6-2 shows the QoS statements in the TCP/IP stack configuration file used by policy agent in our scenarios.

Example 6-2 The pagent etc/pagent.sc33.tcpipd.conf file with QoS configured

```
######################################################################
# This file is used by Policy Agent and contains                     #
# statements for TCP/IP stack TCPIPD in z/OS image SC33              #
######################################################################
#    QoSConfig policy configuration                                  #
######################################################################
  QoSConfig /etc/cfgasst/v1r13/SC33/TCPIPD/qosPol
```

2. From the console, start the PAGENT or refresh the policy agent by issuing the following command:

   ```
   F PAGENT,REFRESH
   ```

   This operation installs the configured policy. The following messages might be displayed on the log:

   ```
   EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : QOS
   ```

6.4 Verifying and diagnosing the QoS implementation

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 received any error messages. In our configuration, the log file is `/tmp/pagent.sc33.log`.
- Use the Network SLAPM2 Subagent to display service policy and mapped application information, and to manage and display Network SLAPM2 performance monitoring.
- Use the SLA Subagent to display service policy and mapped application information, and 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.

6.4.1 Available management tools

There are also tools available with the base product 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.

6.4.2 z/OS Communications Server SNMP SLA Subagent

The z/OS Communications Server 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. See RFC 2758 at the following location for more information about the SLAPM MIB:

http://www.ietf.org/mail-archive/web-old/ietf-announce-old/current/msg07204.html

7

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 an early stage in message handling (and, therefore, efficiently). By “early,” we mean at a point when the packet is just about to enter the protocol stack and before any applications are executed or much “work” is performed.

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 to identify the IPSec behavior to apply to all traffic.

Tip: IBM z/OS Communications Server IP filtering support is packaged with IP Security (IPSec) and is referred to as *integrated IP Security* 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. To configure IP filtering, you have to indicate that you are configuring IPSec in the IBM Configuration Assistant for z/OS Communications Server.

We discuss the following topics in this chapter.

Section	Topic
7.1, “Define IP filtering” on page 218	Basic concepts of IP filtering
7.2, “z/OS IP filtering implementation” on page 220	Selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

7.1 Define 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

7.1.1 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), which is a table that maintains the set of filter rules, for a matching rule against the TCP/IP header. Filter rules have conditions and actions.

Filter rules contain the conditions to apply the rules based on any combination of the following attributes:

- Packet information
  - IP source or destination address (or masked address)
  - Protocol
  - Source or destination port address
- Direction of flow (inbound/outbound)
- Time of day

Filter rules also contain the actions to specify whether the packets that match the filter rule conditions are denied or permitted. The following possible actions can be taken:

- 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 usually not informed. However, a possibility is to configure a packet discard action to send an ICMP error when certain traffic is blocked.

To create the IP filtering policy, you 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 7-1 illustrates the basic concept of IP filtering.

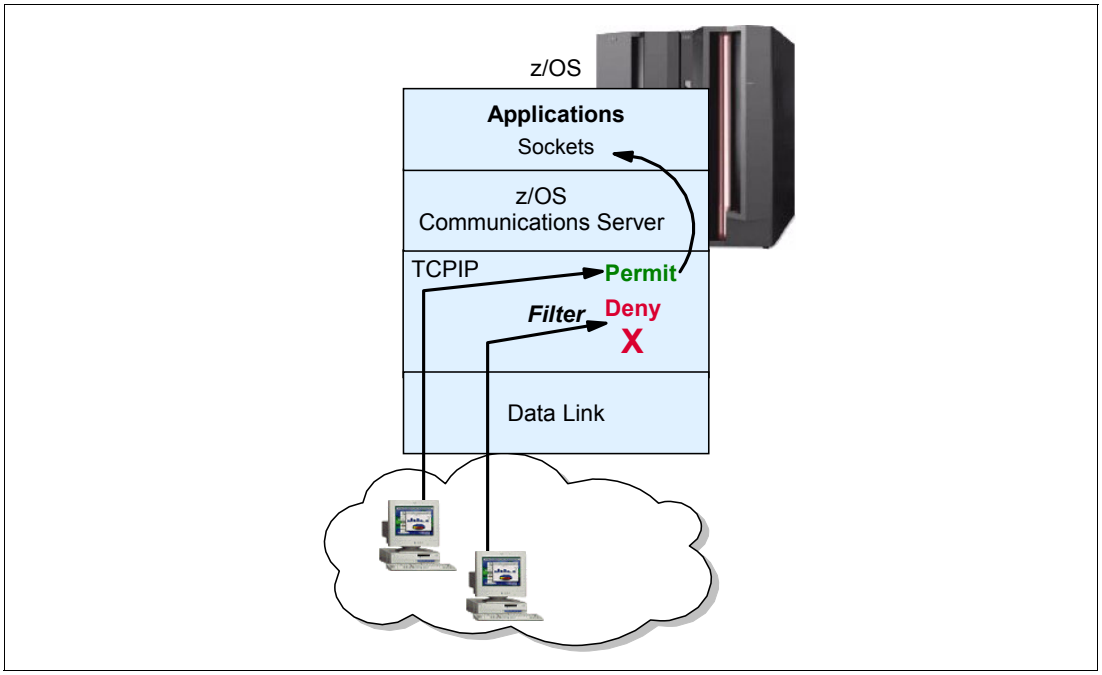


Figure 7-1 IP filtering at the z/OS data endpoint

The following resources are available in the z/OS image:

- 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, 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 following network resources should be mapped outside the z/OS image:

- 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.

Tip: 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.

7.1.2 IP filter policy types

The Security Policy Database is a table that maintains a set of filter rules. The Security Policy Database provides two types of filter policies: the *default IP filter policy* and the *IP security filter policy*:

- The default IP filter policy

  This policy provides only a basic filtering function (only permit rules and no VPN support). The default IP filter policy is defined in the TCP/IP profile; it contains an implicit rule to deny all traffic. It is intended to filter packets when the IP security filter policy is not available, in such situations that IP security filter policy is configured but the policy agent has not been started or completely initialized.

- The IP security filter policy

  This policy is intended to be the primary source of filter rules, and it also contains an implicit rule to deny all traffic. It is defined in a policy agent IPSec configuration file and can be generated by the z/OSMF Configuration Assistant.

  Use the **ipsec** command to switch between the default IP filter policy and IP security filter policy. Figure 7-2 shows a functional view of IP filter policy on z/OS.

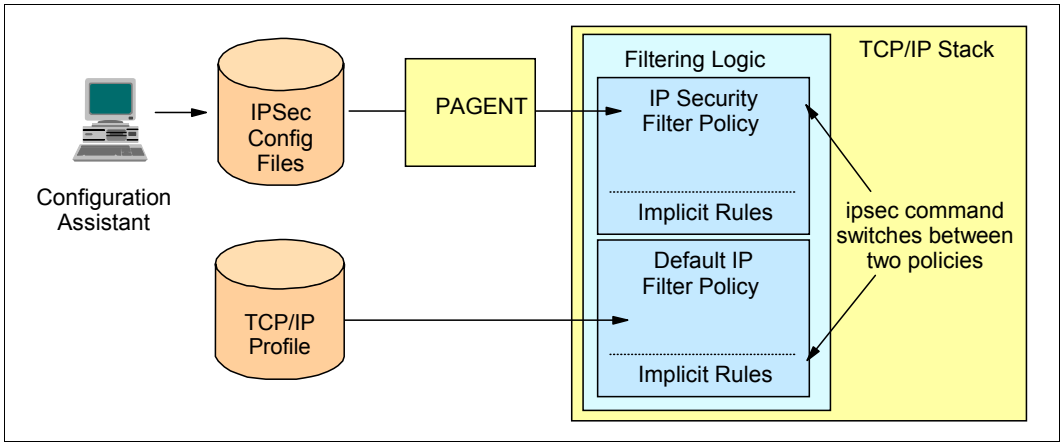


Figure 7-2 IP filter policy on z/OS

7.2 z/OS IP filtering implementation

This section describes the configuration and implementation steps.

7.2.1 Enabling IP Filtering

As mentioned, the two parts to implementing the IP filtering policy are *default IP filter policy* and *IP security filter policy*:

- Default IP filter policy
  - Implemented through the IPSEC statement in the TCP/IP profile data set.
  - Is always loaded by the stack when the configuration profile is first processed. If the initial profile does not have IP filter rules, only the implicit “deny all” rule will be loaded.
  - Contains only permit rules. If no rules are coded, all traffic is denied.
  - Has no option to group similar resources in the default IP filter policy.

- IP security filter policy
  - Is implemented using the PAGENT policy configuration files. These are flat files that can be created in a z/OS UNIX file or in a sequential data set. We discuss the PAGENT implementation in Chapter 4, “Policy agent” on page 103.
  - Provides an option to create deny rules, and also permit rules.
  - Policy agent loads the IP security filter policy into the TCP/IP stack at startup or when you make changes to it and refresh the policy by using the **modify** console command.

For either policy, 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 IPCONFIG IPSECURITY is not specified, 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.

Use the **ipsec** command to manage and monitor the IP security filtering.

7.2.2 Configuring default IP filter policy

The default IP filter policy is always used in the absence of an IP security filter policy. If IP security filter policy is defined, both are loaded into the TCP/IP stack, and the IP security filter policy is used unless you specify with the **ipsec** command to use the default IP filter policy. By using the **ipsec** command, you can switch between the default IP filter policy and the IP security filter policy as necessary.

If you use the default IP filter policy without IP security filter policy, consider the following information:

- The IPSEC/ENDIPSEC block statement is ignored if IPSECURITY is not specified on the IPCONFIG statement.
- Only one IPSEC/ENDIPSEC block statement block is allowed. Any subsequent statement blocks are ignored.
- If you code IPCONFIG IPSECURITY with no IPSEC/ENDIPSEC block statement, only local traffic flows through the stack (that is, loopback addresses).
- When using the default IP filter policy statements in the PROFILE.TCPIP, you can choose to permit local and routed traffic using the parameter ROUTING on the IPSECRULE and IPSEC6RULE statements. The options can be as follows:
  - LOCAL, indicating that this rule applies to packets destined for this stack
  - ROUTED, indicating the rule applies only to packets being forwarded by this stack
  - EITHER, indicating the rule applies to forwarded and non-forwarded packets
- 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.

[1] Define IPSECURITY in the IPCONFIG6 statement to enable IP filtering for IPv6.

Important: The implicit deny rules are always created using either the default or the filter policy. Using the IPSECURITY option in the IPCONFIG statement creates the implicit rules that deny all the inbound and outbound TCP/IP traffic in that z/OS image. Thus, if you code IPSECURITY without either IPSEC statements or filter policies, the stack will be completely inaccessible.

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.

Example 7-1 shows our IP filtering definitions in the TCP/IP profile for our z/OS test system SC33.

Example 7-1 IPSec configuration statements on profile for z/OS system SC33

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING IPSECURITY  SOURCEVIPA 1
DYNAMICXCF 10.1.7.51 255.255.255.0 1
...
; Added IPSEC statement
IPSEC 2
LOGENABLE 3
LOGIMPLICIT 4
;  ;;  OSPF protocol used by Omproute
  IPSECRULE *  *  NOLOG  PROTOCOL OSPF 5
;  ;;  IGMP protocol used by Omproute
 IPSECRULE *  *  NOLOG  PROTOCOL 2
;  ;;  DNS queries to UDP port 53
 IPSECRULE *  *  NOLOG  PROTOCOL UDP      SRCPORT *  DESTPORT 53 SECCLASS 1 6
; ;; Administrative access
; ;; (Telnet to/from local 10.1.1.40 and remote 10.1.100.222)
 IPSECRULE 10.1.1.40 10.1.100.222 LOG PROTO TCP SRCPORT 23 DESTPORT * ROUTING
LOCAL 7
; Use this rule to permit all remaining IPV4 traffic to be routed only.
 IPSECRULE *       *         NOLOG  PROTO *   ROUTING ROUTED 8
ENDIPSEC
..
INTERFACE OSA2080I
DEFINE IPAQENET
PORTNAME OSA2080
SOURCEVIPAINT VIPA1L
IPADDR 10.1.2.41/24
MTU 1492
VLANID 10
VMAC ROUTELCL
SECCLASS 1 6
...
```

In this example, the numbers correspond to the following information:

1. The IPSECURITY option is specified on the IPCONFIG statement, indicating that we want IP Security activated on the stack. If PAGENT is running and there is an IP security filter policy defined, it is installed and activated. If no PAGENT is running, only the rules defined in the IPSEC statement are used.
2. The IPSEC/ENDIPSEC statements are used to define the default IP filter policy. The default IP filter policy is activated and applied if the IP security filter policy is not defined.

Additional rules have to be defined to gain access to services that are running on this TCP/IP stack.

3. The LOGENABLE statement activates packet filter logging. All the log messages are sent to the syslogd by the Traffic Regulation Management Daemon (TRMD).

4. The LOGIMPLICIT statement specifies that we want to log all packets that get blocked by the implicit rules (“deny all”) in the default IP filter policy.

5. Each IPSECRULE statement specifies a permit policy being applied in the stack. The rules applied in the stack profile should be capable of providing a minimum administrative access to allow the support team to correct any unexpected event if the PAGENT fails or does not start. The rules are searched in the order they are defined.

6. The SECCLASS parameter on the IPSECRULE statement defines that the rule has to be applied only for those LINKS or INTERFACES identified by the same SECCLASS number. For further information about the SECCLASS definition, see “Use the SECCLASS parameter to assign a security class number” on page 223.

7. The ROUTING parameter in this IPSECRULE statement indicates that this rule applies only to packets that are destined to this stack (ROUTING LOCAL).

8. The ROUTING parameter in this IPSECRULE statement indicates that this rule applies to packets that are forwarded by this stack.

Use the SECCLASS parameter to assign a security class number

The SECCLASS parameter defined on LINK, INTERFACE, or IPCONFIG DYNAMICXCF statements enables you either to 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.

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 interface is the one on which the packet arrived.
- For outbound traffic, this interface is the one 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 7-3) cannot be distinguished into separate security classes.

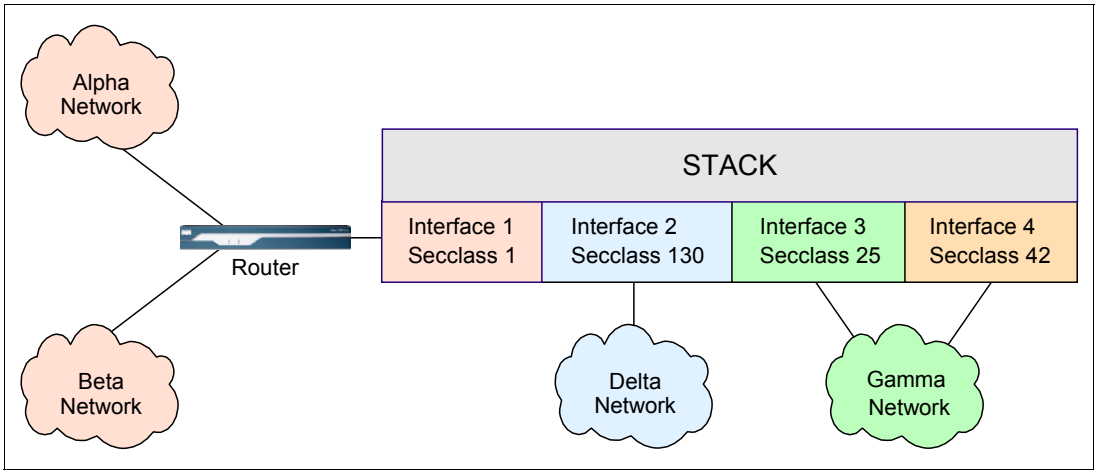


Figure 7-3 Interface security class example

Tip: Although it is possible to configure different security classes for different interfaces into the *same* network (for example, Gamma network in Figure 7-3), 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 that have spoofed source IP addresses. For example, if a packet enters the stack from the Delta network (in Figure 7-3), 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.

7.2.3 Configuring IP security filter policy using PAGENT

In our implementation, we configure and activate the following components:

- PAGENT (policy agent)
- TRMD (Traffic Regulation Management daemon)
- SYSLOGD
- TCP/IP address spaces

To activate the IP security filter policy as configured, PAGENT has to know where the policy file is located, by using the `IpSecConfig` statement.

Figure 7-4 shows the members to specify IP security filter policy file for PAGENT.

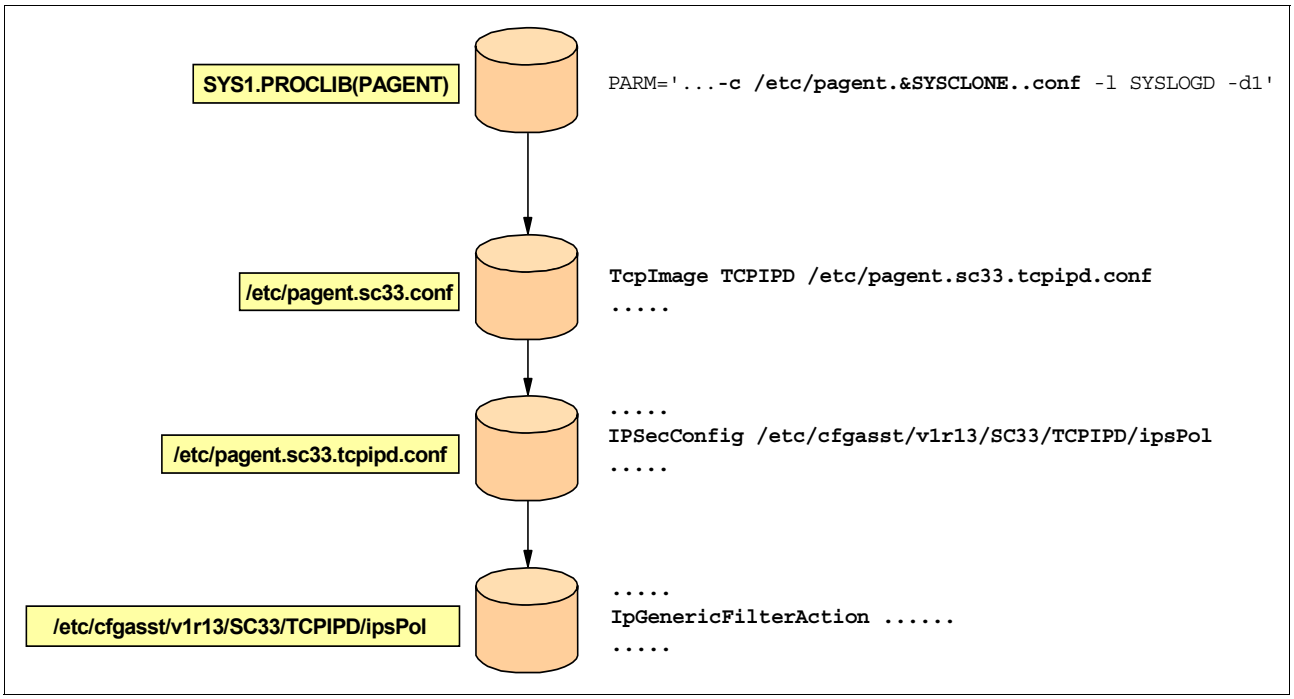


Figure 7-4 IPSec configuration members flow

The PAGENT configuration file in our implementation is `/etc/pagent.sc33.conf`, which points to each TCP/IP image specific configuration file (TCPIPD in our implementation scenario) using the statement `TcpImage` as follows:

```
TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf
```

Within the image configuration file, the `IpsecConfig` statement specifies the path of an IPSec policy file that contains the image IPSec policy statements. If no path name is specified, the common IPSec policy file specified on the `CommonIpSecConfig` statement is used.

z/OSMF Configuration Assistant is provided to more easily configure the policies with a graphical user interface (GUI). See 4.4, “Configuration Assistant for z/OS Communications Server” on page 127 for more information.

For a complete explanation of the IP filtering statements and options, see the following sources:

- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775

FTP and Telnet IP filtering scenario

Figure 7-5 shows the IP filtering implementation scenario.

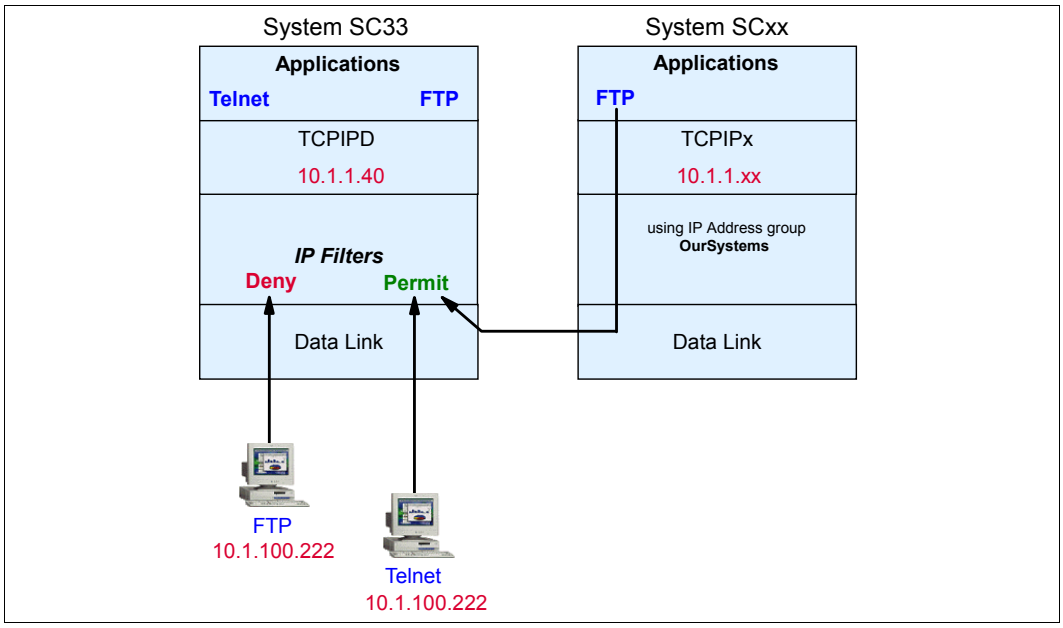


Figure 7-5 IP filtering scenario

In this scenario, we use the z/OSMF Configuration Assistant to define IP filter rules for system SC33 as follows:

- Permit Telnet traffic only between remote workstation (10.100.1.222) and System SC33, stack TCPIPD (10.1.1.40).
- Permit FTP traffic between IP address 10.1.1.40 (SC33) and all z/OS systems in our environment (IP address 10.1.1.xx), creating an IP address Group called OurSystem to represent the z/OS Systems.
- Permit *basic services* traffic between all data endpoints and SC33.

> **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 the following specific point-to-point connections. However, basic services traffic must also be permitted.
>
> - 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 you also need IGMP
> - OSPF: Inbound and outbound: IP protocol 89 (OSPF) and IGMP
>
> Generally, also permit ICMP traffic. This allows you to verify connectivity between data endpoints using the **`ping`** command.

- Deny all other application traffic for System SC33, stack TCPIPD.

Overview of implementation steps

We use the following steps to implement our scenario:

- Step 1: Enable IP filtering
- Step 2: Create the TN3270 traffic filtering rule
- Step 3: Create the FTP traffic filtering rule
- Step 4: Create the basic services traffic
- Step 5: Install the configuration files
- Step 6: Update the Policy Agent configuration file
- Step 7: Verification of IP filtering configurations

Step 1: Enable IP filtering

Complete the following steps:

1. Open the z/OSMF Configuration Assistant.

   Tip: z/OSMF Configuration Assistant help is available through the **Help** button. If you need detailed information for a particular field, click the question mark (**?**) button and then click the desired field.

   If this is a new backing store file, follow the steps to create a z/OS image to represent the z/OS system. Under this z/OS image, add a TCP/IP stack, as described in 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

   If your z/OS image already has an active policy agent running, you can also import the active configuration using the policy agent configuration file import services. Then you can implement the IP Filter policies in the latest version of policy agent configuration in use.

   After configuring or importing the z/OS image and TCP/IP stack files, you can implement IP Filter policies as the remaining steps describe.

2. The Main Perspective panel lists the new z/OS image and TCP/IP stack selected. Select **IPSec** in the z/OS Communications Server technologies list, and then click **Select Action** → **Enable**. The Status of IPSec line displays as `Incomplete` as shown in Figure 7-6 on page 228. Click **Select Action** → **Configure**.

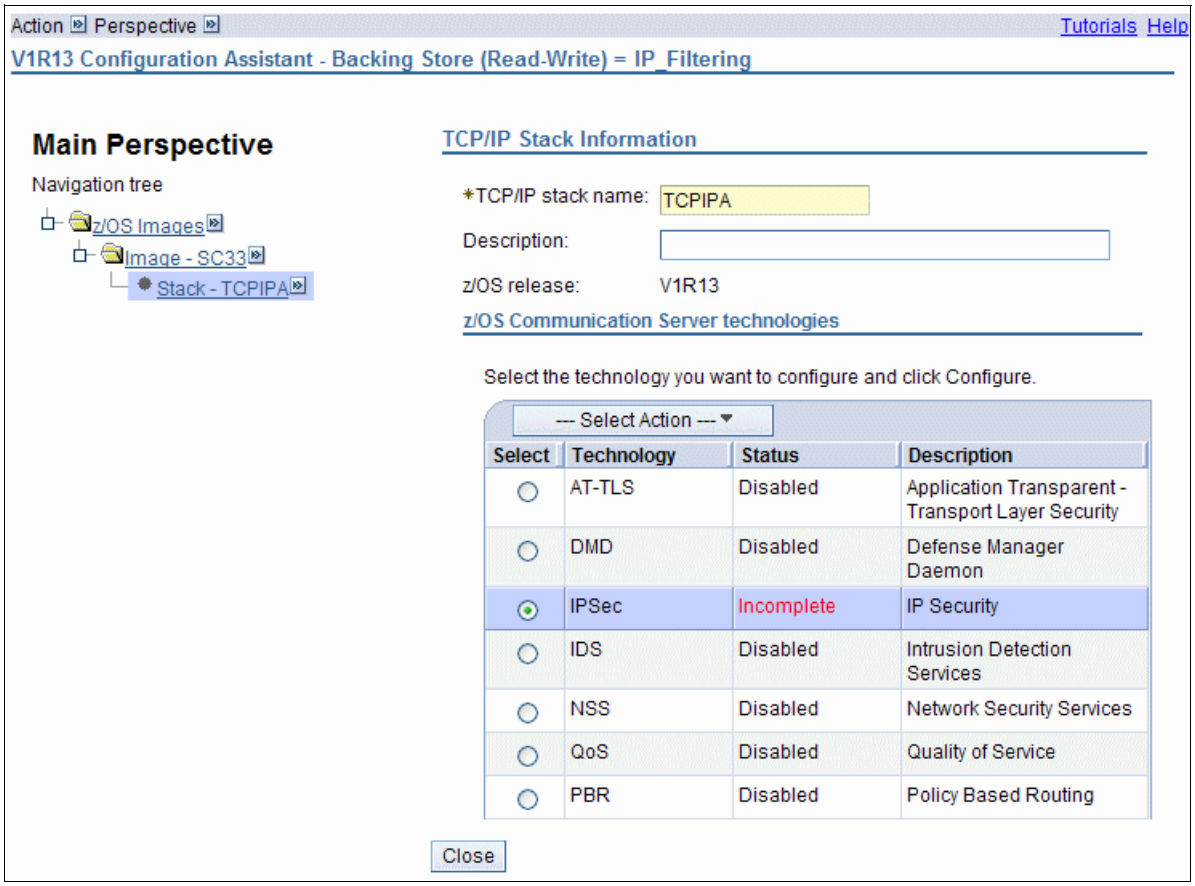


Figure 7-6 Enable IPsec configuration

3. You might be prompted to proceed by adding a connectivity rule. Click **Yes**. In the New Connectivity Rule welcome panel, click **Next** to continue. In these steps, we create a typical rule.

4. In the Network Topology panel (Figure 7-7), select the network topology for your environment. In our case, we want to do filtering only, and we chose to set up for handling local traffic. Click **Next** to continue.

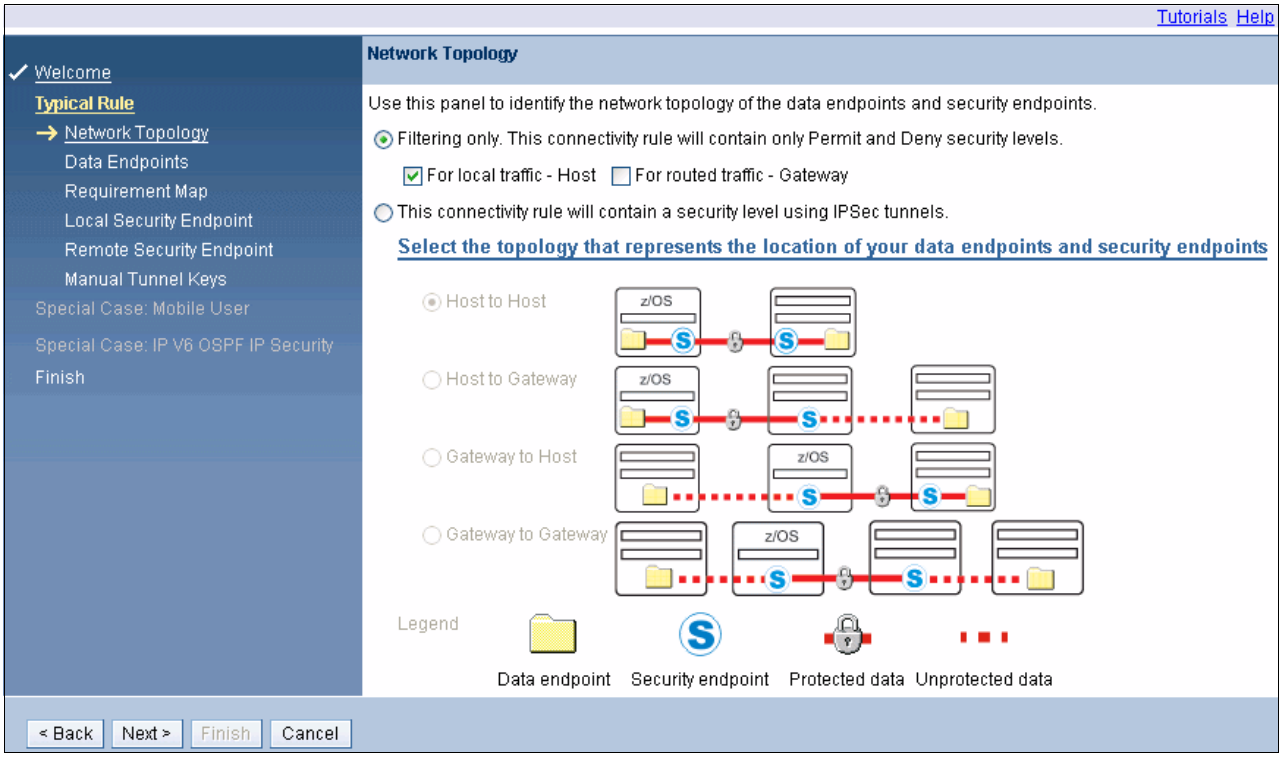


Figure 7-7 Typical connectivity rule, local traffic filtering only

Step 2: Create the TN3270 traffic filtering rule

Complete the following steps:

1. In the Data Endpoints panel (Figure 7-8), specify the IP addresses of the data endpoints. The IP addresses can be a single IP address, an IP address range, an IP subnet, or all IPv4 or all IPv6 addresses. In our scenario, the endpoints are defined as the single IP address TCPIPD stack (10.1.1.40) and Workstation (10.1.100.222). In this panel, we also define the name of our new rule, which is *TN3270_Filter*. To continue the configuration, click **Next**.

Important: The *Source data endpoint* must be the IP address of the stack that you are protecting.

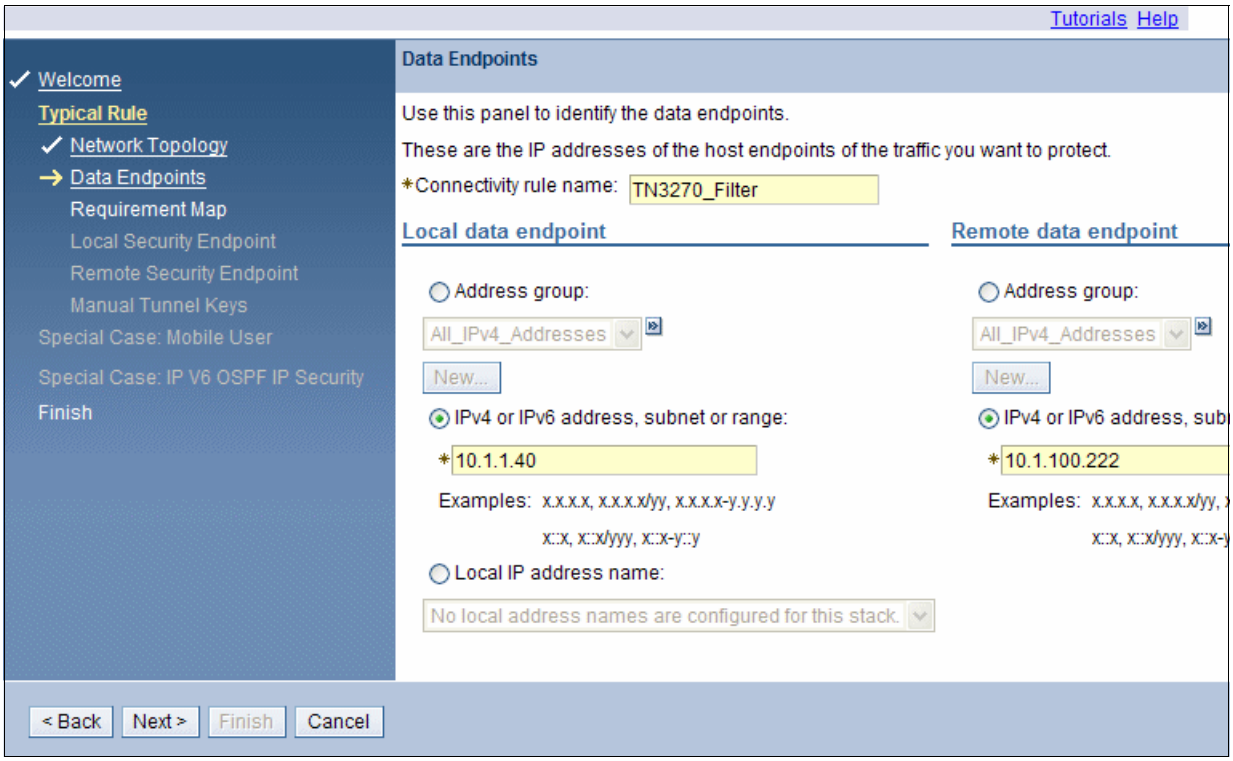


Figure 7-8 Setting up rule endpoints for filtering

2. In the New Connectivity Rule - Requirement Map panel (Figure 7-9), select an IBM-provided traffic descriptor from within the mappings table. We select `TN3270-Server`. Because we are creating a filter to permit TN3270 traffic wit port number 23, we set the Security Level to `Permit` for this TN3270 traffic.

   If you scan down the Mapping table, you can see you can see that `All_other_traffic` will be denied. If other types of traffic needed to be permitted, which is likely in most environments, you can work within this Mapping table to create the filtering environment your organization requires. Pay special attention to the order within the mapping table. Use the **Select Action** → **Move Up** and **Select Action** → **Move Down** options to place the rules in the desired order. Only the first match on a rule will be used.

3. Click **Next** to complete the new requirement map.

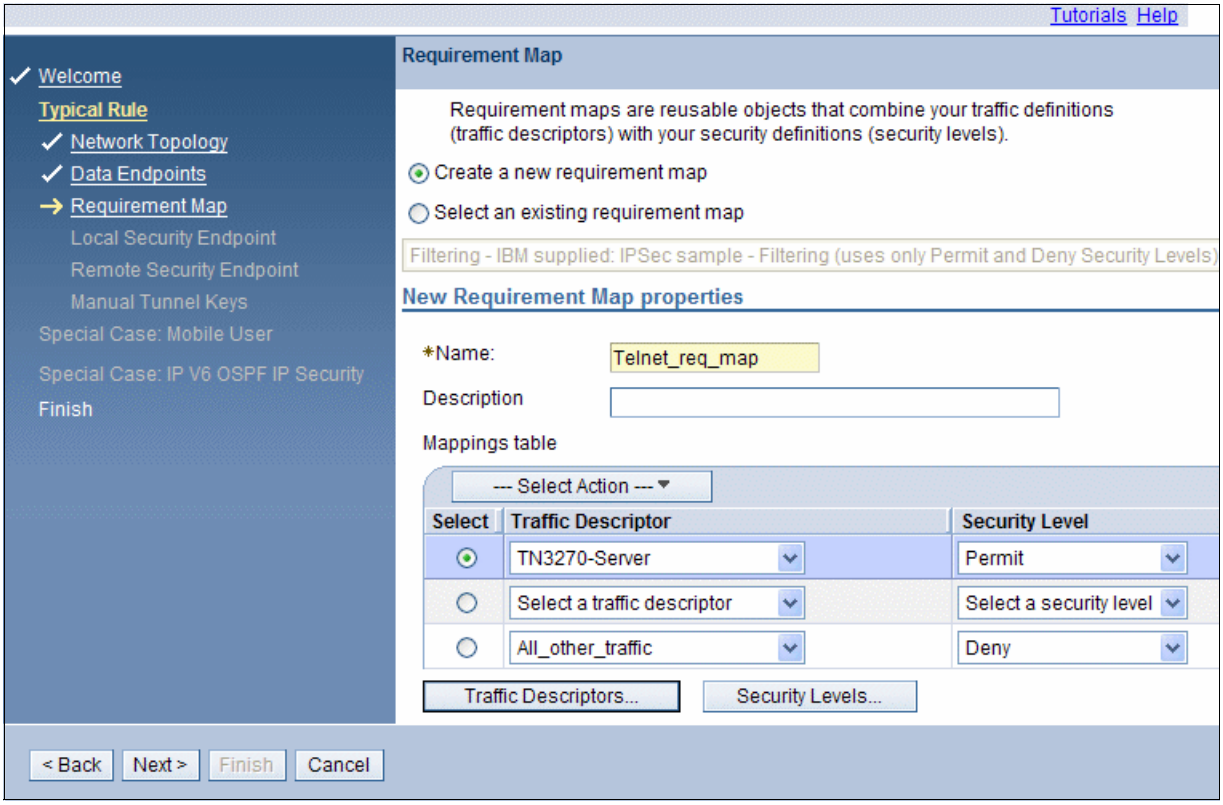


Figure 7-9 Requirement map mappings table for traffic and security

4. Most organizations do not need to change any options in the New Connectivity Rule - Finish panel. The default to not log filter matches is acceptable. Otherwise, you receive a message for every TN3270 packet that is permitted or denied. Click **Finish**, and you are returned to the IPSec Perspective for the stack as shown in Figure 7-10.

   This panel has several folder tabs. Because we are not doing IPSec, you can skip the Local Identity tab. Instead, click the Stack **Settings** tab.

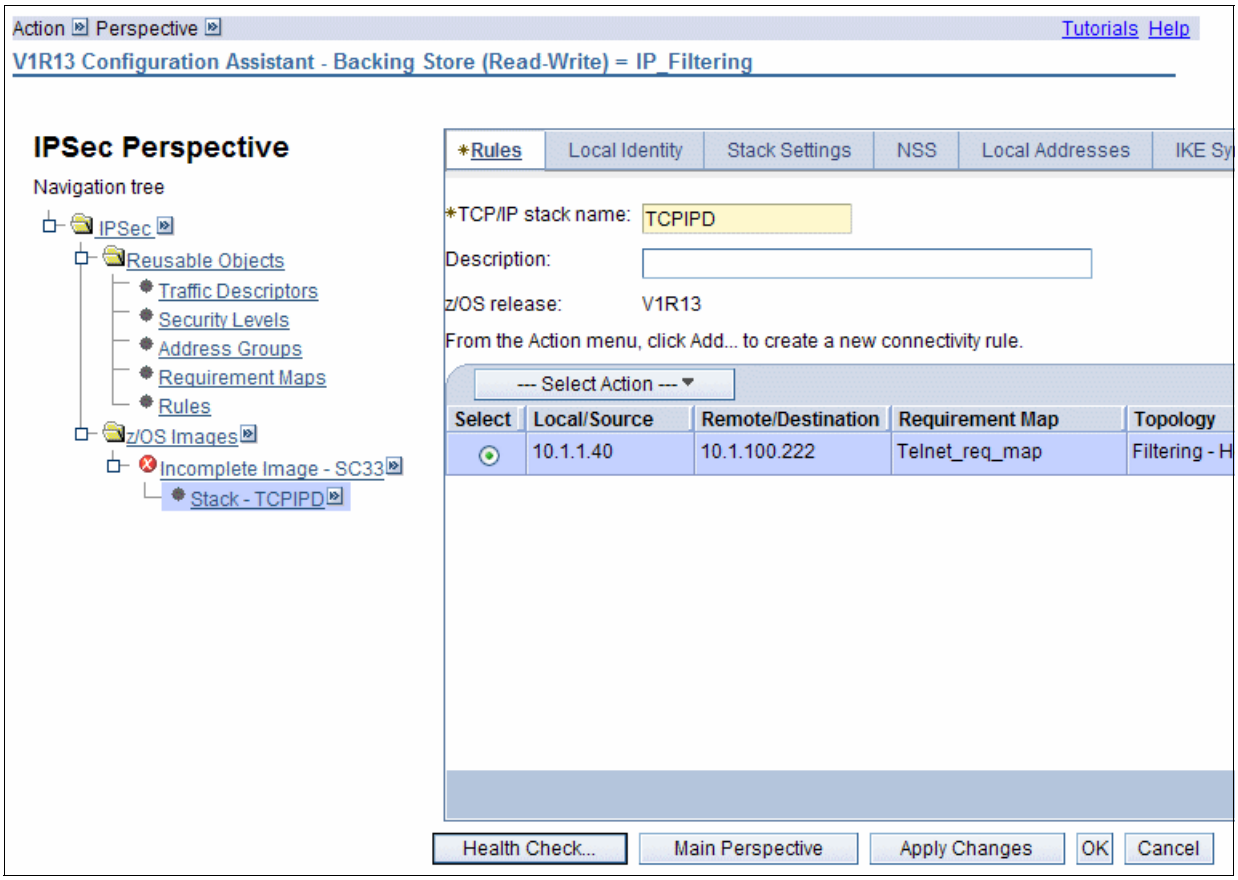


Figure 7-10 Returning to the IPSec perspective after creating connectivity rule

5. In the Stack Settings panel, shown in Figure 7-11, define the parameters that are related to the TCP/IP Stack. For our scenario, we select the following options in the main panel:
   - Enable Filter Logging Policy
   - Do not log implicit deny events
6. Click **OK**. z/OS image is still incomplete, so we select the image name (SC33 in our example). We do not make any changes for z/OS image level settings, and click **OK**.

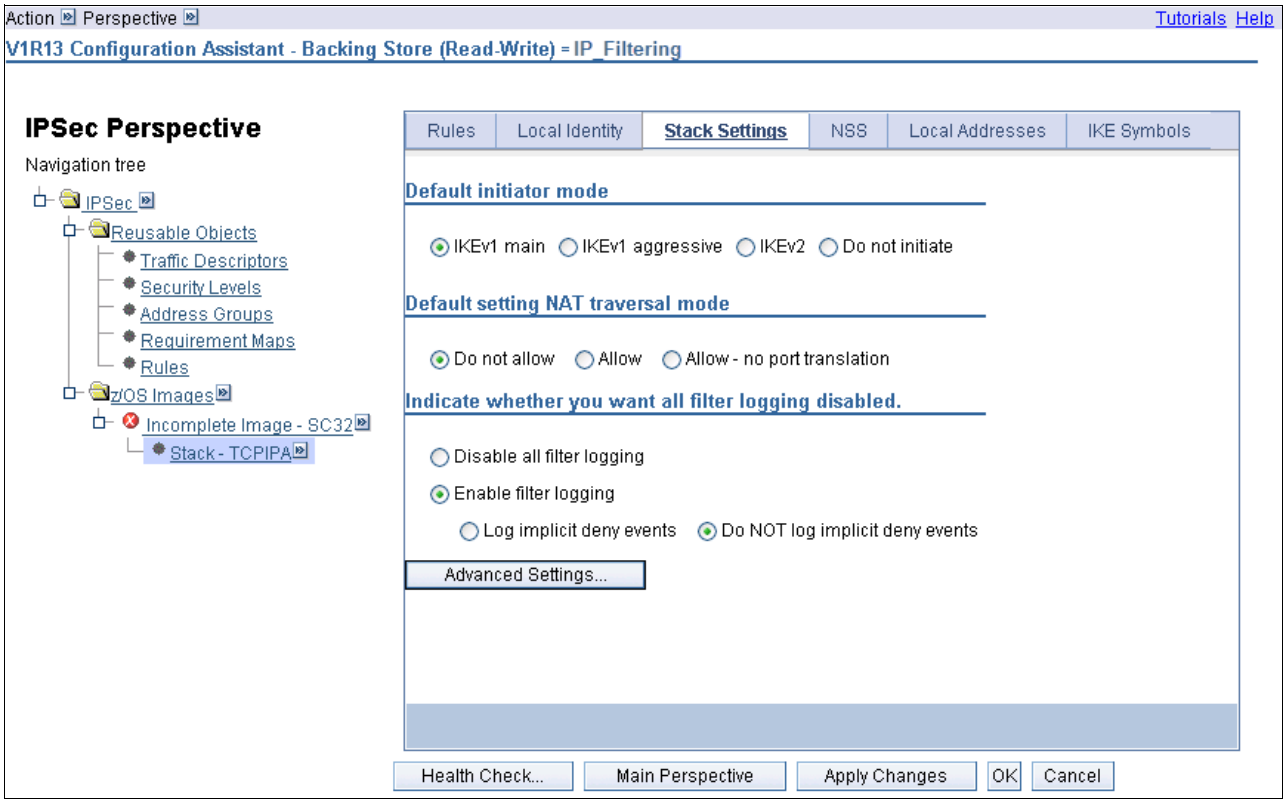


Figure 7-11 Stack level parameters

Step 3: Create the FTP traffic filtering rule

Now that you have completed the connectivity rule for Telnet, you can add another rule to allow FTP traffic between system SC33, stack TCPIPD (10.1.1.40), and the IP Address Group named OurSystems. To create this new Connectivity rule, complete the following steps:

1. On IPSec Perspective panel, go to Configuration Assistant Navigation Tree, under Work with Reusable Objects, and select Address Groups, as shown in Figure 7-12. To create a new Address group, click **Select Action** → **Add**.

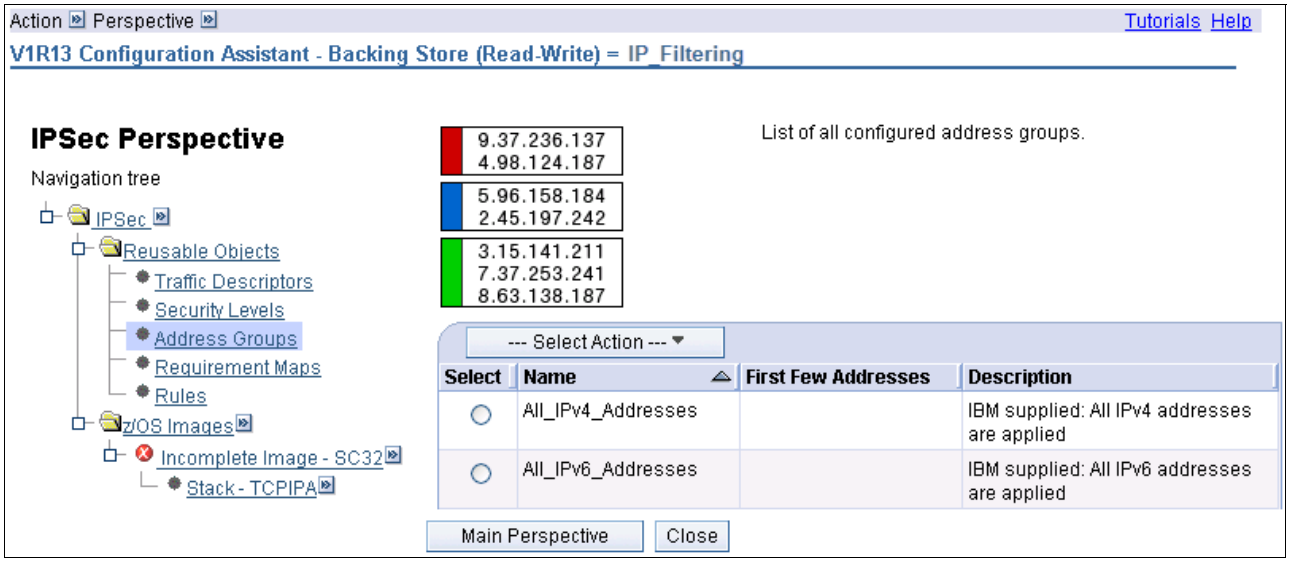


Figure 7-12 List of existing Address Groups

2. In the New IP Address Group panel, define the name of the group (`OurSystems`) as shown in Figure 7-13, and click a row of the table to include the IP address by typing it in directly as shown.

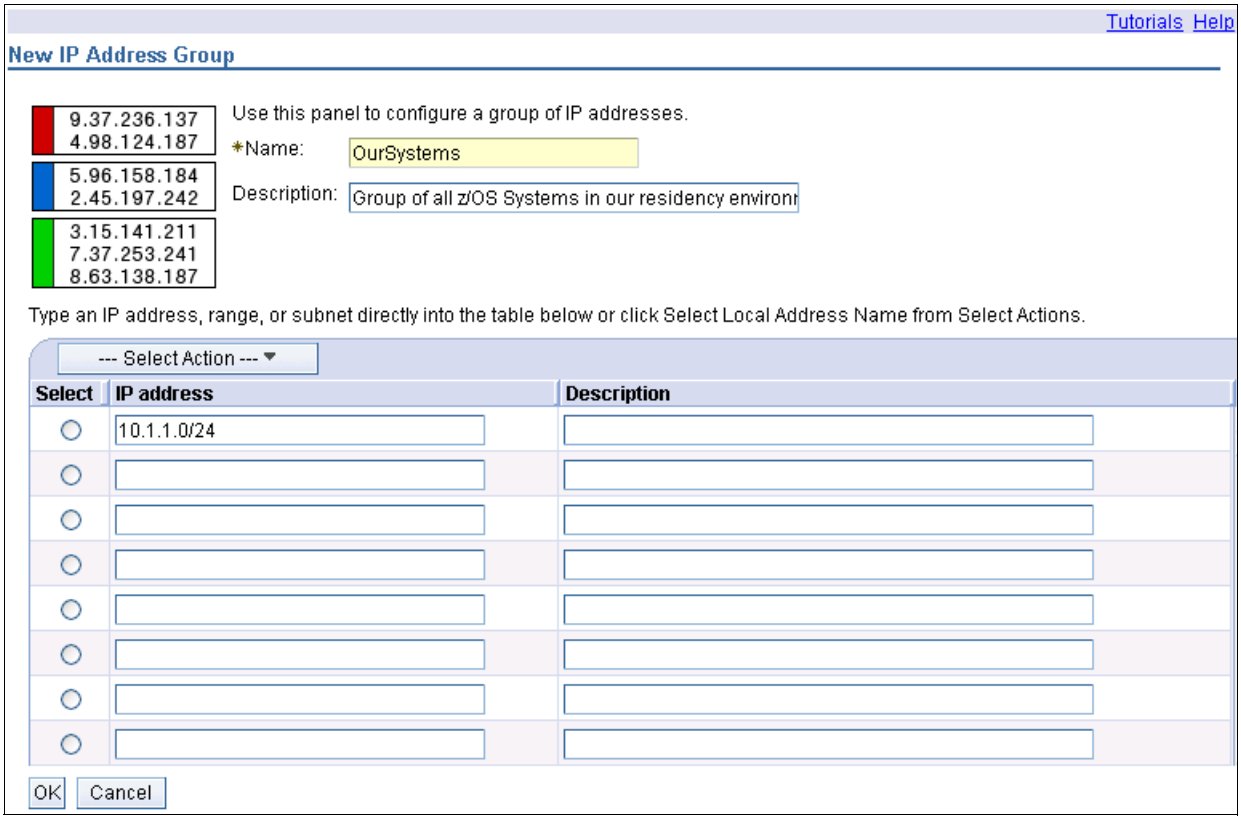


Figure 7-13 Creating a new address group

3. Click **OK** to include the new IP address group in the list of configured address group.
4. In the IPSec Perspective panel, select the stack name (TCPIPD in our environment) so that you are viewing your connectivity rules again. Click **Select Action** → **Add**. When the New Connectivity Rule - Welcome panel opens, select the **Typical** (Default) rule type and click **Next**, following the same steps as the first scenario (“Step 2: Create the TN3270 traffic filtering rule” on page 230).
5. In the New Connectivity Rule - Network Topology panel, select **Filtering Only** → **For local traffic - Host**. Then, click **Next**.

6. In the New Connectivity Rule - Data Endpoints panel (Figure 7-14), complete the following steps:
   a. Specify the IP addresses of local data endpoint (in our case, system SC33, which is 10.1.1.40)
   b. In the “Remote data endpoint” section, select **Address Group**, and then select the IP Address Group **OurSystems**.
   c. In the “Connectivity Rule Name” section, define the name of the new rule as `FTP_Filter`.
   d. Click **Next**.

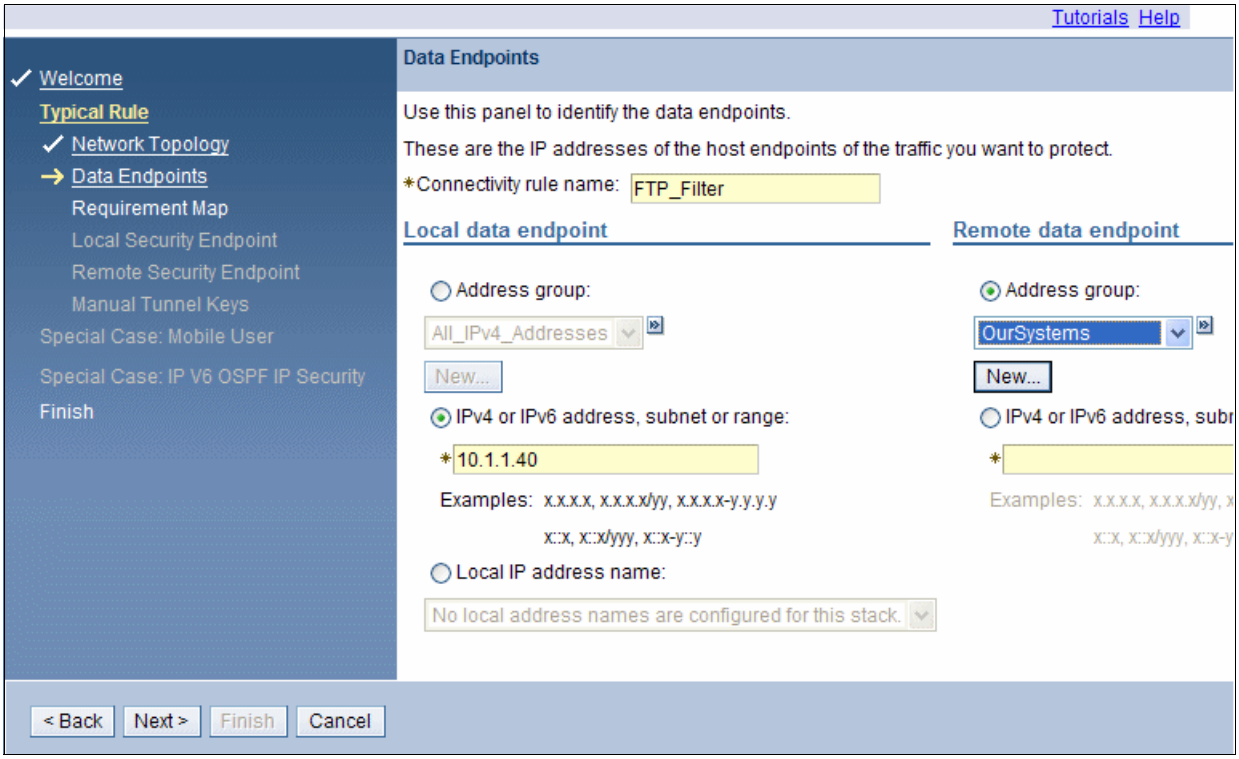


Figure 7-14 Data endpoints IP address definition

7. In the New Connectivity Rule - Requirement Map panel, you can create a new requirement map or select an existing one. For this scenario, we need to create a new requirement map.

   Under Traffic descriptor, we selected two rows:

   – FTP-Client
   – FTP-Server

   We set the security level for both to `Permit` as shown in Figure 7-15.

   Click **Next** after you complete the choices.

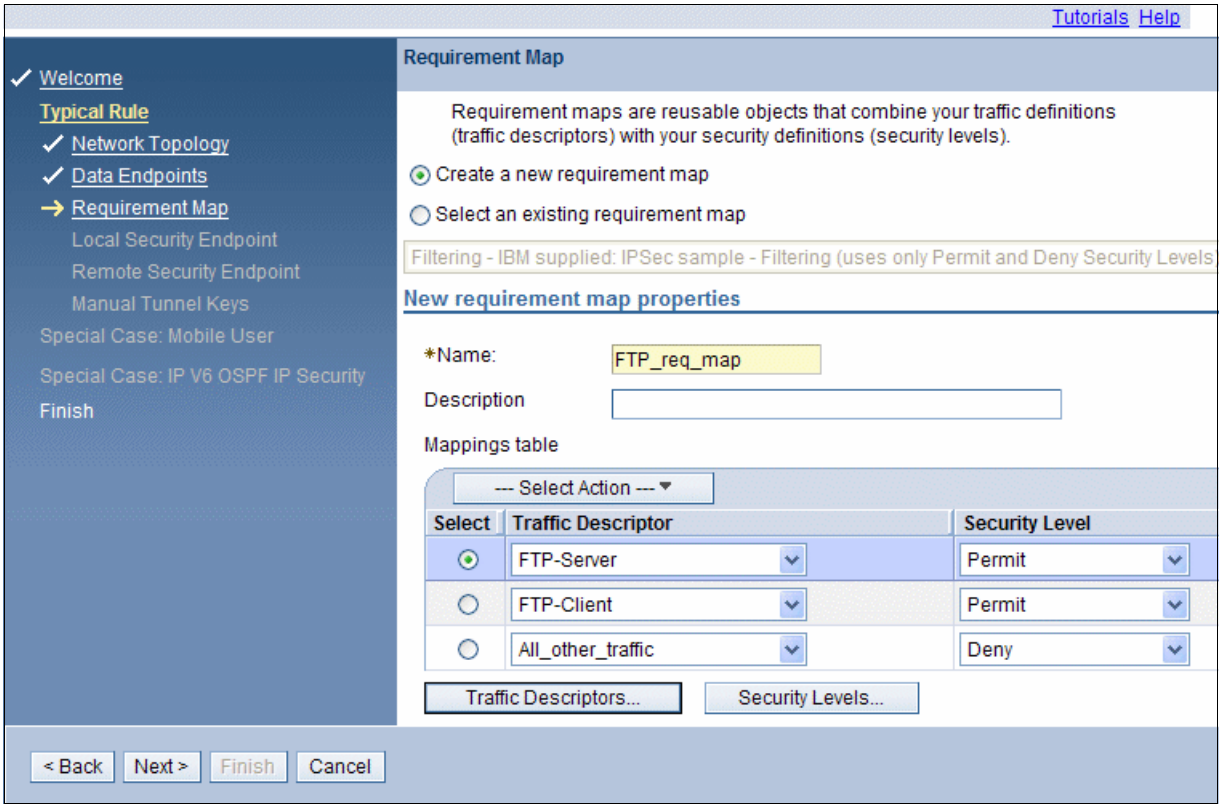


Figure 7-15 Defining the connectivity rule requirement map

8. Click **Next** to view the Finish panel. Again, logging all filter matches can create excessive message volume. If you are unsure of your rules and you are at the testing stage, you could select **Yes, log all filter matches.** Click **Finish**.

   When you have completed these steps, the z/OSMF Configuration Assistant returns to the IPSec Perspective panel and the new connectivity rule is listed, as shown in Figure 7-16.

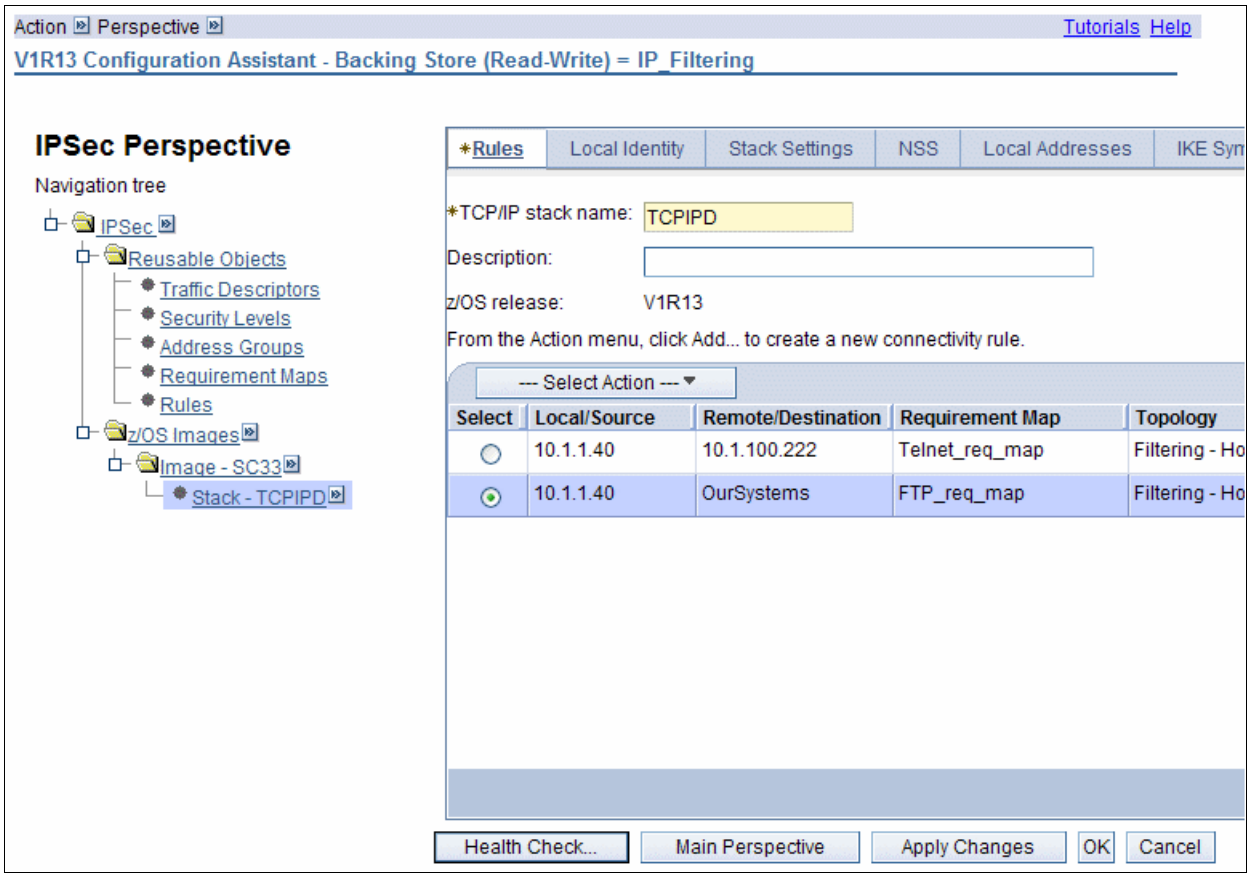


Figure 7-16 IPSec perspective with our new rules listed

Step 4: Create the basic services traffic

As mentioned earlier, there will be more services necessary than just Telnet and FTP. Before you generate the configuration files, it is important to add a new requirement map to permit the basic services in the environment (such as ping, resolver, DNS, OMPROUTE, and Service Connections Traffic between SC33 and all data endpoints). Complete the following steps:

1. In the Configuration Assistant Navigation tree (on the left), click **Requirement Maps** (under Work with Reusable Objects). If prompted, select **Apply Changes**. Then, in the Requirement Maps panel, click **Select Action** → **Add**.
2. On the Requirement Map panel, complete the following steps (see Figure 7-17 on page 239):
   a. Provide a name and description in the appropriate fields.
   b. Add ping, resolver, pagent policy server, DNS, and OMPROUTE by selecting the appropriate traffic descriptors (you might need to click **Select Action** → **Add Row** to define more than three traffic descriptors).
   c. Set the security level to reach descriptor to `Permit` for all entries.

d. Click **OK** when you are done.

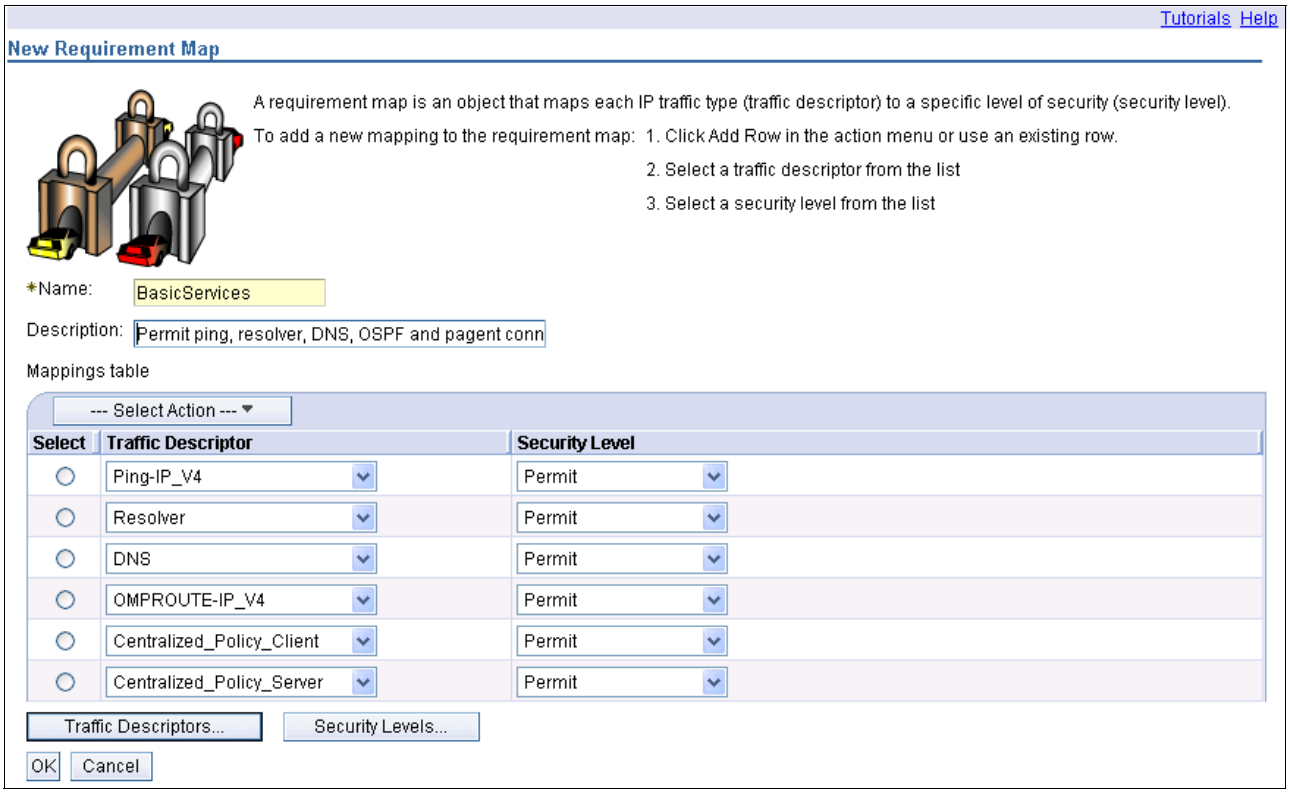


Figure 7-17 Add your required basic services to a requirement map

3. In the Navigation Tree, select your stack so that you are now looking at the IPSec Perspective panel stack view. You can now click **Select Action** → **Add** to add a new connectivity rule to permit all your basic services IP traffic. Use the same steps that you used to create the Telnet and FTP connectivity rule, as follows:

   a. In the Configuration Assistant Navigation Tree, select **Stack - TCPIPD** to open the TCPIP Stack panel with the list of connectivity rules for stack TCPIPD. Click **Select Action** → **Add** to create the new connectivity rule.

   b. In the New Connectivity Rule - Welcome panel, select **Typical** and click **Next**.

   c. In the Network Topology panel, select **Filtering Only**, and click **Next**.

   d. In the Data Endpoints panel, specify the **All IPv4_addresses** for both remote and local endpoints. For your own needs, you might want to select specific addresses that can be used for basic services. Click **Next**.

   e. In the Select Requirement Map panel, select **Select an existing requirement map** option, and select the policy we have just created (`BasicServices`), and click **Next**.

   f. Click **Finish** at the New Connectivity Rule - Finish panel, unless you want to change logging options.

   g. Back in the IPSec Perspective panel, move the BasicServices Rule to the top of the list by using the **Select Action** → **Move Up** button, because you want BasicServices traffic to flow between endpoints in all cases. The most commonly encountered rules should be closest to the top from an efficiency perspective.

Figure 7-18 shows the results.

> **Tip:** The IP addresses in a packet are compared with the IP addresses of the data endpoints of the connectivity rules in the order in which those rules appear in the table. The last rule to be applied must be the **`Deny All`** rule.

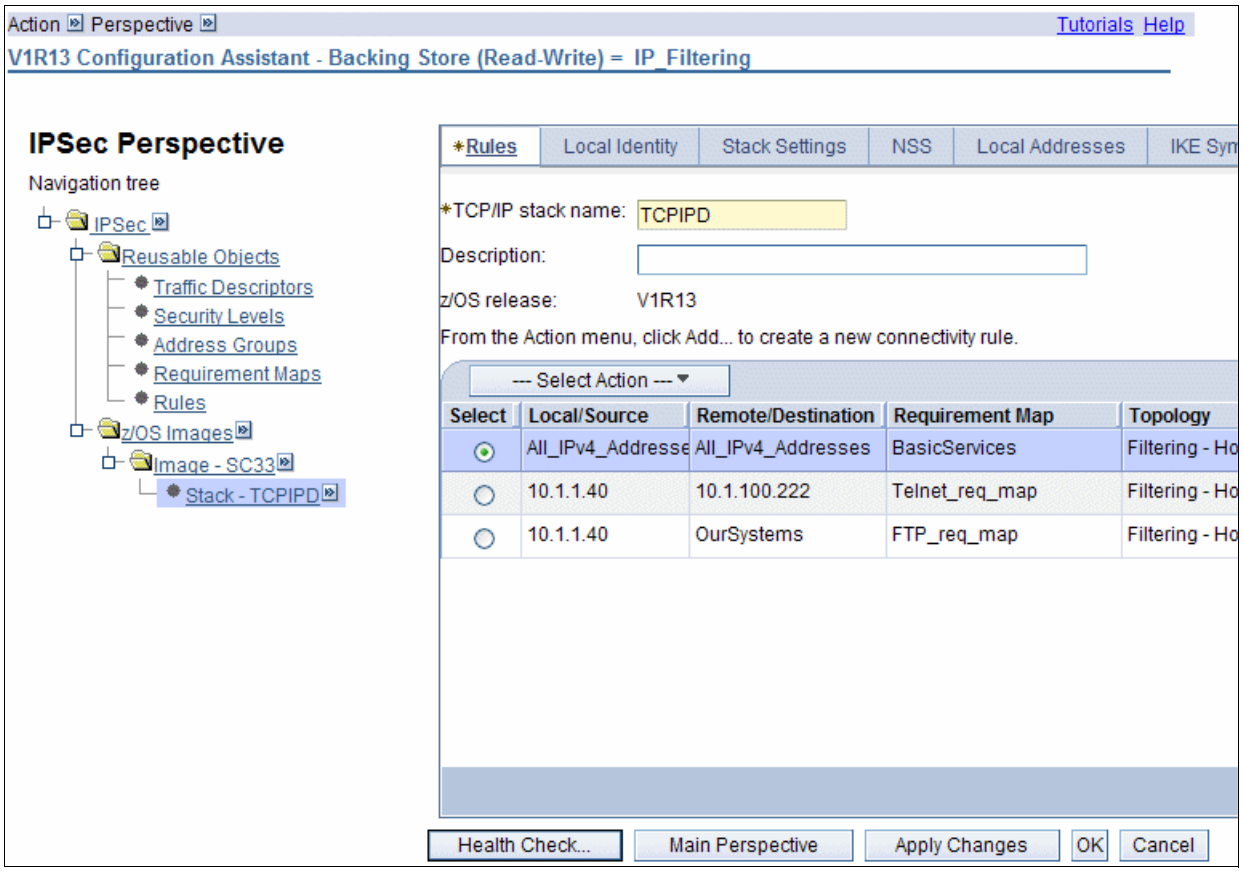


Figure 7-18 IPSec perspective panel with the new connectivity rules

After you complete the IP filtering configuration, click **Health Check** to determine if the TCP/IP stack contains configuration errors or an inappropriate combination of rules. Click **Close**.

Step 5: Install the configuration files

With IP filtering rules complete, install the configuration files as follows:

1. On the IPSec Perspective panel, under the Configuration Assistant Navigation Tree, click the small symbol to the right of the z/OS image name (Image-SC33, in our environment), and select **Install Configuration Files**. Click **Apply Changes**, if prompted.

> **Tip:** The word *incomplete*, in red, is displayed only during a new image configuration. If this is the case, a wizard opens and asks whether this image has Dynamic IP tunnels. Select **No** and click **Next**. The IP filtering for our scenarios do not use Network Security Services (NSS). Therefore, on the NSS Settings panel, click **Next**.

2. Select the policy and click **Select Action** → **Install**, as shown in Figure 7-19.

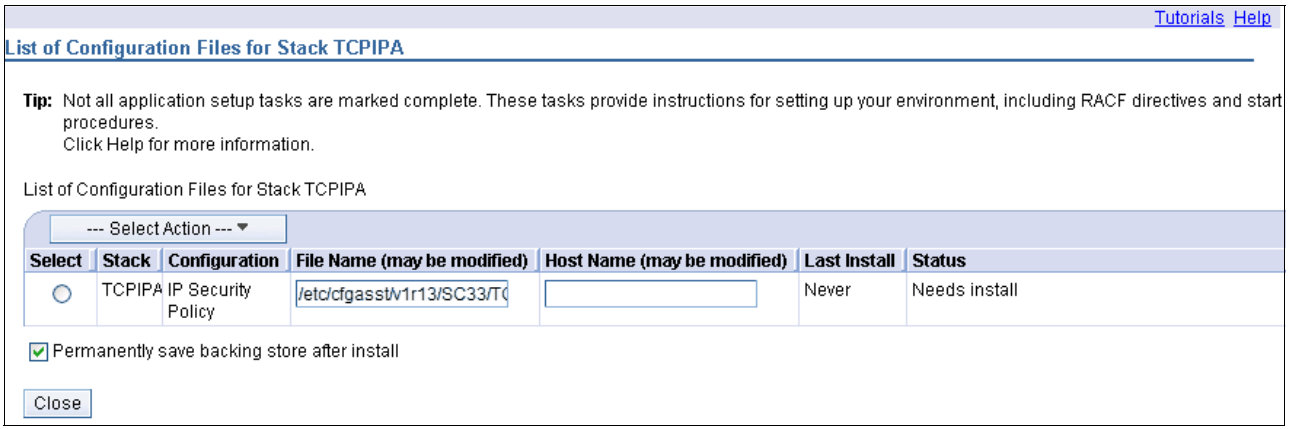


Figure 7-19 Configuration Files installation panel

3. On the Install File panel (Figure 7-20), specify the file name and select the **Save to disk** option to save the policy to z/OS local file system. Another option is to use FTP to move the policy to another z/OS image. In that case, specify IP address, port number, user ID, and password. Click **Go**. The result is displayed at the bottom of the same panel. Click **Close**. You may optionally leave a comment and then click **OK**.

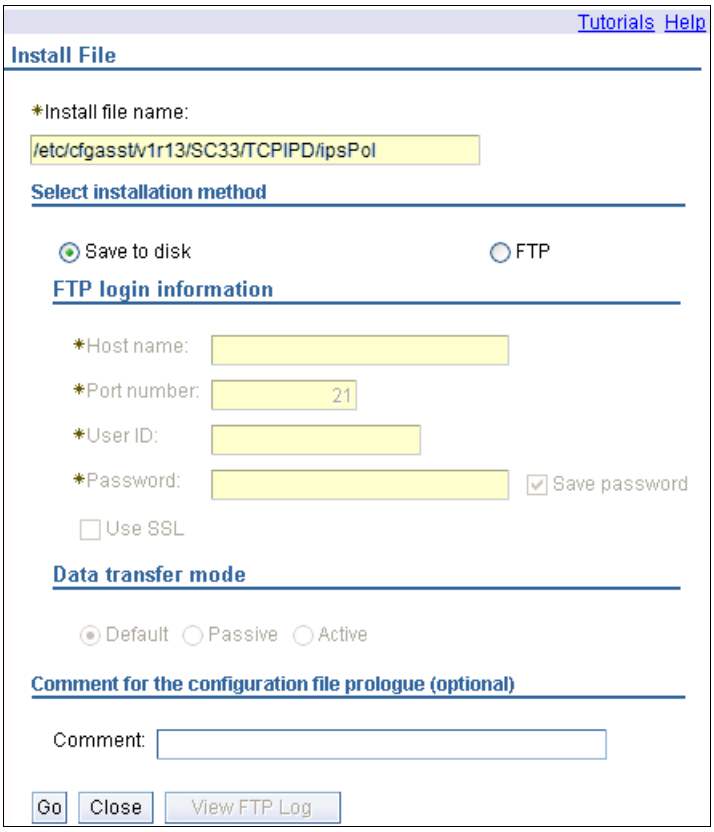


Figure 7-20 FTP process to install the configuration files

4. To verify the newly created configuration files, select the configuration file from the list and click **Select Action** → **Show Configuration File**. Example 7-2 on page 242 shows a small sample of the resulting file contents. Click **Close** after viewing your policy file.

Example 7-2 IPSec configuration file

```
##
## IPSec Policy Agent Configuration file for:
## Image: SC33
## Stack: TCPIPD
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 13
## Backing Store = IP_Filtering
## FTP History:
## 2011-07-19 00:31:33 : Save To Disk
##
## End of Configuration Assistant information

IpGenericFilterAction Permit__LogNo
{
IpFilterAction Permit
IpFilterLogging No
}

IpGenericFilterAction Deny__LogNo
{
IpFilterAction Deny
IpFilterLogging No
}

IpService Ping-IP_V4
{
Protocol ICMP
Type 8
Code Any
Direction BiDirectional
Routing Local
}

IpService Ping-IP_V4~1
{
Protocol ICMP
Type 0
Code Any
Direction BiDirectional
Routing Local
}

IpService Resolver
{
Protocol TCP
SourcePortRange 1024 65535
DestinationPortRange 53
Direction BiDirectional OutboundConnect
Routing Local
}
...
```

Step 6: Update the Policy Agent configuration file

Complete the following steps?

1. Update the policy agent configuration for the specific stack. As shown in Example 7-3, we update `/etc/pagent.sc33.tcpipd.conf` file and include the IPSecConfig statement with the name of policy file we have just created.

   Example 7-3 /etc/pagent.sc33.tcpipd.conf

   ```
   IPSecConfig /etc/cfgasst/v1r13/SC33/TCPIPD/ipsPol
   ```

2. Next, start PAGENT or refresh PAGENT (with F PAGENT,REFRESH command). The following message is displayed:

   ```
   EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD: IPSEC
   ```

Now, PAGENT is running with the policies applied, and you can verify that the policies are working as expected.

Step 7: Verification of IP filtering configurations

Execute the following tasks to verify IP filtering configurations:

1. Log on to System SC33 using TN3270 (should succeed)
2. Use FTP to copy to System SC33 from a remote workstation (should fail)
3. Use FTP to copy from System SC32 to SC33 (should succeed)

Log on to System SC33 using TN3270 (should succeed)

In our scenario, from workstation 10.1.100.222, we configure a TN3270 session to TCPIPD (10.1.1.40) on SC33, port 23. The connection to TSO (the default application) on SC33 established successfully, as shown in Example 7-4. Any attempt to establish a session from another workstation fails.

Example 7-4 Establishing connection from TN3270 client 10.1.100.222 to TN3270D server 10.1.1.40

```
D TCPIP,TN3270D,T,CONN,CONN=70
EZZ6065I TN3270D CONN DISPLAY 726
  CONNECTED: 20:55:14  07/18/2011  STATUS: SESSION ACTIVE
  CLIENT IDENTIFIER FOR CONN: 00000070   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.222..1371
    DESTIP..PORT: ::FFFF:10.1.1.40..23
```

Use FTP to copy from System SC32 to SC33 (should succeed)

As defined in the FTP connectivity rule, any z/OS system in our test environment should be able to create an FTP connection with FTP Server on system SC33, because we did set up IP filtering rules allowing it (see Figure 7-5 on page 226). To verify this rule is in place, we tried to initiate an FTP connection from system SC32, stack TCPIPD.

Example 7-5 Successful FTP connection from SC32 to SC33

```
IBM FTP CS V1R13
 FTP: using TCPIPC
 Connecting to:   10.1.1.40 port: 21.
 220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 21:00:28 on 2011-07-18.
 220 Connection will close if idle for more than 5 minutes.
 NAME (10.1.1.40:CS02):
```

We confirmed it is working as shown in Example 7-5 on page 243 and Example 7-6.

Example 7-6 Successful FTP connection from SC32 to SC33: netstat conn display

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS V1R13 TCPIPD 732
USER ID  CONN     STATE
FTPDD1   00000076 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00000085 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.30..1032
```

Use FTP to copy to System SC33 from a remote workstation (should fail)

From the remote workstation, which has an IP address (10.1.100.222) that does not belong to the IP Address Group OurSystem (10.1.1.0/24), FTP is not allowed into system SC33 stack TCPIPD, as shown in Example 7-7.

Example 7-7 Attempt to connect SC33 from the workstation 10.1.100.222 fails

```
C:\>ftp 10.1.1.40
> ftp: connect :Unknown error number
ftp> ls
Not connected.
ftp>
```

7.2.4 Problem determination

The following tools and documentation can help you with 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.
- IPSEC command: Use to review the implemented policies.
- TCP/IP profile output
- POLICY config file output

7.2.5 For additional information

For additional information about these topics, consult the following resources:

- *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

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. We discuss the following topics.

Section	Topic
8.1, “IPSec description” on page 246	Basic concepts of IPSec and Virtual Private Network (VPN)
8.2, “Basic concepts” on page 247	Key IPSec components
8.3, “Current IPsec support” on page 251	Explains new features in IPSec as they are added in each release
8.4, “Working with the z/OS Communications Server Network Management Interface” on page 262	A brief explanation of the Network Management Interface (NMI) API that network management products can take advantage of in IPSec implementation
8.5, “How IPSec is implemented” on page 264	How to implement z/OS Communications Server components required by IP security
8.6, “Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2” on page 285	How to implement IPSec between two z/OS systems in Pre-shared Key Mode
8.7, “Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1” on page 306	How to implement IP security between two z/OS images using Dynamic Tunnels and RSA mode authentication
8.8, “Additional information” on page 323	References to additional information about IP Security (IPSec)

Modes: For Internet Key Exchange (IKE), the term RSA mode is the same as Digital mode. The Configuration Assistant panels refer to it as *Digital signature mode*. In this book it is referred as *RSA mode* in many places.

8.1 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 Enterprise Extender (EE) traffic can be protected too.

When IPSec is used to protect EE connections, IP processing optimizes the EE path length. If the system is configured to use the zIIP processor, EE makes use of the zIIP.

The IPSec standards have also been structured so that they can accommodate newer, more powerful cryptographic 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.

To verify which IPSec RFCs z/OS Communications Server has implemented, see Appendix E, “z/OS Communications Server IPSec RFC currency” on page 919.

Tip: IBM z/OS Communications Server IP filtering support is packaged with IP Security (IPSec) and is referred to as *integrated IP Security* 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. To configure IP filtering, you must indicate that you are configuring IPSec in the z/OSMF Configuration Assistant. For more information, see Chapter 7, “IP filtering” on page 217.

8.2 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. 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 configured statically and are managed by a security administrator manually. Manual tunnels are not commonly implemented.
- Dynamic IPSec tunnels: The security parameters are negotiated, and the encryption keys are generated dynamically using IKE.

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 Encapsulating Security Payload (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 Authentication Header (AH) protocol is the IPSec-related protocol that provides authentication. The Encapsulating 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, as shown in Figure 8-1 on page 248:

- Transport mode
- Tunnel mode

These modes tell IP how to construct the IPSec packet. Transport mode is used in host-to-host scenarios in which the two endpoints of the tunnel (that is, the security endpoints) are the same as the host endpoints (that is, the data endpoints). Tunnel mode is most frequently used when either endpoint of the tunnel is a router or firewall. With tunnel mode the security endpoints are separate from the data endpoints.

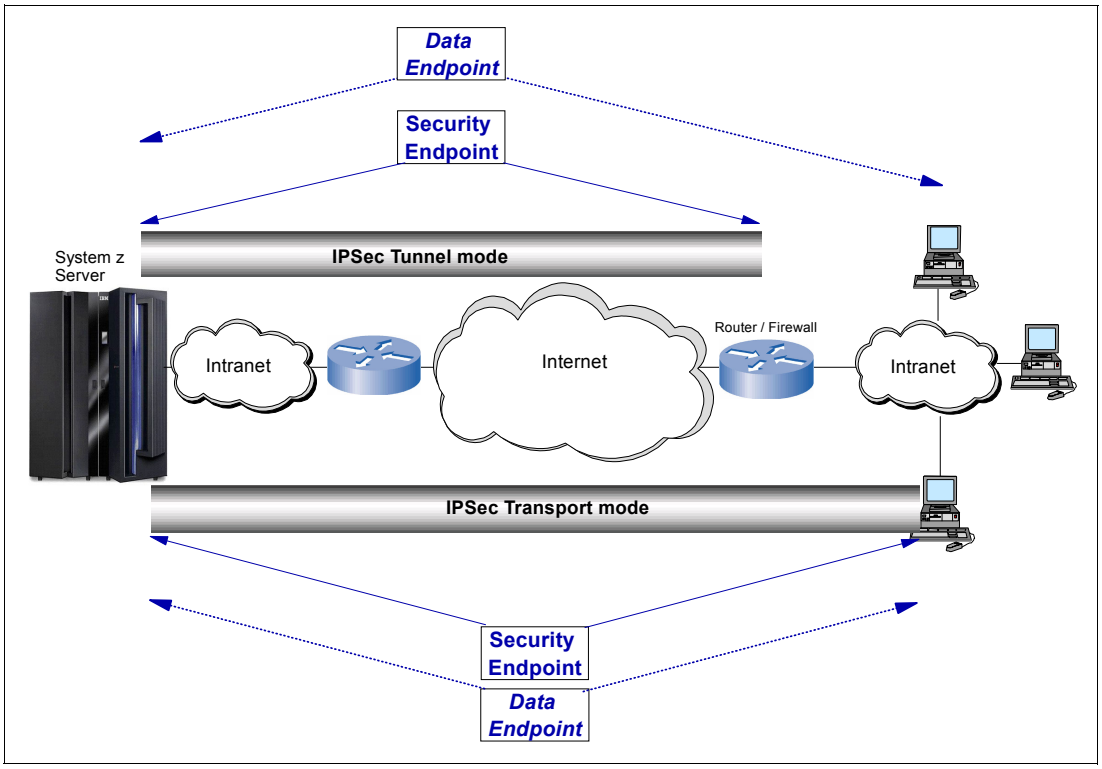


Figure 8-1 Two mode types: Transport mode and tunnel mode

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.

8.2.1 Key components

Three of the most important IPSec components implemented by z/OS Communications Server are as follows:

- RFC 4302: IP Authentication Header (AH) protocol, which provides for data authentication, IP header authentication, and data origin authentication
- RFC 4303: IP Encapsulating Security Payload (ESP), which provides for data authentication, data origin authentication, and data privacy (encryption)
- RFC 5996: The Internet Key Exchange (IKE), which provides protocols for automated encryption key management

In the following sections, we describe these components in more detail.

8.2.2 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 the following features:

- 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

8.2.3 IP Encapsulating Security Payload protocol

Encapsulating Security Payload (ESP) provides additional protection beyond (or in addition to) AH, as follows:

- 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.

8.2.4 Internet Key Exchange protocol: Pre-shared key and RSA signature mode

All of the concepts that we have discussed thus far are integral to building secure IPSec tunnels between two endpoints. Remember that there are two types of secure tunnels:

- Manual IPSec tunnels are implemented with encryption keys that are configured manually. Although simple to establish, manual tunnels are not considered entirely secure because manually configured keys can be compromised easily. Also, with manual tunnels, IPSec keys cannot be changed without deactivating and reactivating the secure tunnels, thus opening up a window of vulnerability during the data transfer.
- Dynamic IPSec tunnels are implemented with encryption keys that can be renegotiated dynamically. Inadvertent or intentional breech of the security keys with dynamic tunnels is nearly impossible. Thus, dynamic IPSec tunnels are preferred for robust security implementations. The SAs and the encryption keys are negotiated using the IKE protocol.

The 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 and is required when implementing IPSec dynamic tunnels. Key exchange, defined in IKE, is normally a multi-step process, as described here and as shown in Figure 8-2 on page 250:

1. First, the partners establish a secure logical connection, an SA, and decide on security parameters such as encryption, hashing algorithms, and authentication methods (IKE SA). This connection is sometimes referred to as the *phase 1 tunnel* or *IKE tunnel*.
2. After the appropriate security parameters are negotiated, the partners set up a second SA for the actual data transfer (IPSec or child SA). This connection is sometimes referred to as the *phase 2 tunnel* or *IPSec tunnel*.
3. Thereafter, the SAs and the session keys are renegotiated periodically. The IKE daemon uses the IP security policies that you define in the policy agent (PAGENT) and manages the keys dynamically that are associated with dynamic IPSec VPNs.

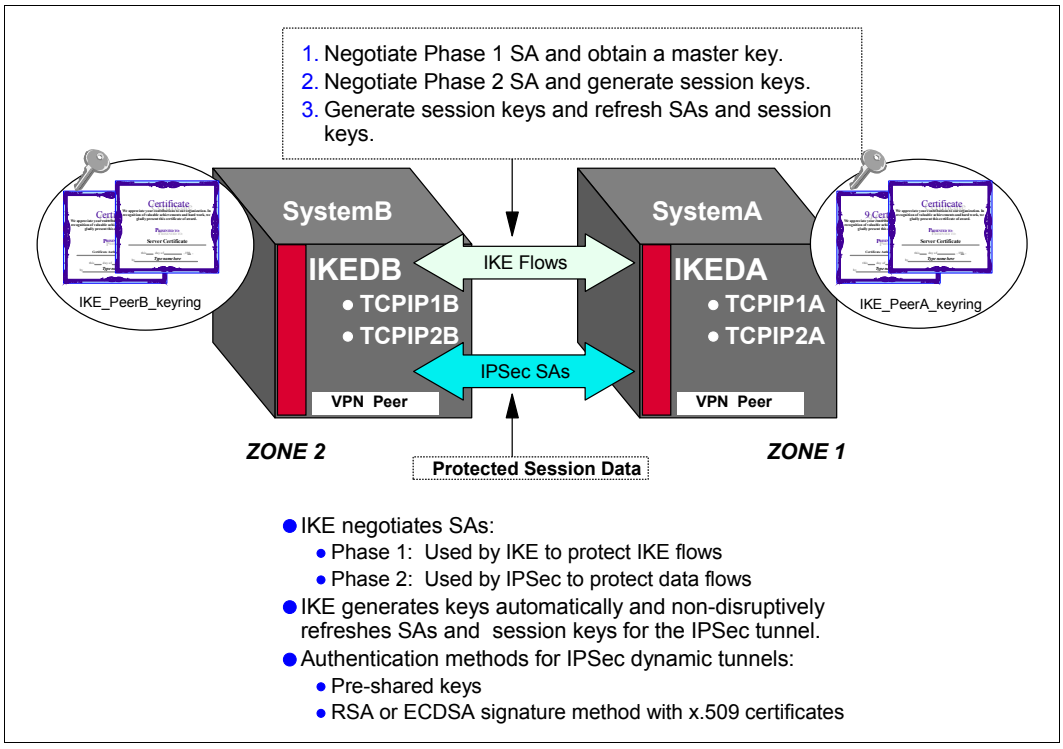


Figure 8-2 IKE concepts

As Figure 8-2 depicts, there are two methods of authenticating the IPSec peers when using dynamic tunnels:

- Authenticate the IPSec partner with a pre-shared key
- Authenticate the partner using digital signature mode

The term *pre-shared key* can be considered a misnomer, because the so-called “keys” are more akin to secret “passwords” that are shared between the IPSec peers to authenticate the partner during the Phase 1 IKE exchange and to provide a value to the Diffie-Hellman exchange that produces a cryptographic key to protect and authenticate the Phase 1 IKE negotiations.

Digital signature mode authentication relies on x.509 certificate exchanges to provide verification of the trusted partner. z/OS supports two digital algorithms: RSA and ECDSA. The certificates are stored in the IKE key rings of the peers, as shown in Figure 8-2. The local identity of an IPSec peer must be configured in the IPSec policy, and it must represent an identity established in x.509 certificate on the local IKE key ring. The remote identity of an IPSec peer must also be configured in the IPSec policy, and it must represent an identity established in the x.509 certificate presented by the remote IKE peer during Phase I negotiations.

When implementing digital signature mode on z/OS, certificate management services can be provided at the local z/OS IKE daemon (IKED) or by a Network Security Services daemon (NSSD) running locally or in another, more secure network zone on z/OS. IKED supports local digital signature mode for IKE version 1 (IKEv1) negotiations only, whereas NSSD supports both IKEv1 and IKE version 2 (IKEv2). Consult Chapter 9, “Network Security Services for IPSec clients” on page 325 for specific configuration examples and a description of how IKED takes advantage of the Network Security Services.

8.3 Current IPsec support

IBM has added a number of features to the IP security component of *z/OS Communications Server.* This section describes these features.

8.3.1 IKE version 2 (IKEv2) support

The *z/OS Communications Server* supports IKEv2 in addition to IKE version 1(IKEv1). IKEv2 has better performance and operational characteristics than IKEv1. Many government agencies expect the vendors who do business with them to use IKEv2 to establish secure communications with them.

Differences between IKEv1 and IKEv2

Consider the following differences:

- IKEv2 does not interoperate with IKEv1.
- IKEv1 supports AH in combination with ESP, but IKEv2 does not.
- One IKE daemon can support both IKEv1 and IKEv2.
- One IPSec policy file can contain both IKEv1 and IKEv2 policies.
- Each TCP/IP stack can support tunnels activated by IKEv1 and IKEv2.
- Security Association (SA) terminology differences:
  - IKEv1
    - Phase 1 SA
    - ISAKMP SA
    - Phase 2 SA
    - IPsec SA
  - IKEv2
    - IKE SA
    - Child SA
- The negotiation modes used by IKEv1 and IKEv2:
  - IKEv1
    - Main
    - Aggressive
    - Quick
  - IKEv2
    - Initial exchanges
    - Subsequent exchanges
- The encapsulation mode (tunnel or transport)
  - IKEv1: A negotiated attribute of the SA. Local value must match peer's value
  - IKEv2: Negotiated based on topology and user preference and SA. Local and peer can have different values

IKEv2 advantages compared to IKEv1

Advantages of IKEv2 are as follows:

- Requires fewer network flows in most cases compared to IKEv1.
- Can do rekeying without reauthentication.
- Has built-in dead-peer detection.
- Supports avoidance of duplicate or orphaned tunnels.
- Supports ECDSA signature mode.
- Has solved some of the problems that plagued IKEv1.

The Network Security Services (NSS) daemon (NSSD) provides all digital signature and certificate-related services required by IKEv2, including HTTP retrieval of certificates and certificate bundles and certificate revocation lists. This means that you must use NSS if you want to use digital signature mode authentication with IKEv2. In addition, most of the newest NSS digital signature services can also be used with IKEv1 on an optional basis.

The **ipsec** command can display, activate, refresh, and deactivate both IKEv1 and IKEv2 tunnels.

How to exploit IKEv2

If you intend to use digital mode with IKEv2, you must set up Network Security Services (NSS) first. See 9.2, “Configuring NSS for the IPSec discipline” on page 338 to set up NSS.

After NSS is set up, you need to use the configuration assistant to use IKEv2. In the IPSec perspective of the configuration assistant panel, select the stack in the left navigation tree, click **Stack Settings,** and then select the **IKEv2** radio button as shown in Figure 8-3.

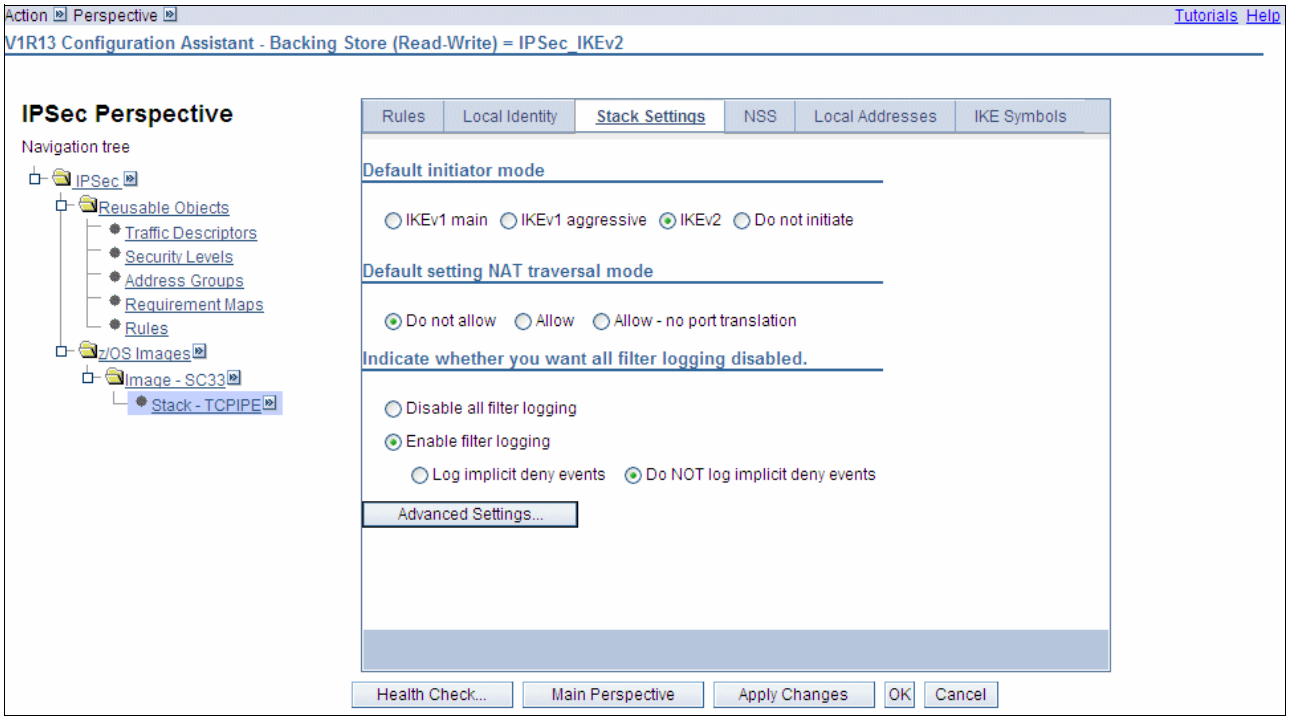


Figure 8-3 How to activate IKEv2 support

We activated the IKEv2 enabled policy for TCPIPE on SC33 and displayed the KeyExchange policy entries by issuing the following command:

```
pasearch -p TCPIPE -v k
```

Example 8-1 shows part of the output from this command.

Example 8-1 IKEv2 activated

```
KeyExchange Action:   All_Traffic_RSA
  Version:            3                 Status:            Active
  HowToInitiate:      IKEv2             HowToRespondIKEv1: Either
  AllowNat:           No                FilterByIdentity:  No
  HowToAuthMe:        DigitalSignature  ReauthInterval:    0
  BypassIpValidation: No                CertURLLookupPref: Tolerate
  RevocationChecking: Loose
  ConstrainSource:    0
   FromAddr:          192.168.1.40
   ToAddr:            192.168.1.40
  ConstrainDest:      0
   FromAddr:          10.1.100.222
   ToAddr:            10.1.100.222
  KeyExchangeOffer:   0
   HowToEncrypt:      3DES              KeyLength:         N/A
   HowToAuthPeers:    RsaSignature      DHGroup:           Group1
   HowToAuthMsgs:     SHA1
   HowToVerifyMsgs:   HMAC_SHA1_96      PseudoRandomFunc:  HMAC_SHA1
   RefLifeTmPropose:  480
   RefLifeTmAcptMin:  240               RefLifeTmAcptMax:  1440
   RefLifeSzPropose:  None
   RefLifeSzAccept :  None
  Policy created: Tue Jul 19 09:08:23 2011
  Policy updated: Tue Jul 19 18:02:48 2011
```

8.3.2 IPSec support for certificate trust chains

The IKE protocol for authentication requires the peers to exchange digital certificates. IKED will validate the certificate by checking it against the certificate of the certificate authority (CA) that has issued it. In most cases, IKED finds the certificate of the issuing CA in its key ring. For cases where the issuing CA certificate is not present in the keyring, IPSec also supports *certificate trust chains* for certificate validation.

What is a certificate trust chain

Trust is an important concept in digital certificates. Each organization or user must determine which CAs can be accepted as trustworthy. A user of a security service requiring knowledge of a public key generally needs to obtain and validate a digital certificate containing the required public key. Receiving a digital certificate from a remote party does not give the receiver any assurance about the authenticity of the digital certificate.

To verify that the digital certificate is authentic, the receiver needs the public key of the certificate authority that issued the digital certificate. If the public key user does not already hold an assured copy of the issuer's public key, then the user needs a copy of the issuer's digital certificate to obtain that public key. In general, a chain of multiple digital certificates might be needed, comprising a digital certificate of the public key owner (the end entity) signed by one CA, and optionally one or more additional digital certificates of CAs signed by other CAs.

Figure 8-4 shows a simple chain of trust.

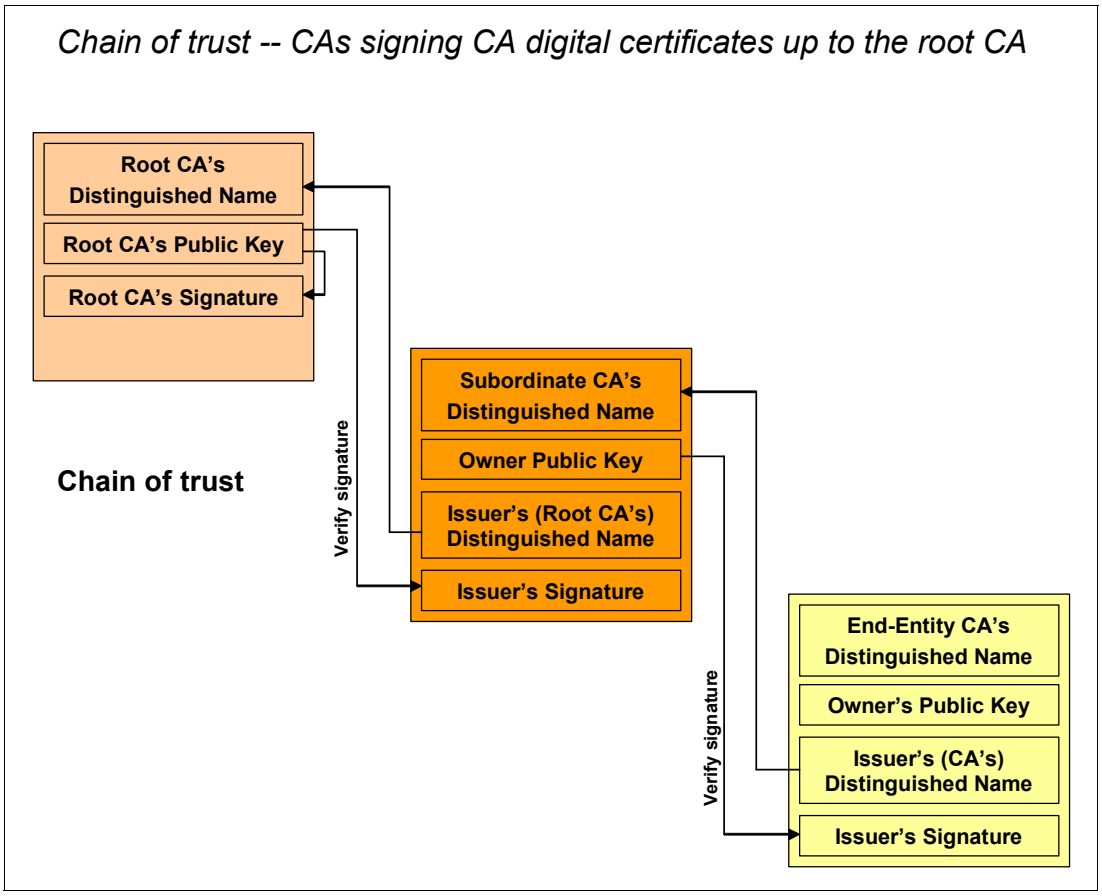


Figure 8-4 Chain of trust: CAs signing CA digital certificates up to the root CA

The figure describes the following information:

- At the bottom is a digital certificate that has been is issued and signed by a subordinate certificate authority. This certificate represents an end-entity such as a person, application, and so on.
- In the middle is the certificate of the subordinate CA (the issuer), which has been delegated the responsibility to issue certificates on behalf of the root certificate authority.
- At the top is the certificate of the root authority that signed the intermediate CA's certificate.
- Each certificate has a public/private key pair that is bound to its identity (the name of a person, company or an IP address). The public keys are present in the certificate and the associated private keys are stored securely in the owner's own keyring.

Implementation

Figure 8-5 shows how this is implemented. As response to a certificate request, the peer sends its end-entity certificate and any intermediate subordinate CA certificates that the peer IKED might not have its key ring in the trust chain. All the certificate authorities in the trust chain are considered when NSS is creating or verifying a signature for certificate authorities that are in the key ring.

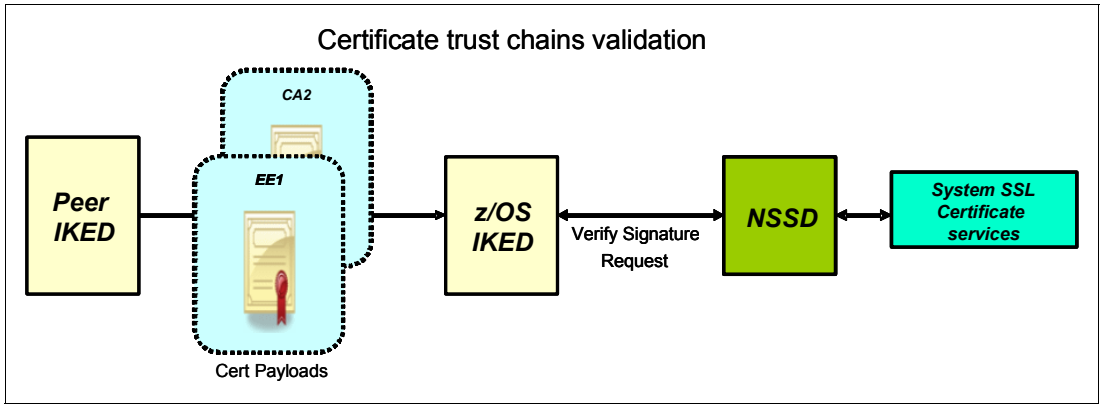


Figure 8-5 Certificate trust chains validation

This conforms to RFC 5996 and RFC 4945, which state that the first certificate payload can contain CA certificates in the trust chain.

Restriction: Certificate trust chains and certificate revocation checking is applicable only to IKEv1 and IKEv2 configurations that use NSS certificate services.

8.3.3 IPSec support for certificate revocation lists

A certificate revocation list (CRL) is a time-stamped list of revoked certificates that is signed by a certificate authority. Certificates can be revoked for one of the following reasons:

- Private key has been compromised
- Termination of an affiliation employment or membership
- Certificate no longer valid for stated purpose

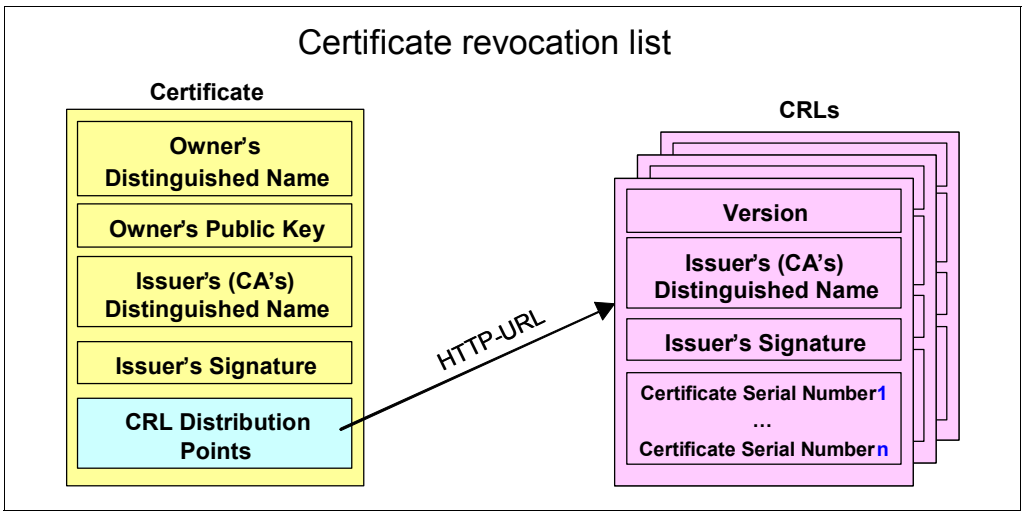


Figure 8-6 Certificate revocation lists (CRLs)

IPsec insures that all certificates are valid when it verifies a signature sent in the IKE flow. The NSS server supports the checking of certificate revocation lists (CRLs) when verifying a signature. CRLs are obtained by NSSD using a certificate's HTTP CRL distribution point. The NSS server also supports the retrieval of certificate bundles, which can also contain a CRL. If a CRL cannot be retrieved using the CRLDistributionPoints extension of a certificate, the NSS server looks for a CRL in any certificate bundle that has hash and URL information that matches what is provided by the IKED network security client. IKED obtains certificate bundle hash and URL information from certificate payloads sent by a remote security endpoint.

Figure 8-7 shows the Configuration Assistant panel where you may select CRL testing options (none, loose, or strict); this is in the IPSec Perspective in the Advanced Stack Settings panel.

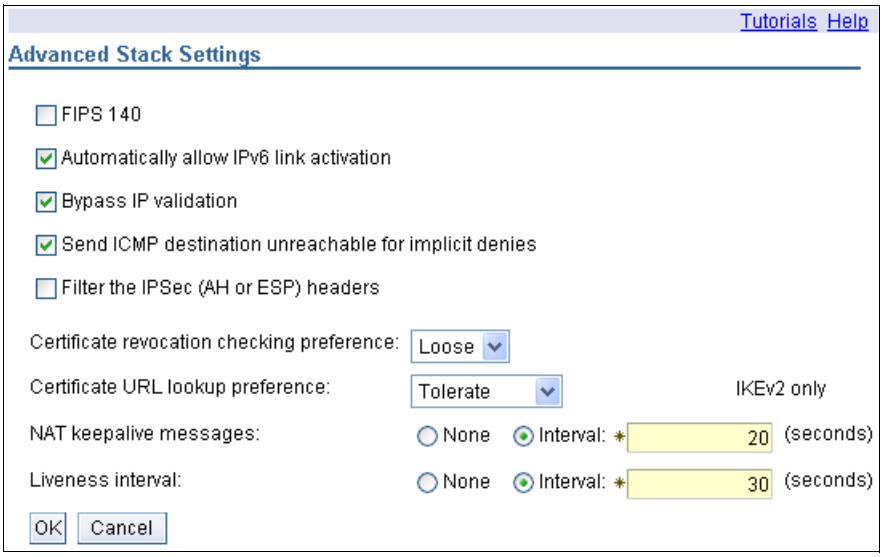


Figure 8-7 Configuration Assistant support for CRL verification

8.3.4 IPSec support for cryptographic currency

IPSec protection consists two phases in the establishment of two types of security associations. An SA is sometimes referred to as a "tunnel".

- Phase 1 SA (IKEv1) or IKE SA (IKEv2) creates a secure communications channel over an insecure network to negotiate the data protection SA in secret.
- Phase 2 SA (IKEv1) or Child SA (IKEv2) to actually protect the application data.

IPSec uses a number of cryptographic algorithms during these processes:

- Advanced Encryption Standard (AES) algorithm with a 256-bit key length in Cipher Block Chaining (CBC) mode.
- AES algorithm in Galois Counter Mode (GCM) in 128-bit and 256-bit key lengths that provides both data origin authentication and confidentiality, and is an efficient algorithm for high-speed packet networks.
- AES algorithm in Galois Message Authentication Code (GMAC) mode in 128-bit and 256-bit key lengths provides data origin authentication, but does not provide confidentiality. This is also an efficient algorithm for high-speed packet networks.
- Hashed Message Authentication Mode (HMAC) in conjunction with the SHA2-256, SHA2-384, and SHA2-512 algorithms. You can use these algorithms as the basis for data

origin authentication and integrity verification. The new algorithms (HMAC-SHA2-256-128, HMAC-SHA2-384-192, and HMAC-SHA2-512-256) ensure that the data is authentic and has not been modified in transit. Versions of these algorithms that are not truncated are available as pseudorandom functions (PRFs). These algorithms are called PRF-HMAC-SHA2-256, PRF-HMAC-SHA2-384, and PRF-HMAC-SHA2-512.

- AES128-XCBC-96, that ensures the data is authentic and not modified in transit.
- Support for elliptic curve digital signature algorithm (ECDSA) authentication.

Restriction: Support for ECDSA is limited to IKEv2 configurations that use NSS certificate services.

8.3.5 IPSec support for FIPS 140 cryptographic mode

Federal Information Processing Standards (FIPS) are standards adopted by the wider IT community and published by the federal government of the United States. These are often required to do business with various government agencies.

The FIPS 140 standard specifies the Security Requirements for Cryptographic Modules. These are the hardware or software that implements cryptographic algorithms or performs cryptographic key operations. To get FIPS 140 certified, the cryptographic modules must satisfy a number of requirements:

- How the information flows into and out of the cryptographic modules, and how that information is managed
- Authentication and role requirements specify who is allowed to perform what actions with the cryptographic modules
- Physical security requirements include locks and other tamper-resistant features, and the ability to withstand environmental conditions
- Cryptographic key management and the generation and storage of cryptographic keys

z/OS Communication Server supports Security Associations (SAs) distribution when the SAs are running in FIPS 140 mode. When FIPS mode is negotiated with AES-GCM combined-mode encryption and authentication algorithm, or with the AES-GMAC authentication algorithm and in a System-Wide Security Association (SWSA), IPSec supports both in IPSec tunnels. This combination is not supported in prior releases.

Notes:

- For environments where you want to use Sysplex-Wide Security Associations (SWSA) in FIPS 140 mode, with AES-GCM or AES-GMAC algorithms, and where you have z/OS 1.13 and z/OS 1.12 running simultaneously, PTF PM29788 should be applied on z/OS 1.12.
- For this release, ensure that APAR OA34403 is applied. Additionally, if the CSFSERV SAF class is active on your system, the IKE daemon must be permitted to the CSF1DVK and CSF1DMK resource profiles.
- Running in FIPS 140 mode can impact performance. Consider running in non-FIPS 140 mode for better performance, unless you are required to run in FIPS 140 mode.

Figure 8-8 shows z/OS Communications Server cryptographic landscape in the new FIPS 140 cryptographic mode for IPSec.

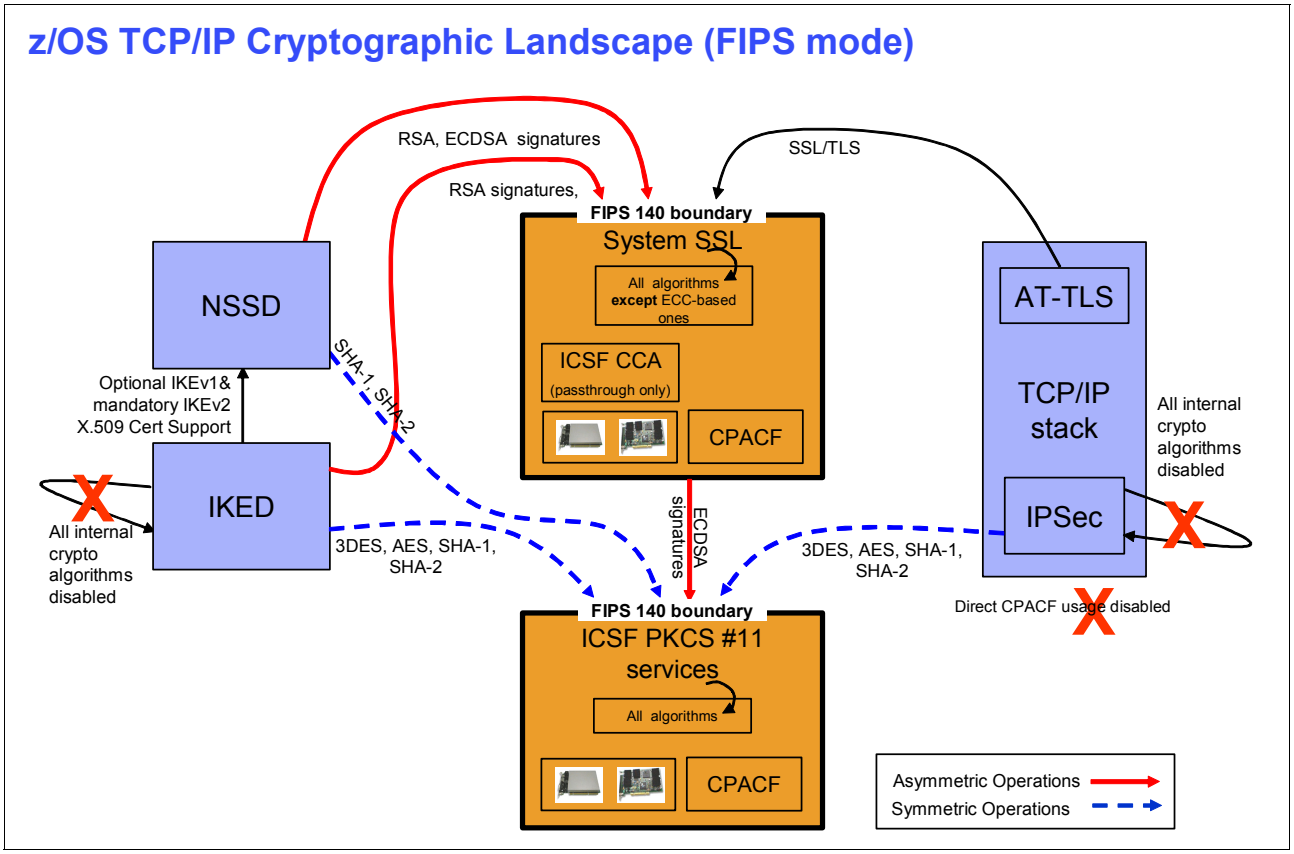


Figure 8-8 FIPS-140 cryptographic mode for IPSec

The figure shows the following information:

- The three major components of z/OS IPSec are NSSD, IKED, and the TCP/IP stacks. These can be independently configured for FIPS 140 cryptographic mode.
- A logical cryptographic boundary is introduced that separates the cryptographic operation requestors from the cryptographic operation providers.
- All cryptographic operations are performed inside the boundary, and be initiated by cryptographic modules in FIPS 140 modes of operation.

Reminder: Start ICSF in FIPS compatibility mode rather than in pure FIPS mode so that FIPS mode algorithm restrictions on the PKCS #11 interface into ICSF are only imposed on daemons and users wanting to comply with FIPS mode requirements.

FIPS 140 support must be implemented in stages, in “top down” order:

1. Configure FIPS 140 mode for NSSD. When operating in FIPS 140 mode, NSSD can serve both FIPS and non-FIPS IKE daemon clients, but in both cases NSSD will only create and verify for certificates that conform to FIPS 140 requirements.
2. Configure FIPS 140 mode for IKED. When operating in FIPS 140 mode, IKED can serve both FIPS and non-FIPS TCP/IP stacks, but in both cases the IKE daemon will omit restricted algorithms from any proposal it builds, and only use the PKCS #11 interface to ICSF.
3. Configure FIPS 140 mode for TCP/IP stacks.

Incorrect configurations: Incorrect configurations can arise if a “bottom up” approach is attempted. If a TCP/IP stack is enabled for FIPS 140 mode but IKED is not, IKED cannot provide any cryptographic services to that TCP/IP stack. If IKED is enabled for FIPS 140 mode but NSSD is not, the NSS daemon will not provide certificate services to that IKE daemon.

For details, see “Cryptographic Services ICSF Overview” at the Cryptographic Services bookshelf.

`http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/CSFBKZC0`

To set up the NSSD, IKED, and the TCP/IP stacks in FIPS 140 mode, use the z/OS Configuration Assistant.

Figure 8-9 shows how to select FIPS 140 mode for NSS Server from the Advanced NSS Server Settings panel of the Configuration Assistant. Note the check box for the FIPS 140 option. The Configuration Assistant provides similar options for IKED and the TCP/IP stacks in the “Advanced IKE Daemon Settings” and the “Advanced Stack Settings” panels.

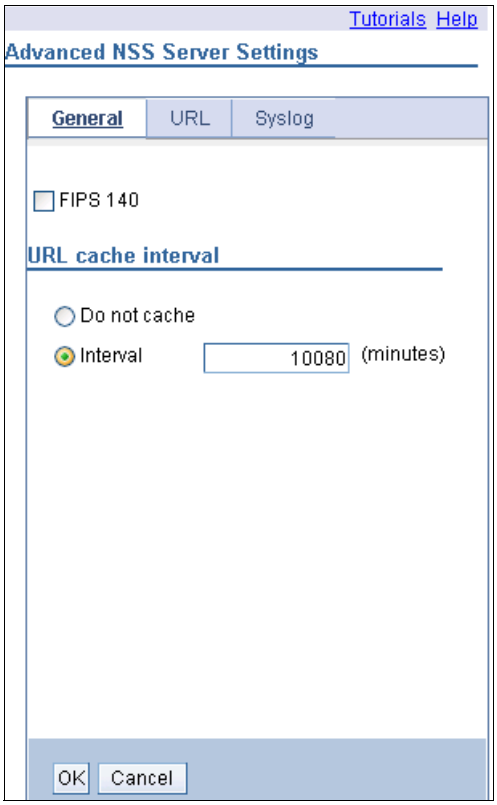


Figure 8-9 NSSD configuration of FIPS 140 mode for IPSec

Use the following commands to display the FIPS 140 mode information:

- The **`F NSSD,DISPLAY`** command for the NSS daemon
- The **`F IKED,DISPLAY`** command for the IKE daemon
- The **`ipsec -f display`** command for the TCP/IP stacks

For details about AES-256, see 8.3.6, “AES cryptographic support for integrated IPSec in a VPN” on page 260. For some detail on trusted TCP connections, see 8.3.7, “Trusted TCP connections” on page 260.

8.3.6 AES cryptographic support for integrated IPSec in a VPN

The z/OS Communications Server supports the Advanced Encryption Standard (AES) algorithm for IP security with a 256-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, you can create a Security Level object by using the steps that are described in “Creating a security level for PFS” on page 269. In the New Security Level - Cipher Selections window, select **AES** as the encryption algorithm (Figure 8-10). Use **HMAC SHA1** because MD5 is no longer considered secure enough.

The way AES (and other algorithms) is invoked changes depending on whether FIPS 140 is turned on. In this section, we use the non-FIPS approach

Important: The AES encryption software is subject to export restrictions and might not be available in your country.

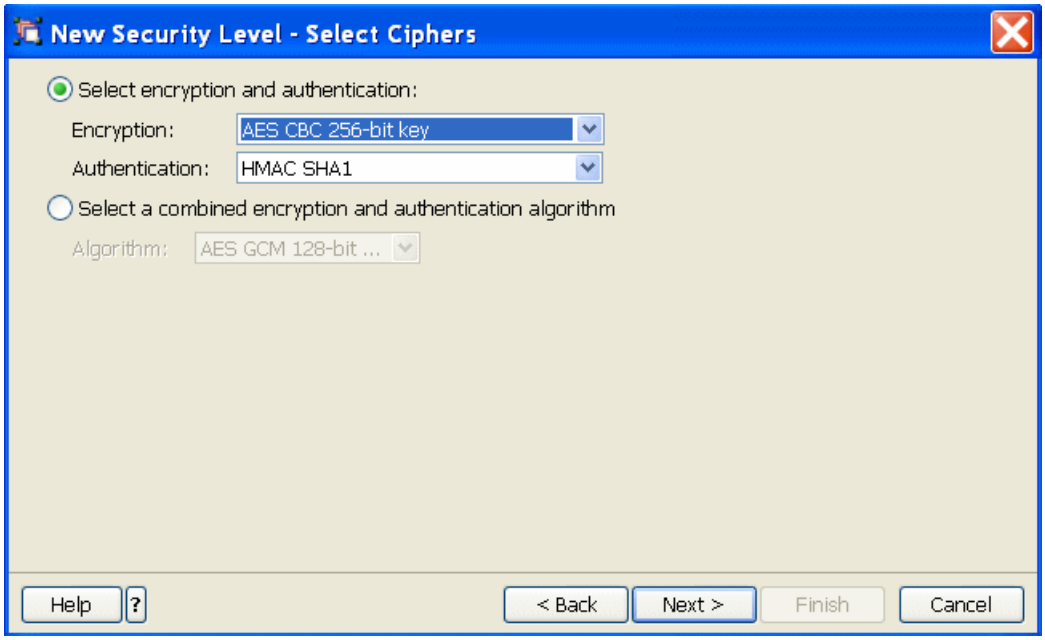


Figure 8-10 Implementing AES security level

Tip: TCP/IP and IKED require ICSF to perform 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.

8.3.7 Trusted TCP connections

End-point authentication support is available to exchange security credentials between partners in client/server communication. If the partners are on separate TCP/IP stacks, the exchange is across a cross-system coupling facility (XCF) connection.

Figure 8-11 shows how two partners, Bob and Alice, exchange credentials across XCF. This method eliminates the overhead in using the normal TCP security extension protocols, such as SSL/TLS, Kerberos, or SSH for the communication between Alice and Bob.

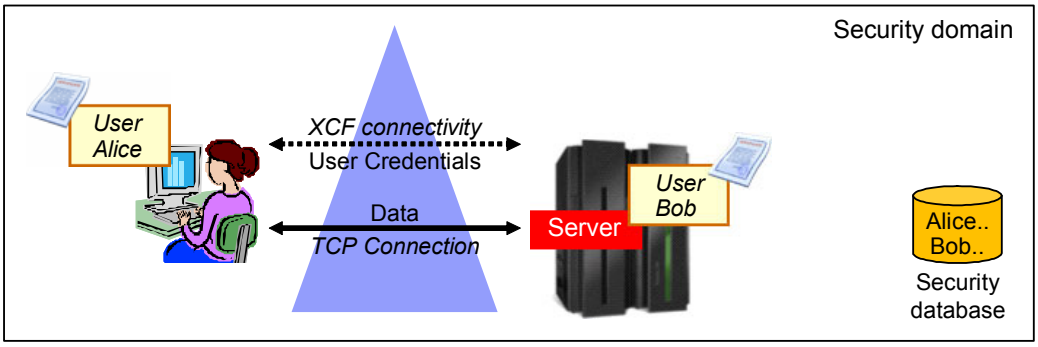


Figure 8-11 Exchange partner security credentials to create trusted TCP connections

To use this support, applications can code the **SIOCGPARTNERINFO ioctl** API to request the partner security credentials such as the user ID or the UTOKEN and sysplex-specific connection routing information. The application can then perform access control checks of the partner.

This exchange is not part of the TCP/IP connection setup and only the applications that request credentials need modification. The exploiting socket partners can use the partner security credentials to perform access control checks. For example, they can check whether the partner has sufficient privileges and has a “trust relationship” with the partner to create a trusted TCP connection.

This **ioctl** command is supported on both IPv4 and IPv6 TCP sockets.

This trusted relationship can be used for one-way or two-way communications. *One-way communication* is when one endpoint extracts partner security credentials. *Two-way communication* is when both end points extract partner security credentials.

For information about trusted TCP/IP connections and the **SIOCGPARTNERINFO** and **SIOCSPARTNERINFO ioctl** calls, see *z/OS Communications Server: IP Programmer’s Guide and Reference*, SC31-8787.

8.3.8 zIIP Assisted IPSec function

The IBM System z Integrated Information Processor (zIIP) is a specialty engine available on System z servers. The zIIP’s execution environment accepts eligible work from z/OS, which manages and directs the work between the general purpose processor and the zIIP.

The zIIP Assisted IPSec function allows z/OS Communications 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 IPsec protocol processing (AH and ESP), including their use of encryption and authentication algorithms, 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 might be able to achieve a significant reduction in general purpose CPU consumption.

For information about configuring zIIP Assisted IPSec, see Appendix D, “zIIP Assisted IPSec” on page 903.

8.4 Working with the z/OS Communications Server Network Management Interface

The z/OS Communications Server includes a Network Management Interface (NMI) API that network management products can take advantage of to provide simple problem determination for complex environments. IBM 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. Therefore, IKED must be running to make use of this NMI service. Figure 8-12 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.

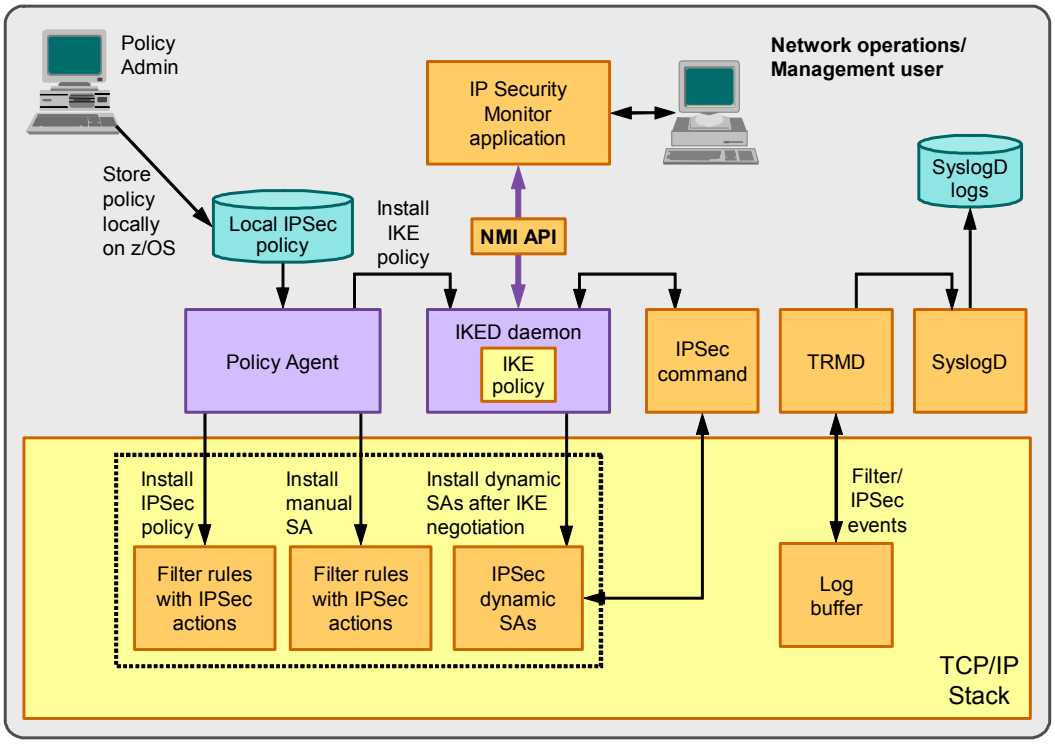


Figure 8-12 IPSec network management interface support

Data that is 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 might choose to include:

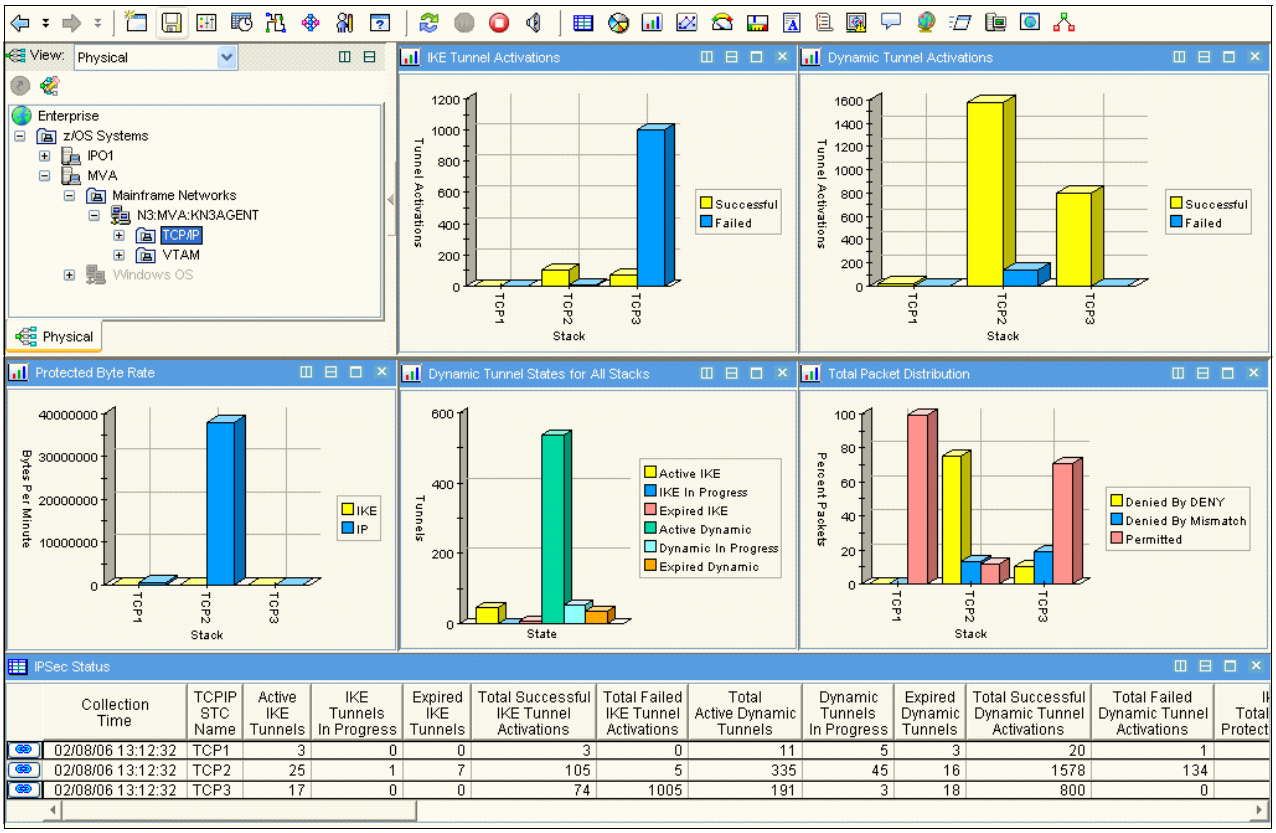
- Stack information (global IPSec configuration settings)
- Summary statistics (various counters and statistics for TCP/IP and IKED)
- Current IP filters (either default or policy filters)
- Default IP filters (defined in the TCP/IP profile)
- Policy IP filters (defined using the policy agent)
- Port 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 might 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 8-13 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	I 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 8-13 IPSec Status Workspace, the dashboard

Authorization to use the NMI

SAF authorization in the SERVAUTH class to 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.

8.5 How IPSec is implemented

IPSec uses the services of a number of z/OS Communications Server components (illustrated in Figure 8-12 on page 262) to provide policy-based security.

Figure 8-12 on page 262 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 5, “Central Policy Server” on page 161.

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. We discuss this in detail in Chapter 9, “Network Security Services for IPSec clients” on page 325.

Figure 8-14 shows the various z/OS processes and the interaction between them.

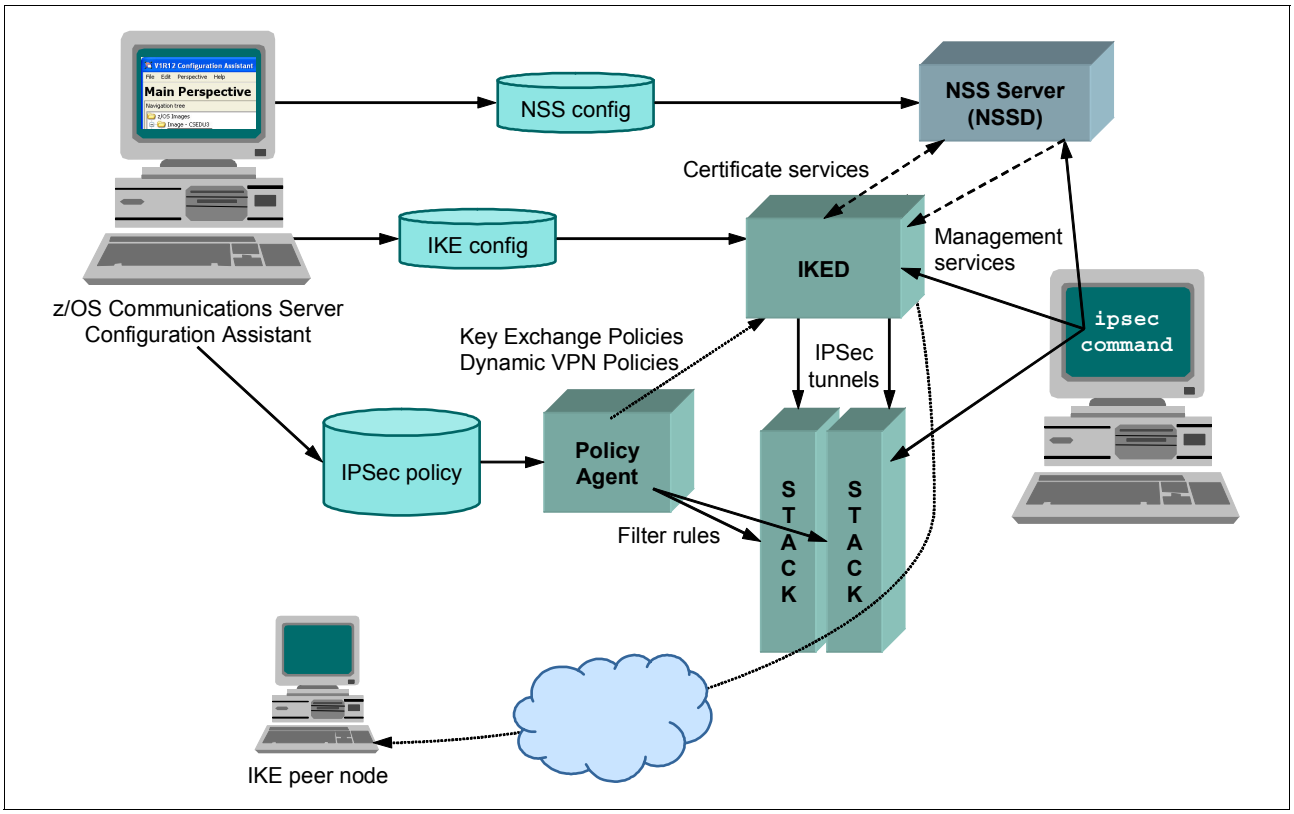


Figure 8-14 z/OS processes in IPSec

In the following sections, we describe the steps that are required to implement IPSec:

- Installing the PAGENT
- Setting up the Traffic Regulation Management daemon (TRMD)
- Updating the TCP/IP stack to activate IPSec
- Restricting the use of the **ipsec** command
- Installing the IBM Configuration Assistant for z/OS Communications Server
- Description of the IPSec scenarios
- Defining the IPSec policies to PAGENT
- Setting up the IKE daemon
- Setting up the system logging daemon (SYSLOGD) to log IKED messages
- Starting the IKE daemon and verifying initialization
- Commands used to administer IP security

8.5.1 Installing the PAGENT

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 4, “Policy agent” on page 103.

After setting up PAGENT, define the IpSecConfig statement to specify the path of the policy file that contains stack-specific IPSec policy statements to PAGENT. The PAGENT started task contains the following path statement:

```
//STDENV   DD PATH='/etc/pagent.sc&SYSCLONE..env',PATHOPTS=(ORDONLY)
```

In the statement, `sc&SYSCLONE` specifies the z/OS system SC30, SC31, SC32, or SC33. On SC32, this points to `/etc/pagent.sc32.env` file.

Example 8-2 shows the contents of the `/etc/pagent.sc32.env` file that contains the pointer to the PAGENT configuration file `/SC32/etc/pagent.sc32.conf` that is common to all stacks.

Example 8-2 Contents of /etc/pagent.sc32.env

```
LIBPATH=/usr/lib:.
PAGENT_CONFIG_FILE=/SC30/etc/pagent.sc32.conf
PAGENT_LOG_FILE=SYSLOGD
TZ=EST5EDT
```

One of the statements in `pagent.sc32.conf` points to the TCP/IP stack specific configuration file used for stack TCPIPA as shown in Example 8-3.

Example 8-3 Reference to /etc/pagent.sc32.tcpipc.conf file with IP security configured

```
TcpImage TCPIPC /etc/pagent.sc32.tcpipc.conf  FLUSH PURGE 600
```

8.5.2 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 4.2.10, “Centralized Policy Server” on page 124 and 4.3, “Setting up the Traffic Regulation Management daemon”.

8.5.3 Updating the TCP/IP stack to activate IPSec

To activate IPSec, you need to add the IPSECURITY option 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 8-4 on page 267.

Important: Make certain that you build 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 might 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.

Using the EZACMD System REXX command to issue ipsec from the z/OS console is a good method to revert to the IPSEC statements in the PROFILE. For more information about using EZACMD see the following sources:

- 8.7, “MVS console support for selected TCP/IP commands” in *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7996
- 2.6.5, “EZACMD console command security” on page 29 of this book
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781

Example 8-4 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  SECCLASS 100
;  ;;  Administrative access
 IPSECRULE *  10.1.1.10 LOG  PROTOCOL *
ENDIPSEC
```

8.5.4 Restricting the use of the ipsec command

The **ipsec** command is powerful and needs to be protected from unauthorized use. We created a RACF profile for this and gave command access to user CS03. Example 8-5 shows the commands to do this. For more information, see 2.6.4, “IPSec command access control” on page 29

Example 8-5 Define access control for the ipsec command

```
SETROPTS GENERIC(SERVAUTH) RDEFINE SERVAUTH EZB.IPSECCMD.* UACC(NONE)
PERMIT EZB.IPSECCMD.* CLASS(SERVAUTH) ID(CS03) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

8.5.5 Installing the IBM Configuration Assistant for z/OS Communications Server

The policy infrastructure required to implement IPSec depends on various configuration files and policy definitions files. The IBM Configuration Assistant for z/OS Communications Server enables flat-file configuration of all supported policy types for z/OS. The IBM Configuration Assistant for z/OS Communications Server is an optional GUI-based tool that provides a guided interface for configuring TCP/IP policy-based networking functions. You can use the Configuration Assistant to generate the Policy Agent files.

IBM provides a graphical user interface (GUI) named the IBM Configuration Assistant for z/OS Communications Server in two forms:

- As a task in IBM z/OS Management Facility (z/OSMF)

  z/OSMF provides a web browser interface for a variety of z/OS system management functions. When you invoke the Configuration Assistant in z/OSMF, the Configuration Assistant runs natively in the z/OS system and you can access it through a web browser. To use the Configuration Assistant in z/OSMF, your system must be z/OS V1R11 or later.

  This interface is fully supported and we are using it.

- As a stand-alone application that you can run on your workstation

  This GUI is a Windows-based interface that you can download from the IBM website.

> **Note:** z/OS V1.13 is planned to be the final release for which the IBM Configuration Assistant for z/OS Communications Server tool that runs on Microsoft Windows will be provided by IBM. This tool is currently available "as-is," and is a non-warranteed web download. Customers who currently use Windows-based IBM Configuration Assistant for z/OS Communications Server tool should migrate to the z/OS Management Facility (z/OSMF) Configuration Assistant application. The IBM Configuration Assistant for z/OS Communications Server that runs within z/OSMF is part of a supported IBM product and contains all functions supported with the Windows tool.

We use the Configuration Assistant in z/OSMF to help code our security policies. We refer to it as *z/OSMF Configuration Assistant*.

When your policy is coded using this software, you can either upload it to the z/OS system through the FTP client that is built into the Configuration Assistant, or also save it to a z/OS UNIX System Services file when using z/OSMF Configuration Assistant, so that it can be used by the PAGENT.

For more details about using z/OSMF Configuration Assistant see 4.4, "Configuration Assistant for z/OS Communications Server" on page 127.

8.5.6 Description of the IPSec scenarios

This section includes testing scenarios in which we used z/OSMF Configuration Assistant.

Dynamic tunnels with pre-shared key mode

In our testing, we used z/OSMF Configuration Assistant to implement scenarios using dynamic IPSec tunnels that rely on pre-shared key mode.

We configured IPSec between two z/OS systems, described in 8.6, "Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2" on page 285. We executed the following steps:

- Setting up the IKE daemon
- Setting up the IPSec policy
- Installing the configuration files
- Verifying IPSEC between two z/OS images

We configured IPSec between z/OS and a Windows platform. In "IPSec between z/OS and Windows: Pre-shared Key Mode" on page 860, we provide a detailed description of the following steps for this configuration process:

- Setting up the IKE daemon
- Setting up the z/OS IPSec policy
- Setting up the Windows IPSec policy
- Verifying that things are working

Dynamic tunnels with RSA mode

In our testing, we used z/OSMF Configuration Assistant to implement several scenarios using dynamic IPSec tunnels that rely on RSA mode for authentication.

We configured IPSec between two z/OS systems, described in 8.7, “Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1” on page 306. In that section we executed the following steps:

- Creating the x.509 certificates for RSA mode
- Setting up the IKE daemon
- Setting up the IPSec policy
- Installing the configuration files
- Verifying IPSEC between two z/OS images

We configured IPSec between z/OS and a Windows workstation. In “IPSec between z/OS and Windows: RSA mode” on page 882, we provide a detailed description of the steps for this configuration process, including the following steps:

- Setting up the IKE daemon
- Creating the x.509 certificates for RSA mode
- Setting up the z/OS IPSec policy
- Setting up the Windows IPSec policy
- Verifying that things are working

Manual tunnels

We do not include an example of manual tunnels in this document because they are not considered a secure form of IPSec. Consult the IBM Configuration Assistant Help and the z/OS Communications Server product manuals if you want more information about configuring manual tunnels.

8.5.7 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 the security requirements of your installation. You can define your IPSec security policies to PAGENT in one of two ways:

- Use the z/OSMF Configuration Assistant to create the IP security configuration file. We are using the supported version of Configuration Assistant running under z/OSMF.
- Manually code all of the required policy statements to create a configuration file in a z/OS UNIX file or an MVS data set. We do not recommend this method because it is laborious.

To implement these security requirements, the z/OSMF Configuration Assistant provides, by default, three predefined security level objects, which are used for the Phase2 negotiation:

- IPSEC_Gold
- IPSEC_Silver
- IPSEC_Bronze

Creating a security level for PFS

You can also create your own security levels to meet business requirements using z/OSMF Configuration Assistant. IPSec on z/OS Communications Server supports Perfect Forward Secrecy (PFS) to generate keys on phase 2 (IPSec tunnel) negotiation, and 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 for phase 1 must be reestablished from scratch, followed by a phase 2 key rebuild as well. Phase 2 refreshes, if no PFS is configured, use earlier keys as a starting point when refreshing the phase 2 key. This implies that if a hacker compromises an earlier key, the presently-used key can be determined more easily.
- 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. Thus, the processor (or crypto card) is required to do a little more work during a phase 2 refresh than if no PFS is in use. However, security is enhanced.

Using PFS is optional in a phase 2 negotiation. To work with different clients, z/OS Communications Server accepts multiple PFS values. Some clients might only be able to support lower PFS groups. Other clients might support higher PFS groups for higher security.

In our case, before we defined our scenarios, we used z/OSMF Configuration Assistant to create a specific security level called *PFS*, which implements PFS values that allow multiple security levels. If you want PFS but do not want to create the required security level, you can use VPN~A.[1] This security level includes the PFS with Diffie-Hellman Group 2. The security level we created supports additional PFS algorithms.

To create a security level for PFS, start the IBM Configuration Assistant, and then complete the following steps:

1. In the Main Perspective panel, select the IPSec technology click **Select Action** then click **Configure**.
2. In the IPSec Perspective main panel, click **Security level**.

[1] See RFC4308 at http://www.rfc-editor.org/rfc/rfc4308.txt.

3. In the Security Levels panel, click **Select Action**, and then click **Add**, as shown in Figure 8-15.

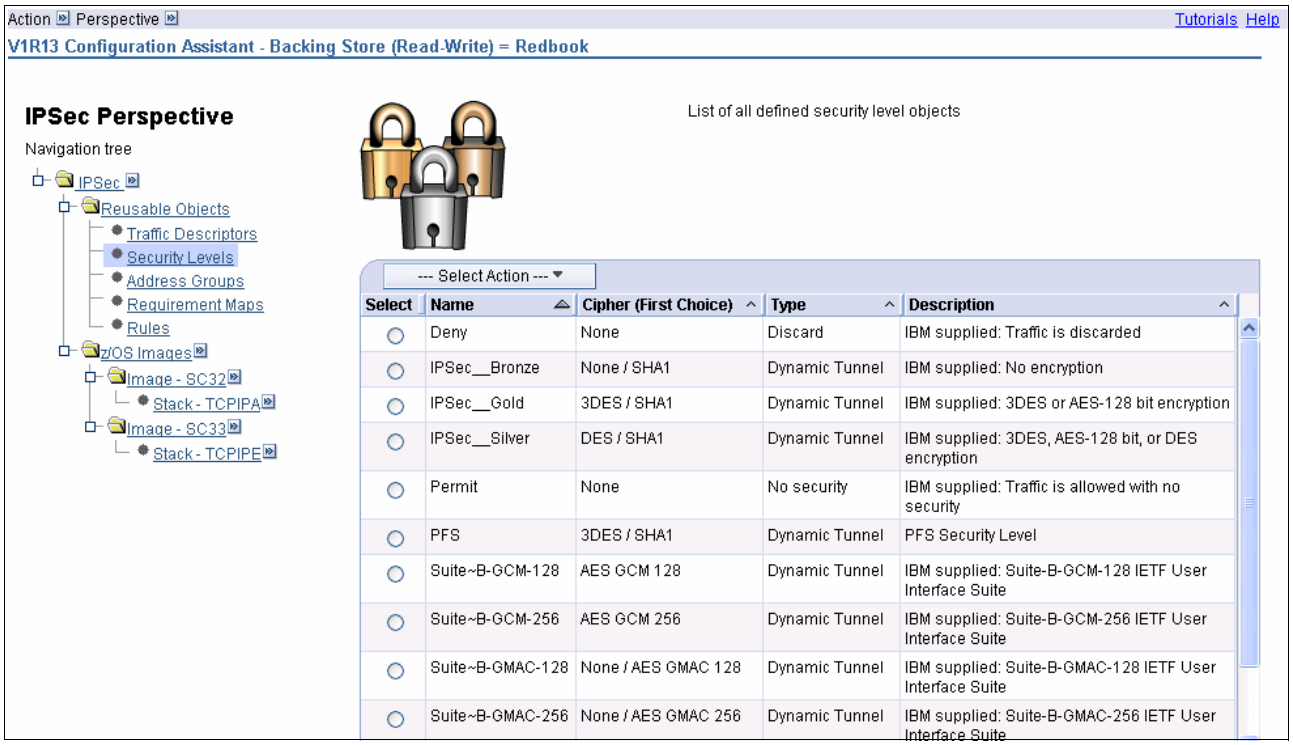


Figure 8-15 Add a new security level object

4. In the New Security Level: Name and Type panel, enter the name of the new security level object (in our case, PFS) and a description as shown in Figure 8-16. Select **IPSec Dynamic Tunnel**, and click **Next**.

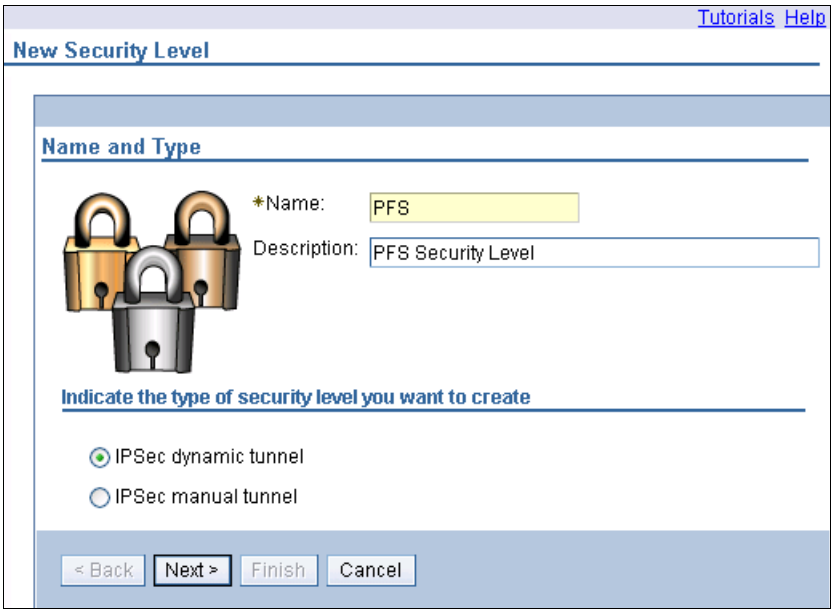


Figure 8-16 New Security Level: Name and Type panel

5. In the New Security Level: Select Ciphers panel, select **Triple DES** as the encryption algorithm and **HMAC SHA1** as the authentication algorithm, as shown in Figure 8-17. Click **Next**. You can also select AES 128-bit, because AES is rapidly becoming the de facto top standard for security. Choose SHA over MD5 because SHA is considered more secure.

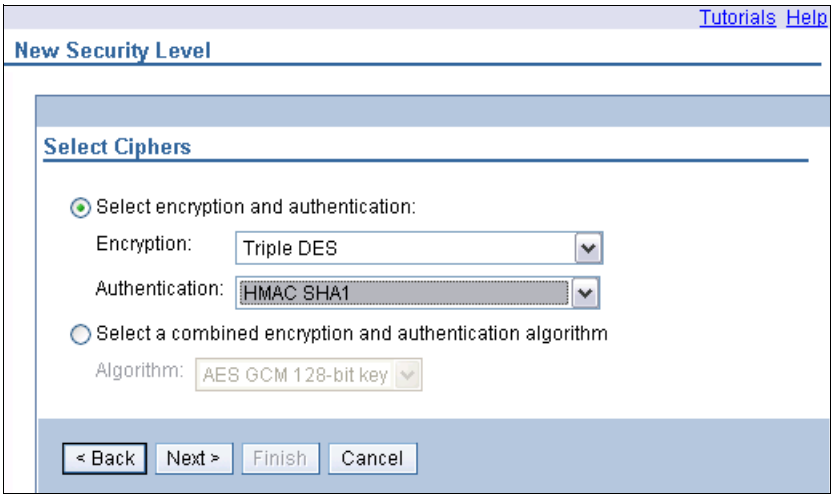


Figure 8-17 New Security Level: Select Ciphers

6. In the New Security Level: Additional Settings panel, click **Advanced Settings**.
7. In the Advanced Dynamic Tunnel Settings panel, click the PFS Diffie-Hellman tab. Select **Group 2** from the “Initiate using Diffie Hellman” drop-down menu and select **None**, **Diffie-Hellman Group 1, Diffie-Hellman Group 2**, and **Diffie-Hellman Group 5** in the “Acceptable Perfect Forward Secrecy Levels” section**,** as shown in Figure 8-18 on page 273. We do not really need to select Diffie-Hellman Group 14. Diffie-Hellman Group 14 is more appropriate with an even stronger symmetric encryption such as AES. Click **OK**, and then click **Finish**.

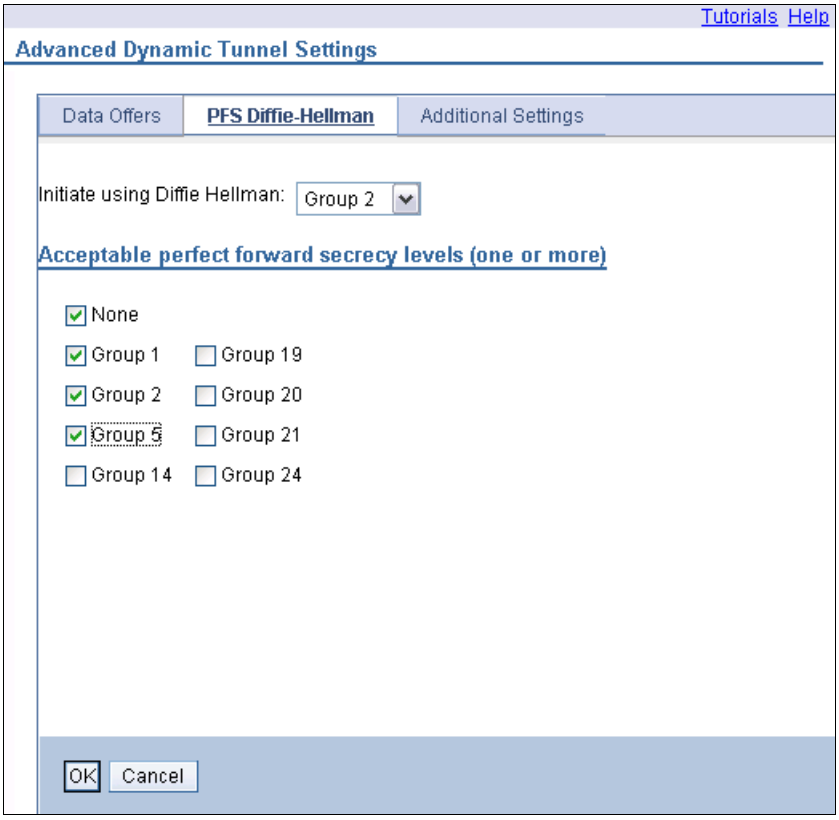


Figure 8-18 Advanced Dynamic Tunnel Settings panel

In the IPSec Perspective panel, the list of all defined security level objects should now include the PFS object as shown in Figure 8-19.

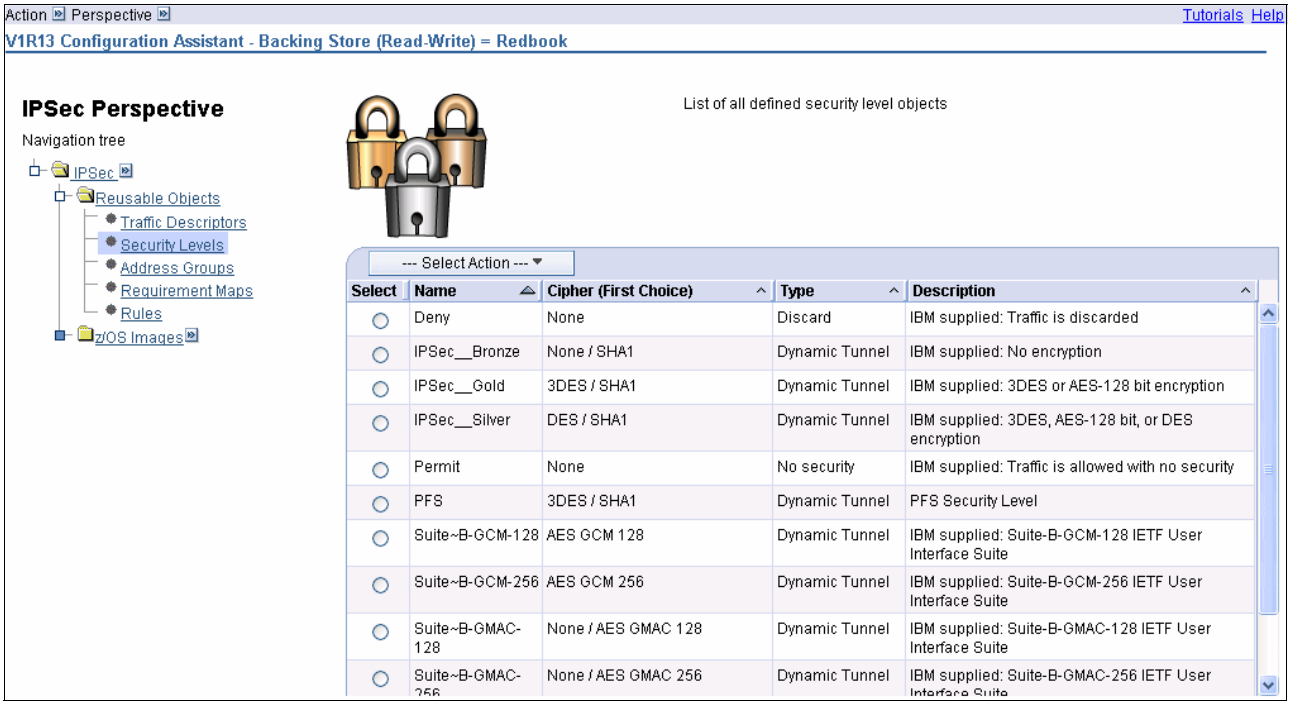


Figure 8-19 Security Levels panel with PFS security level added

8.5.8 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 can 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.

To set up the IKE daemon, you need to complete the following steps:

1. Set up the IKE daemon cataloged procedure.
2. Create the IKE daemon configuration file.
3. Reserve the IP ports for the IKE daemon.
4. Associate a RACF user ID and group with the IKE daemon.
5. Define profiles to control access to the RACDCERT command.
6. Create a RACF key ring (for the IKEv1 RSA signature mode scenarios only).
7. Install an X.509 digital certificate for the IKE daemon (for the IKEv1 RSA signature mode scenarios only).
8. Optional: Authorize use of hardware cryptographic encryption.

We explain these steps in the following sections.

Reminder: If you are not using IKEv1 with RSA signature mode, steps 6 and 7 are optional.

Set up the IKE daemon cataloged procedure

You can obtain a sample of the procedure from the z/OS Communications Server installation file TCPIP.SEZAINST(IKED). Example 8-6 shows the procedure that we used. Copy this procedure into your SYS1.PROCLIB library.

Example 8-6 IKE daemon cataloged procedure

```
//IKED     PROC
//*
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD PATH='/etc/iked.sc&SYSCLONE..env',PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

Example 8-7 shows the contents of the STDENV file, /etc/iked.sc32.env. The /etc/iked.sc32.conf file is the IKE daemon configuration file shown in Example 8-8. IKED_CTRACE_MEMBER is the name of a parmlib member that contains default CTRACE settings for IKE daemon.

Example 8-7 /etc/iked.sc32.env contents

```
IKED_FILE=/etc/iked.sc32.conf
IKED_CTRACE_MEMBER=CTIIKE00
```

Create the IKE daemon configuration file

A sample configuration is supplied in the z/OS Communications Server installation file /usr/lpp/tcpip/samples/IBM/EZAIKCFG. To configure our environment, we copied this file to /etc/iked.sc32.conf and changed it to suit our needs as shown in Example 8-8.

Example 8-8 The /etc/iked32.conf file used as our IKE daemon configuration file

```
IkeConfig
{
  IkeSyslogLevel    127
  PagentSyslogLevel 64
  Keyring           IKED/IKED32_keyring
  KeyRetries        10
  KeyWait           30
  DataRetries       10
  DataWait          15
  Echo              no
  PagentWait        0
  SMF119            IKEALL
}
```

The following parameters control the workings of the IKE daemon:

IkeSyslogLevel — Level of logging from the IKE daemon. We set it to 127 to get useful debug messages during testing. On the production system, this can be set to 1.

PagentSyslogLevel — Level of logging from pagent. We set it to 64 for testing.

Keyring — Owning user ID and ring name for RSA mode of authentication for IKEv1. We demonstrate how to set this up in “Create a RACF key ring” on page 279.

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.
SMF119	Specifies the level of logging to send to the SMF facility.

Tip: In our case, IkeSyslogLevel and PagentSyslogLevel were set to higher levels 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 also set the SMF119 to IKEALL to generate the SMF records related to all events.

Reserve 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. In our environment, we updated the TCPIPC stack profile on SC32 LPAR, located in TCPIPC.TCPPARMS(PROFC32) as shown in Example 8-9.

Example 8-9 TCPIPC.TCPPARMS(PROFC32) PORT statements

```
PORT
...
 500 UDP IKED
...
4500 UDP IKED
...
```

Be sure to specify the correct name of the IKE daemon. We used the name IKED.

Associate a RACF user ID and group with the IKE daemon

We defined a user ID, IKED, and connected it to a group, IKE, with an OMVS segment. A home directory was also assigned to this user ID.

We then defined the started task IKED to RACF and associated the user IKED 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 8-10 shows the commands that 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 8-10 Associate a RACF user ID and group with IKE daemon

```
ADDGROUP IKE OMVS(GID(931))
ADDUSER IKED DFLTGRP(IKE) OMVS(UID(130) HOME('/var/ike/')) NOPASSWORD
CONNECT IKED GROUP(IKE) UACC(READ)
RDEFINE  STARTED IKED.*  STDATA(USER(IKED))
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 these commands. You might need to work with the RACF administrator, who has the authority to issue them.

Additional requirements for IKED running as non-superuser

When running IKED with a nonzero UID it does not have superuser authority. Additional access requirements are required in this case. The following examples show how we addressed those requirements in our environment. For more information see the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Example 8-11 shows the changes we made to the UNIX file system attributes for the /var/ike directory.

Example 8-11 /var/ike file attribute changes for IKED as non-superuser

```
CS03 @ SC32:/SC32/var/ike>ls -pal
total 40
drwxr-xr-x   2 SYSPROG  SYS1        8192 Oct 15  2010 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
CS03 @ SC32:/SC32/var/ike>su
CS03 @ SC32:/SC32/var/ike>chown IKED /var/ike
CS03 @ SC32:/SC32/var/ike>ls -pal
total 40
drwxr-xr-x   2 IKED     SYS1        8192 Oct 15  2010 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
CS03 @ SC32:/SC32/var/ike>chgrp IKE /var/ike
CS03 @ SC32:/SC32/var/ike>ls -pal
total 40
drwxr-xr-x   2 IKED     IKE         8192 Oct 15  2010 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
CS03 @ SC32:/SC32/var/ike>chmod 770 /var/ike
CS03 @ SC32:/SC32/var/ike>ls -pal
total 40
drwxrwx---   2 IKED     IKE         8192 Oct 15  2010 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
```

Example 8-12 shows the changes we made to the UNIX file system attributes for the /var/sock directory. Changing the owner to root with the **chown** command is done because this directory is shared by a number of TCP/IP applications, and we would not view any one of them as the owner of the directory itself.

Example 8-12 The /var/sock file attribute changes for IKED as non-superuser

```
CS03 @ SC32:/SC32/var/sock>ls -pal
total 32
drwxrwxrwt   2 SYSPROG  SYS1        8192 Jul 15 13:31 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
crwxrwxrwx   1 NET      SYS1       6,  0 Jul 13 20:48 SNAMGMT
CS03 @ SC32:/SC32/var/sock>chown 0 /var/sock
CS03 @ SC32:/SC32/var/sock>ls -pal
total 32
drwxrwxrwt   2 SYSPROG  SYS1        8192 Jul 15 13:31 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
crwxrwxrwx   1 NET      SYS1       6,  0 Jul 13 20:48 SNAMGMT
CS03 @ SC32:/SC32/var/sock>chmod 777 /var/sock
```

```
CSO3 @ SC32:/SC32/var/sock>ls -pal
total 32
drwxrwxrwx   2 SYSPROG  SYS1        8192 Jul 15 13:31 ./
drwxrwxrwt   7 3456     SYS1        8192 Oct  4  2010 ../
crwxrwxrwx   1 NET      SYS1       6,  0 Jul 13 20:48 SNAMGMT
```

Example 8-13 shows the RACF changes we made to allow IKED to retrieve IP security policies from Policy Agent. When IKED is running without superuser authority, it needs READ access to `EZB.PAGENT.sysname.stackname.IPSEC`. Because a SERVAUTH profile named `EZB.PAGENT.SC33.TCPIPE.*` already existed, we had to grant IKED access to it.

Example 8-13 RACF authorization to allow IKED to retrieve IPSec polices from PAGENT

```
PERMIT EZB.PAGENT.SC33.TCPIPE.* CLASS(SERVAUTH) ID(IKED) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Example 8-14 shows the RACF changes we made to allow IKED to issue console messages without the BPXM023I prefix.

Example 8-14 RACF changes to avoid BPXM023I prefix

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

Authorize 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 use digital signature authorization.

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 8-15.

Example 8-15 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, see the following resources:

- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683, for use of the RACDCERT command
- *z/OS Security Server RACF Command Language Reference*, SA22-7687, for other RACF commands

For more information about System z hardware cryptography, see the following resources:

- *z/OS Cryptographic Services ICSF Overview*, SA22-7519
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521

8.5.9 RACF certificate definitions for IKED

When using digital signature (RSA) mode with IKED, RACF allows you to create and maintain security keys, key rings, and digital certificates in the RACF database. We use RACF in our environment and describe here the steps to set up digital key rings and certificates in RACF.

Define profiles to control access to the RACDCERT command

Tip: This step is necessary if IKE is implemented with RSA signature mode authentication.

You can use the RACDCERT command to generate keys and key rings and to connect the keys to key rings. This facility needs to be protected, and only authorized users, such as the person creating and managing certificates, IKE daemon, and NSS daemon should have access to it. In our environment, CS03 is the user ID used to create and manage certificates. Example 8-16 shows the commands that we used.

Example 8-16 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(CS03) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(CS03) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(CS03) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(CS03) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(IKED) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IKED) ACC(UPDATE)
```

Create a RACF key ring

Tip: This step is necessary if IKE is implemented with RSA mode authentication for IKEv1. If you want to exploit centralized certificate management services for an IKED client, consult Chapter 9, “Network Security Services for IPSec clients” on page 325. Note that NSS *must* be used if you want to use any digital -based authentication with IKEv2.

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 following TSO command:

```
RACDCERT ID(IKED) ADDRING(IKED32_keyring)
```

Tip: The value used for the key ring name is case-sensitive.

Install an X.509 digital certificate for the IKE daemon

Tip: This step is necessary if IKE is implemented with RSA mode authentication for IKEv1. If you want to exploit centralized certificate management services for an IKED client, consult Chapter 9, “Network Security Services for IPSec clients” on page 325. Note that NSS *must* be used if you want to use any digital -based authentication with IKEv2.

You can install an X.509 digital certificate using the following methods:

- 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.

We explain these steps 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 as shown in Example 8-17. The label for our certificate was `My Local Certificate Authority`. We use this label to refer to the certificate in the steps that follow.

Example 8-17 Generate a self-signed certificate

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(                    -
            O('I.B.M Corporation')                       -
            CN('ITSO z/OS CS')                           -
            C('US'))                                     -
            NOTBEFORE(DATE(2008-11-11))                  -
            NOTAFTER(DATE(2012-11-11))                   -
            WITHLABEL('My Local Certificate Authority')  -
            KEYUSAGE(certsign)
setropts raclist(facility) refresh
racdcert certauth list(label('My Local Certificate Authority'))
```

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, as shown in Example 8-18.

Example 8-18 Create a certificate for the server

```
RACDCERT ID(IKED) GENCERT                                        -   1
         SUBJECTSDN  (CN('IKE Daemon on SC32')                   -   2
                      OU('ITSO')                                 -
                      C('US'))                                   -
                      NOTBEFORE(DATE(2010-11-11))                -   3
                      NOTAFTER(DATE(2012-11-11))                 -   3
                      WITHLABEL('IKE Daemon on SC32')            -   4
                      ALTNAME (IP(10.1.1.30)                     -   5
                      DOMAIN('sc32a.itso.raleigh.ibm.com')       -   5
                      EMAIL('sc32a@sc32a.itso.raleigh.ibm.com'))  -   5
                      SIGNWITH(CERTAUTH                          -
                      Label('My Local Certificate Authority'))       6
  setropts raclist(DIGTCERT) refresh                                 7
  racdcert ID(IKED) list(label('IKE Daemon on SC32'))                8
```

Example 8-18 highlights several components of this definition and the commands that follow:

1. The ID parameter identifies this certificate that is being generated (GENCERT) as a personal user certificate. It is not a CA or SITE certificate.

2. SUBJECTSDN identifies several components that comprise the x.509 Distinguished Name (DN) of the certificate owner or holder. Be sure each DN is unique at least within a RACF database, and also be unique worldwide, because this DN is used to distinguish identities in many cases. We have used only three components of the DN for SC32: CN, OU, and C. See Chapter 3, “Certificate management in z/OS” on page 39, for detailed descriptions of these and other components.

3. 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.

4. The RACF database requires a label for organizing the certificates within a RACF database. The label must be unique.

5. Coding an ALTNAME makes the IKE peer identity definitions easier. We code three alternate names:

- An IP address
- A domain name
- A fully qualified domain name

6. This definition tells the GENCERT process which CA should be the signing authority for the user’s personal certificate.

7. The **`setropts`** command refreshes the DIGTCERT class so that the changes are made immediately known in the running RACF environment and operating system.

8. The **`racdcert`** command allows us to verify that our certificate was created properly. It displays the certificate with its attributes.

Connecting the certificates to IKED’s key ring

The certificates that we created in the previous two steps need to be connected to IKED’s key ring as shown in Example 8-19.

Example 8-19 Connect the certificates to IKED's existing key ring

```
RACDCERT ID(IKED) CONNECT(ID(IKED)                        -
                  LABEL('IKE Daemon on SC32')             -
                  RING(IKED32_keyring)                      -
                  USAGE(personal))
RACDCERT ID(IKED) CONNECT(ID(IKED) CERTAUTH               -
                  LABEL('My Local Certificate Authority') -
                  RING(IKED32_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/iked.sc32.conf that we defined earlier:

```
Keyring IKED/IKED32_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 8-20 shows the commands to do the verification.

Example 8-20 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(IKED32_keyring)
```

Example 8-21 shows the output of these commands.

Example 8-21 Verify certificate creation

```
RACDCERT ID(iked) LIST(LABEL('IKE Daemon on SC32'))
Digital certificate information for user IKED:

  Label: IKE Daemon on SC32
  Certificate ID: 2QTJOsXEydLFQMSBhZSWlUCWlUDiw/Py
  Status: TRUST
  Start Date: 2010/11/11 00:00:00
  End Date:   2012/11/11 23:59:59
  Serial Number:
       >0F<
  Issuer's Name:
       >OU=ITSO z/OS CS.O=I.B.M Corporation.C=US<
  Subject's Name:
       >CN=IKE Daemon on SC32.OU=ITSO.C=US<
  Subject's AltNames:
    IP: 10.1.1.30
    EMail: sc32a at sc32a.itso.raleigh.ibm.com
    Domain: sc32a.itso.raleigh.ibm.com
  Key Type: RSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
    Ring Owner: IKED
    Ring:
```

```
       >IKED32_keyring<

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: 2008/11/11 01:00:00
  End Date:   2012/11/12 00:59:59
  Serial Number:
       >00<
  Issuer's Name:
       >OU=ITSO z/OS CS.O=I.B.M Corporation.C=US<
  Subject's Name:
       >OU=ITSO z/OS CS.O=I.B.M Corporation.C=US<
  Key Usage: CERTSIGN
  Key Type: RSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
    Ring Owner: IKED
    Ring:
       >IKED32_keyring<

RACDCERT id(IKED) LISTRING(IKED32_keyring)
Digital ring information for user IKED:
  Ring:
       >IKED32_keyring<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  IKE Daemon on SC32                  ID(IKED)       PERSONAL     NO
  My Local Certificate Authority      CERTAUTH       CERTAUTH     NO
```

8.5.10 Setting up the system logging daemon (SYSLOGD) 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/syslog.conf to route all IKED daemon logs:

```
*.IKED*.*.*     /var/syslog/%Y/%m/%d/iked.log -F 640 -D 770
```

8.5.11 Starting the IKE daemon and verifying initialization

Start IKED and make sure it starts correctly. Example 8-22 shows the startup messages.

Example 8-22 Starting IKED

```
S IKED
$HASP100 IKED     ON STCINRDR
IEF695I START IKED     WITH JOBNAME IKED     IS ASSIGNED TO USER IKED
   , GROUP IKE
$HASP373 IKED     STARTED
```

```
IEE252I MEMBER CTIIKE00 FOUND IN SYS1.PARMLIB
EZD0967I IKE RELEASE CS V1R13 SERVICE LEVEL CS110518 CREATED ON May
18 2011
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/iked.sc32.conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPC   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPC
EZD1046I IKE INITIALIZATION COMPLETE
```

8.5.12 Commands used to administer IP security

You can use the following commands to administer IP security:

ipsec Displays information about active filters and security associations and controls aspects of security association negotiation. Authority to use this command is controlled through the z/OS security server (RACF).

nssctl Displays information about currently connected NSS clients if you have implemented Network Security Services (NSS) on behalf of IKED clients, including both IKED and XML clients. This command also allows you to set the debug level for a connection while filtering on an NSS client name or discipline name.

pasearch Displays PAGENT 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 Makes the IKE daemon reread the IKED configuration file or makes the policy agent reread the policy configuration agent files.

Netstat Displays IPSECURITY status for a particular stack or displays the SecurityClass (SECCLASS) for a specific interface.

For more detailed information about the syntax and usage of those commands, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

8.6 Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2

In this scenario, we show how to set up a VPN tunnel between two z/OS systems as shown in Figure 8-20.

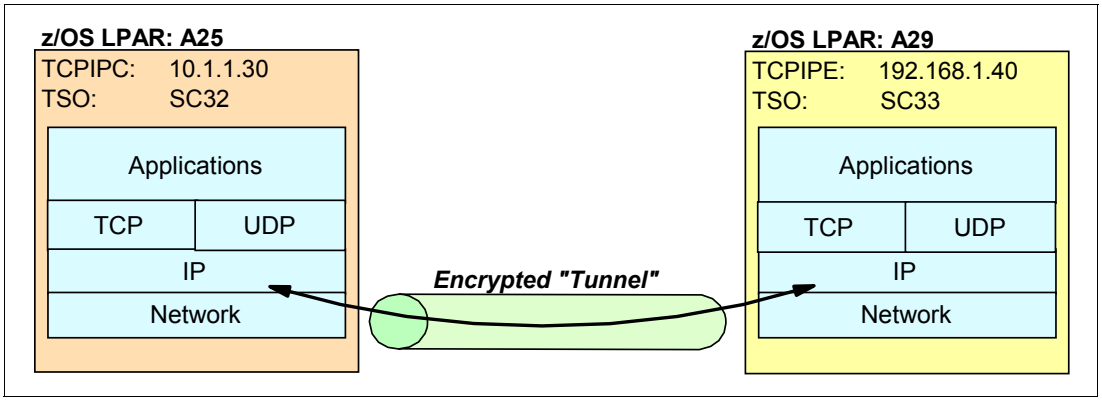


Figure 8-20 VPN traffic between two z/OS systems

We used the z/OSMF Configuration Assistant 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 two connectivity rules:

- One rule for basic services such as PING, Resolver, DNS, and omproute
- One rule for all the other traffic

This section demonstrates the process of defining the policy to set up this tunnel. This configuration creates a tunnel with the following characteristics:

- Permit basic services in our environment, such as PING, Resolver, DNS, OMPROUTE, and Service Connections Traffic to flow without encryption.
- All other IP packets between stacks TCPIPC on SC32 and our peer, 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 pre-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.

8.6.1 Using z/OSMF Configuration Assistant to set up the IPSec policies

To set up the IPSec policies using z/OSMF Configuration Assistant, complete the following steps:

1. Complete the general implementation steps for the IPSec scenarios.
2. Create the desired rules.

   For our scenario, we created the requirement map object named *BasicServices*, which has common policies that can be used for all z/OS images.
3. Add connectivity rules for the TCP/IP stack.
4. Add another new z/OS image.

Complete the general implementation steps for the IPSec scenarios

Connect to the z/OSMF Configuration Assistant.

Tip: z/OSMF Configuration Assistant Help is available through the **Help** button.

If this is a new backstore file, follow the steps to create a z/OS image to represent the z/OS system. Then, under this z/OS image, add a TCP/IP stack and enable the IPSec technology, as described in 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

If your z/OS image already has an active policy agent running, you can also import the active configuration using the policy agent configuration file import services. Then, you can implement the IP Filter policies in the latest version of policy agent configuration in use. For information about how to import the active configuration files see 4.6.4, “Importing the policy file to Configuration Assistant” on page 156.

After configuring or importing the z/OS Image and TCP/IP stack files, you can implement the IPSec policies.

Create the desired rules

Connectivity rules consist of a set of local and remote endpoints associated with a requirement map. A *requirement map* is a set of mappings of traffic descriptors to security levels. With a requirement map, you can implement the level of security that you want to provide for any given type of traffic. For example, you can create a map named *TN3270_Tunnel* that uses IPSec only for TN3270E traffic and denies all other protocols.

To use the IBM Configuration Assistant to create connectivity rules, follow these steps:

1. In the IPSec Perspective, select the TCP/IP stack in the Navigation tree. If necessary, click **Enable IPSec**, and when prompted as shown in Figure 8-21, click **Yes**.

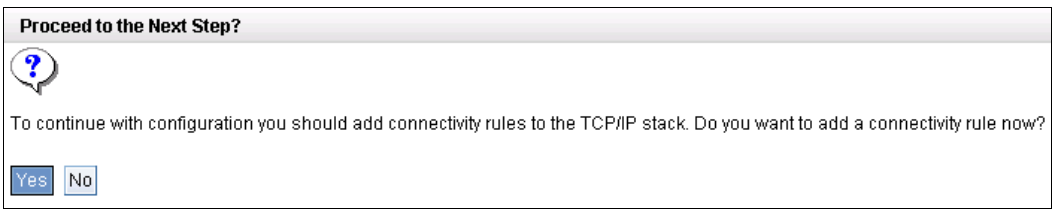


Figure 8-21 Configuration assistant prompting to create a connectivity rule

2. In the New Connectivity Rule - Welcome panel, ensure that **Typical** is selected (the default) and click **Next**.

3. At this point, you need to create rules for basic services, and these basic services do not require any IPSec protection. Select the **Filtering only** option in the Network Topology panel as shown in Figure 8-22. Because we did not need routed traffic in our test environment, we left that option cleared. Click **Next** to continue.

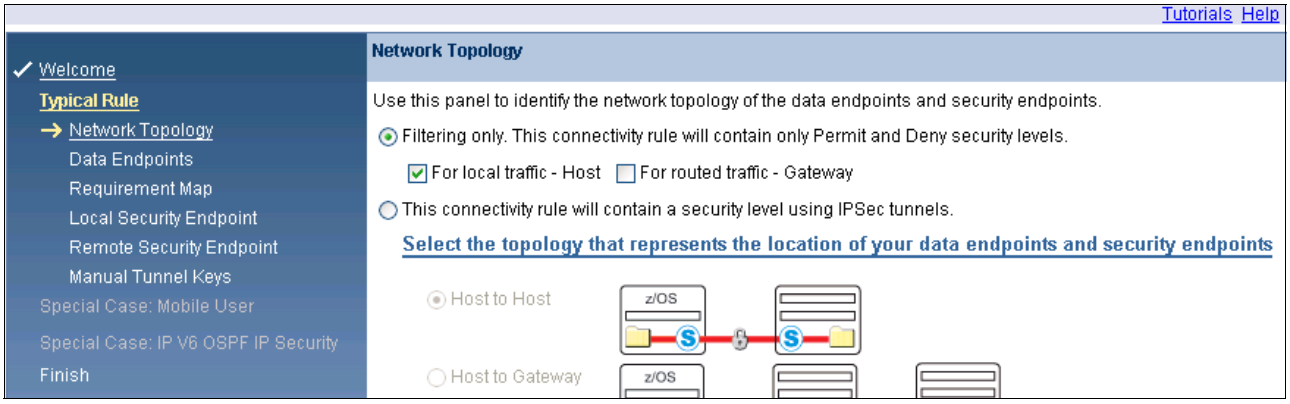


Figure 8-22 Choosing filtering only for the basic services rule

4. Update the connectivity rule name field to `BasicServices`, and make it applicable to all IP addresses as shown in Figure 8-23. Depending on your needs, you might want to limit the IP addresses that can use basic services. You might even want to divide this rule into more than one rule to provide more granularity in selecting the IP addresses that are permitted. Click **Next** to continue.

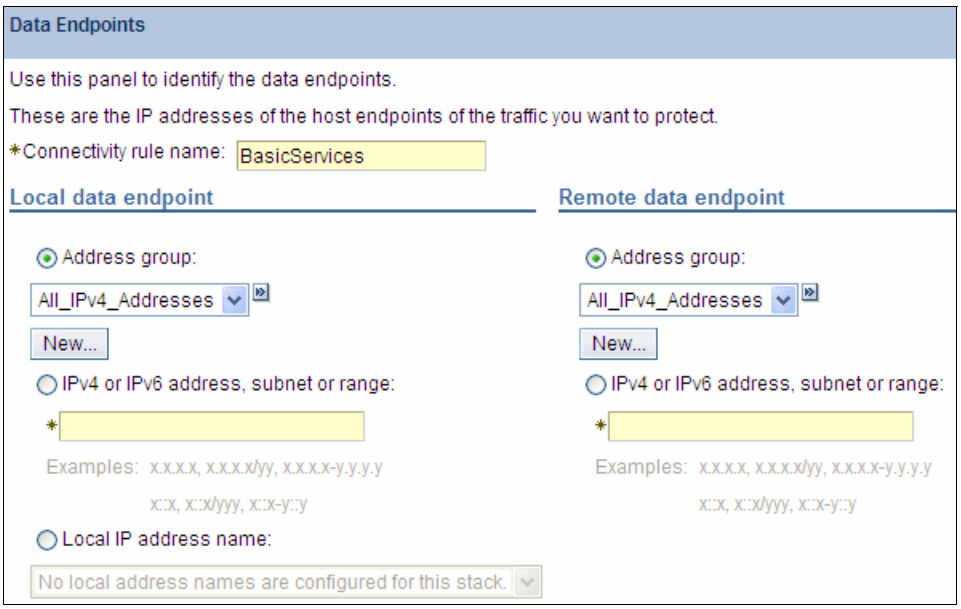


Figure 8-23 Coding all IP addresses for the endpoints for basic services

5. You can now create the requirement map that is associated with this rule. On the Requirement Map panel, shown in Figure 8-24 on page 288, define the following fields:
   a. Provide a name and description in the appropriate fields.
   b. Remove the All_other_traffic Traffic Descriptor by selecting **All_other_traffic** and clicking **Remove**. Alternatively, you can overwrite this descriptor with your values.
   c. Select the appropriate traffic descriptors from the Objects (such as PING, resolver, DNS, and so forth), and click **Add**.

d. On the Security Level tab for each Traffic descriptor in the list, select **Permit**.

e. Click **Next**.

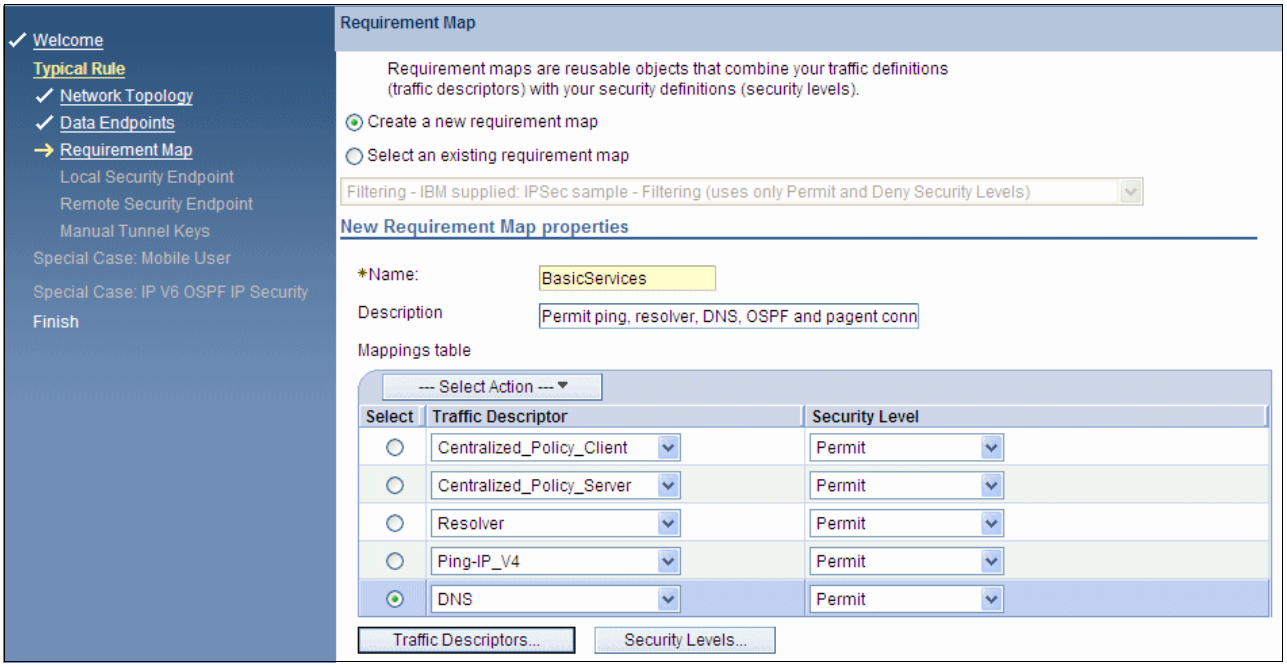


Figure 8-24 Add the BasicServices requirement map

6. In the New Connectivity Rule - Finish panel, click **Finish** to complete the setup for basic services unless you need to change the advanced parameters or logging settings.

Add connectivity rules for the TCP/IP stack

Next, build a rule for the traffic that you want to secure:

1. From the IPSec Perspective, with the stack selected, click **Select Action** → **Add** to begin the rule-building process. Again, accept **Typical** as the default on the Welcome panel. Click **Next** to continue.
2. In the New Connectivity Rule: Network Topology panel, in the "This Connectivity Rule will contain a Security Level using IPSec tunnels" section, select the **Host to Host** topology option as shown in Figure 8-25. Click **Next**.

> **Tip:** An IPSec VPN does not need to be an end-to-end (also called *host-to-host*) entity. In many instances, a VPN might not begin at your local workstation. A VPN endpoint can begin at the border between a secure network and a non-secure network, which is considered a gateway VPN. The same concept applies to the remote end. That is, the VPN might end at the gateway to the destination secure network, or the VPN might make it all the way to the destination host or workstation.

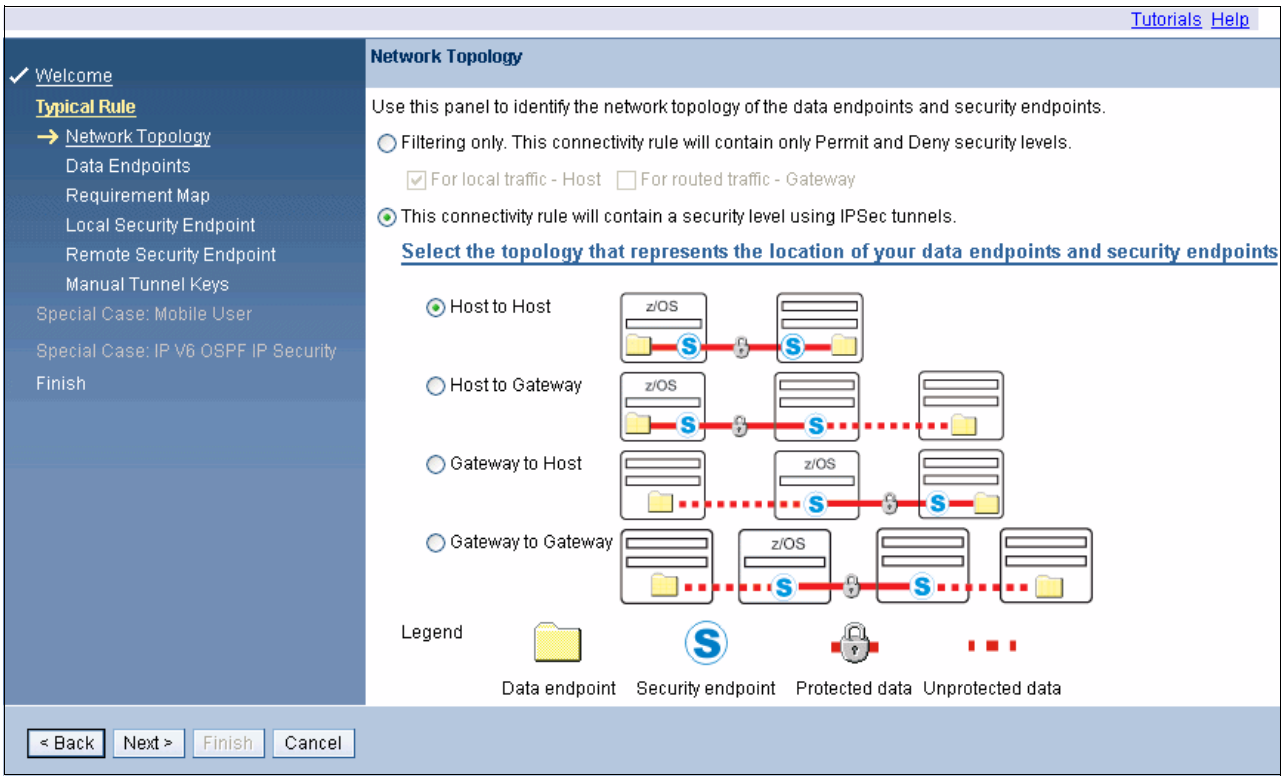


Figure 8-25 Selecting a Host to Host topology for our IPSec tunnel

3. In the New Connectivity Rule: Data Endpoints panel, shown in Figure 8-26, specify the endpoints of the connection by entering the IP addresses of the z/OS images, and provide a name for the rule. We use the name *All_Silver_Traffic*, based on the security level that we intend to use for this rule. Click **Next**.

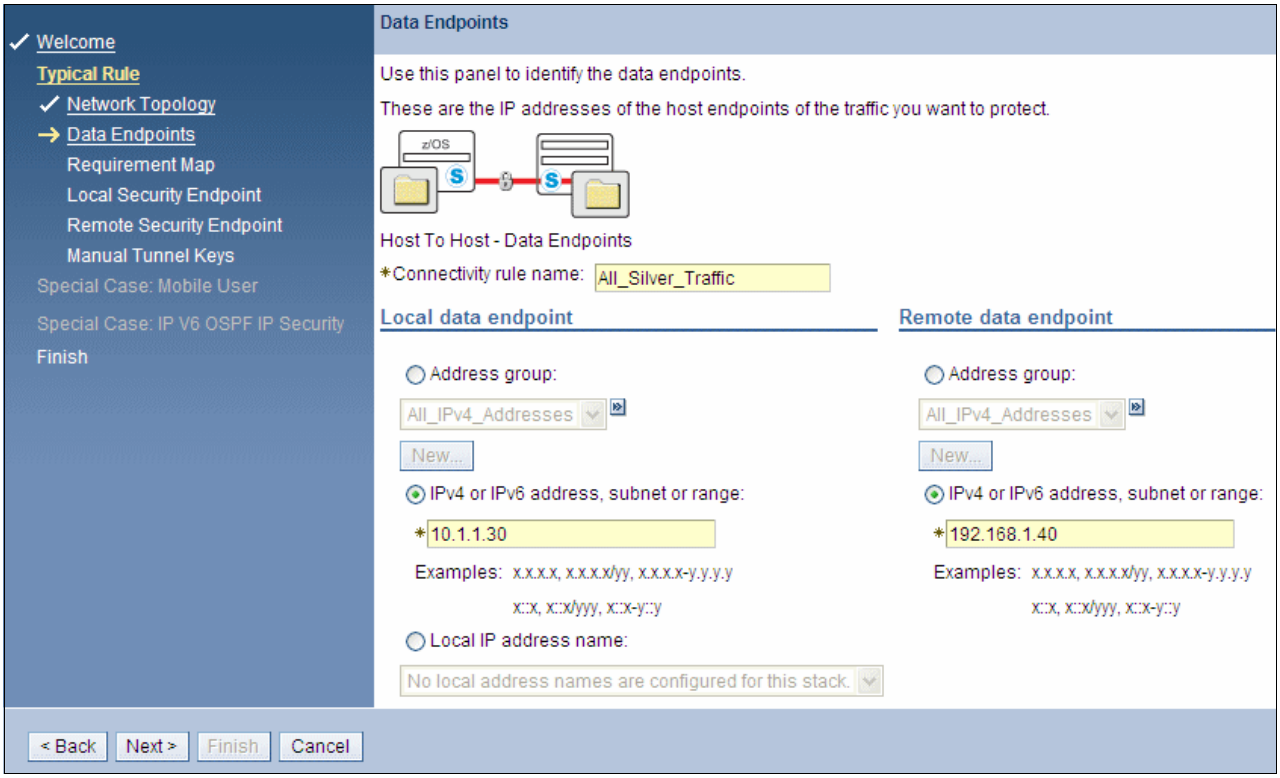


Figure 8-26 Entering the data endpoints for our IPSec tunnel

4. In the Requirement Map panel, select the traffic descriptor for `All_other_traffic` and a security level of IPSec_Silver, as shown in Figure 8-27 on page 291. To see details of the IPSec_Silver rule, click **Security Levels**, select IPSec_Silver then **Select Action** → **View Details**. Notice in the details that IPSec_Silver offers DES encryptions as its first choice. For a test environment, this option is acceptable, but DES encryption is not considered adequate to ensure confidentiality in a production environment. Remove all other rows from the mapping table, and click **Next**.

> **Tip:** IBM provides the following built-in security level objects:
>
> - Permit
> - Deny
> - IPSec_Bronze
> - IPSec_Silver
> - IPSec_Gold
>
> The permit and deny objects are for non-secured connections. The other IPSec objects are used for secured connections. Each object has a different definition, which you can view by selecting **IPSec** → **Work with Reusable Objects** → **Security Levels**. You can also define new security objects from this panel.

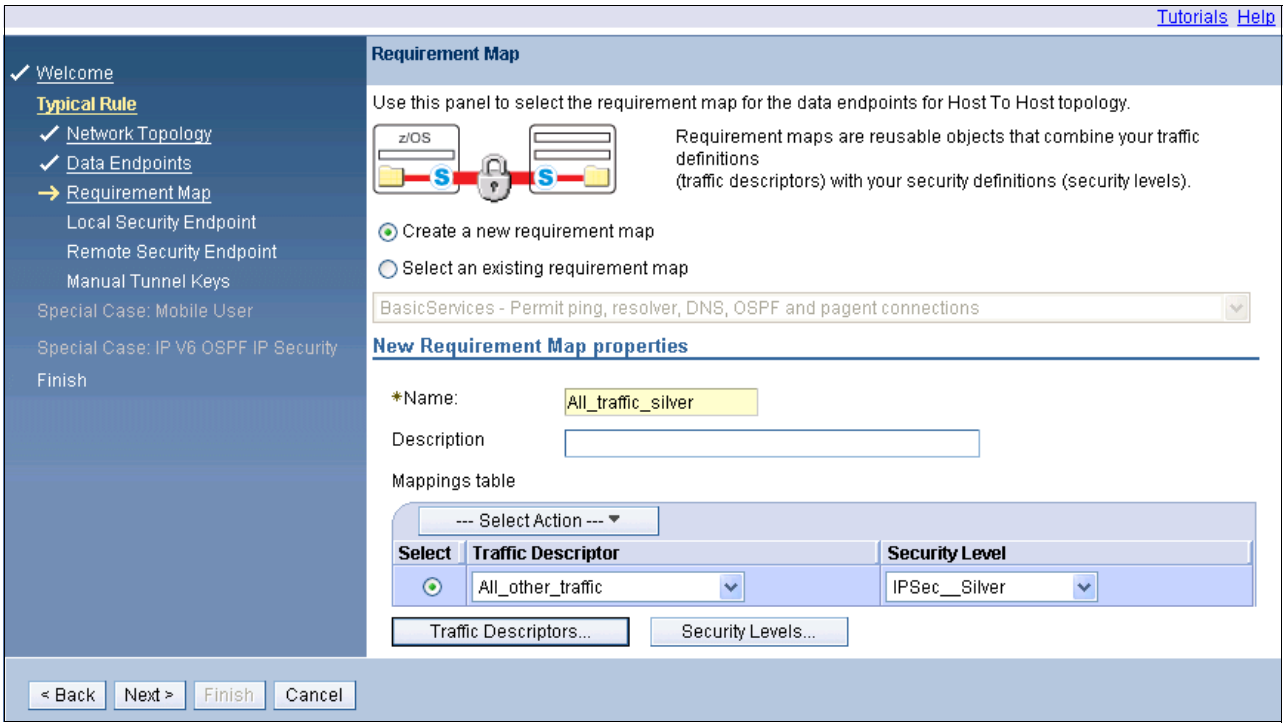


Figure 8-27 Connecting our rule to traffic and security level

5. Select the local IKE identity. In our scenario, we used the IP address 10.1.1.30. This identify must be the IP address that the IKE daemon uses when it starts tunnel negotiations. Click **Next**.
6. In the New Connectivity Rule: Remote Security Endpoint panel, shown in Figure 8-28 on page 292, select the remote IKE identity for the remote endpoint. In our scenario, we selected **IP address** and entered the IP address of the remote z/OS image of this rule (in our case, `192.168.1.40`). Recall the two ways to authenticate the remote IKE peers:
   - Using a digital signature
   - Using a shared key

   You may specify a shared key as an ASCII string, an Extended Binary Coded Decimal Interchange Code (EBCDIC) string, or a hexadecimal string.

Tip: Digital authentication takes advantage of the public key cryptography and X.509 certificate capabilities. It is secure but requires the overhead (relatively minor, in most cases) of establishing certificate authorities, personal certificates, and certificate repositories (key rings).

Shared key authentication can be quite robust, and is useful for testing because it is easy. However, ultimately, the safe distribution and storage of shared keys can present a long-term issue, such as where the shared secret is stored so that it can be recalled later.

For our test environment, we chose **Shared key** and entered an EBCDIC string, as shown in Figure 8-28. Click **Next**, and **Finish** in the next panel.

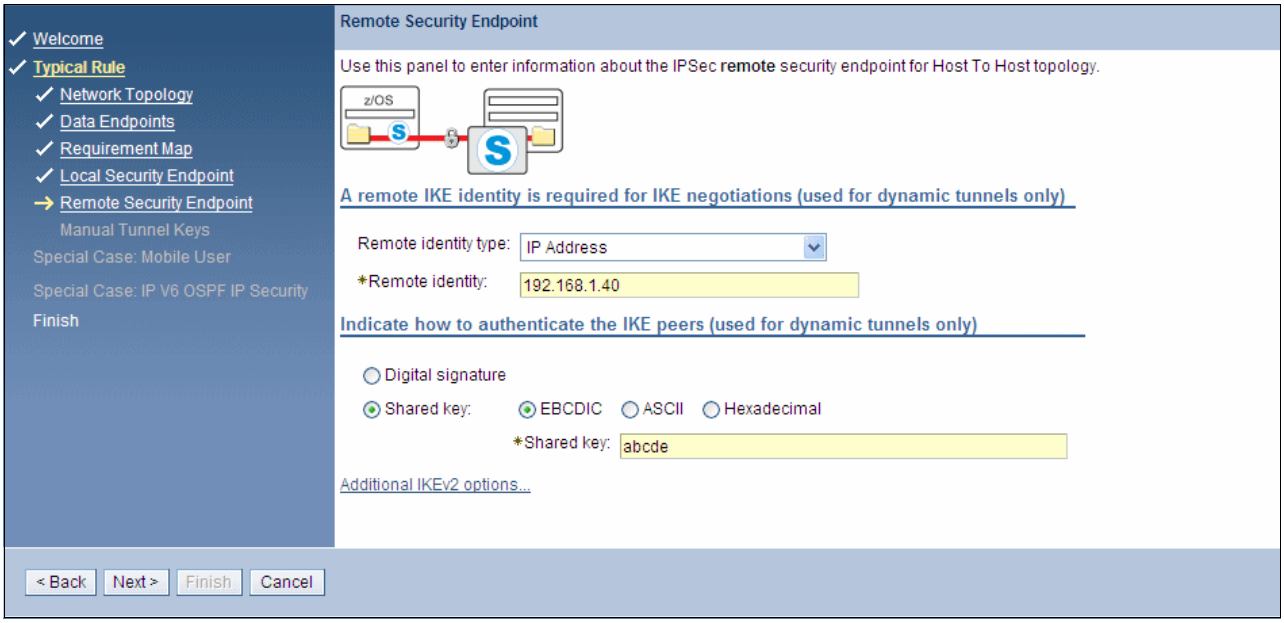


Figure 8-28 Configuring the remote security endpoint

7. After clicking **Finish**, you are returned to the IPSec Perspective. Click the Stack Settings tab, and select the following settings, as shown in Figure 8-29 on page 293:
   a. For the default initiator mode, choose IKEv2.
   a. Select whether the default allows Network Address Translation (NAT) traversal. In our scenario, we selected the “Do not Allow” option.
   b. Select whether to enable filter logging and whether to log implicitly denied events. We selected the “Enable Logging” and “Log Implicitly Deny Events” options. This last option can create a lot of log messages. Only select it in a testing environment.

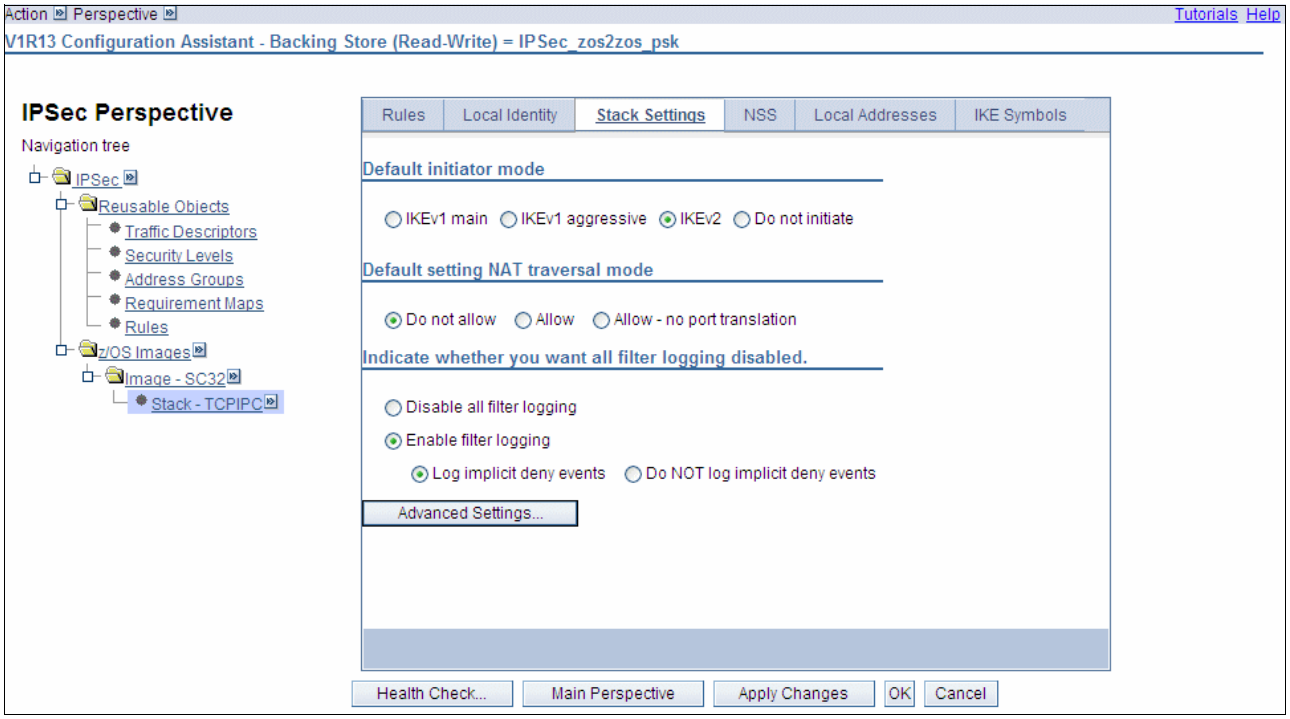


Figure 8-29 Stack Settings for TCPIPC

c. You can use the Advanced Settings button to select additional definitions as shown in Figure 8-30. In our scenario, we used all the default options. Click **OK** after setting the advanced options.

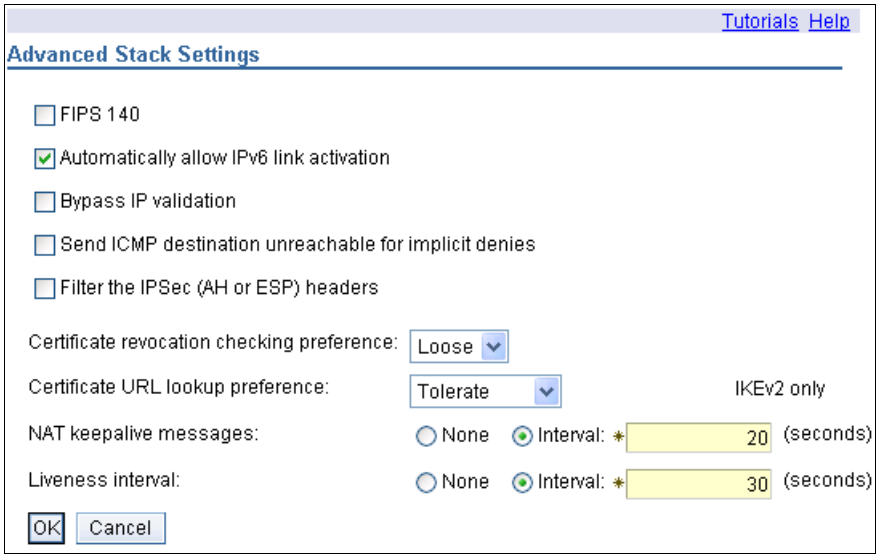


Figure 8-30 Advanced Stack options

d. Click **OK** again to continue.

8. Back in the IPSec Perspective panel, select the image name, **SC32**, in the Navigation tree. Apply the changes and, when prompted as shown in Figure 8-31, use a key ring named `IKED32_keyring` that is owned by user ID IKED. Click **OK** to continue.

Note: In this scenario, which uses a pre-shared key, the RACF key ring is not required or used. However the Configuration Assistant prompts for the name anyway.

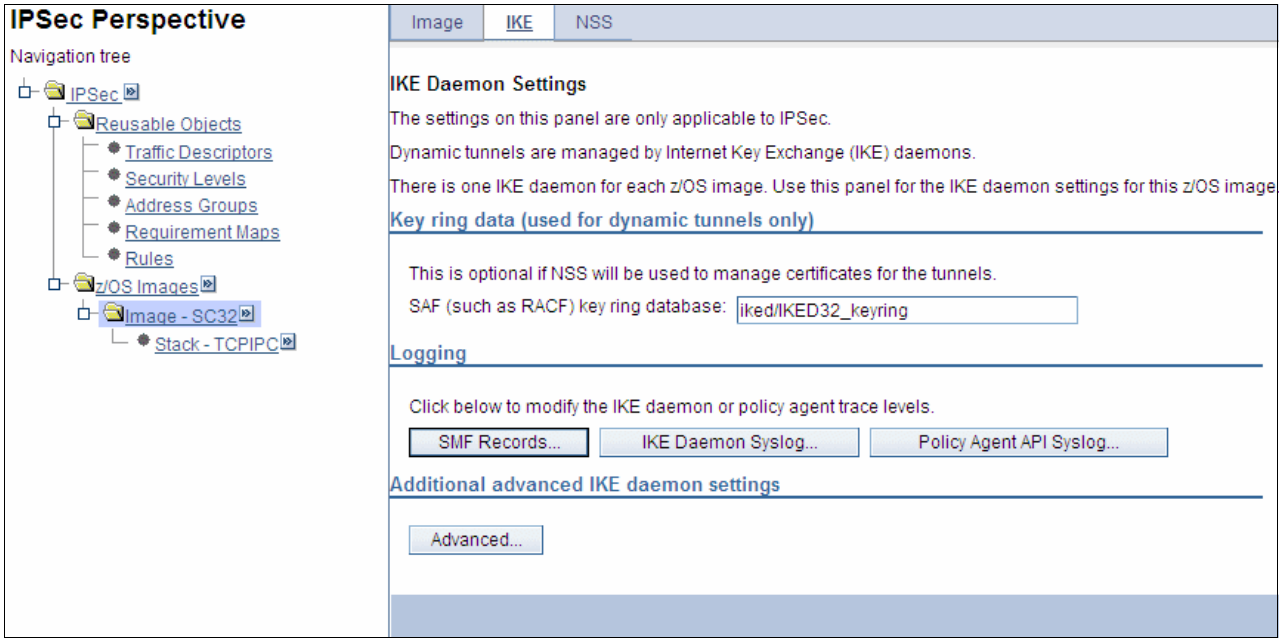


Figure 8-31 Enter the key ring when prompted

9. To see that the rule creation proceeded as desired, select the TCPIP stack in the Navigation tree. The information shown in Figure 8-32 is displayed.

> **Tip:** The order of the connectivity rules in a stack is important. When the policy is active and there is an attempt to send packets from or to the z/OS image, the first rule in the list is checked first. If a match exists with the definition of this first rule, it is used. If no match exists, the second rule is checked, and so on. There are performance implications too. If possible, place your most used rules at the top of the list.
>
> Inherent in the rules is the default rule that always exists for the policy agent, which is to deny everything. If no matching rule is found, the packet is denied by this default rule.

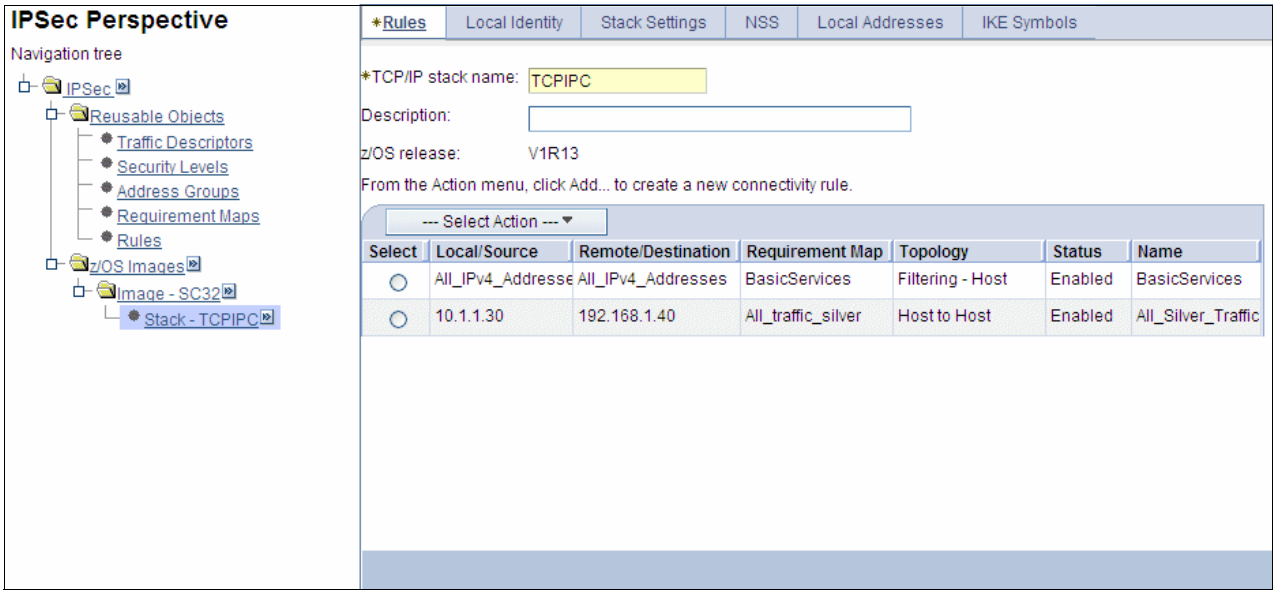


Figure 8-32 Confirming your connectivity rules

Now, you can create the other (remote) endpoint of the tunnel as described next.

Add another new z/OS image

To create the second endpoint with TCP/IP stack and connectivity rules, follow the steps described in “Create the desired rules” on page 286. In our testing, 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 8-33. Note that the remote and local IP address endpoints for the rule are reversed for this stack.

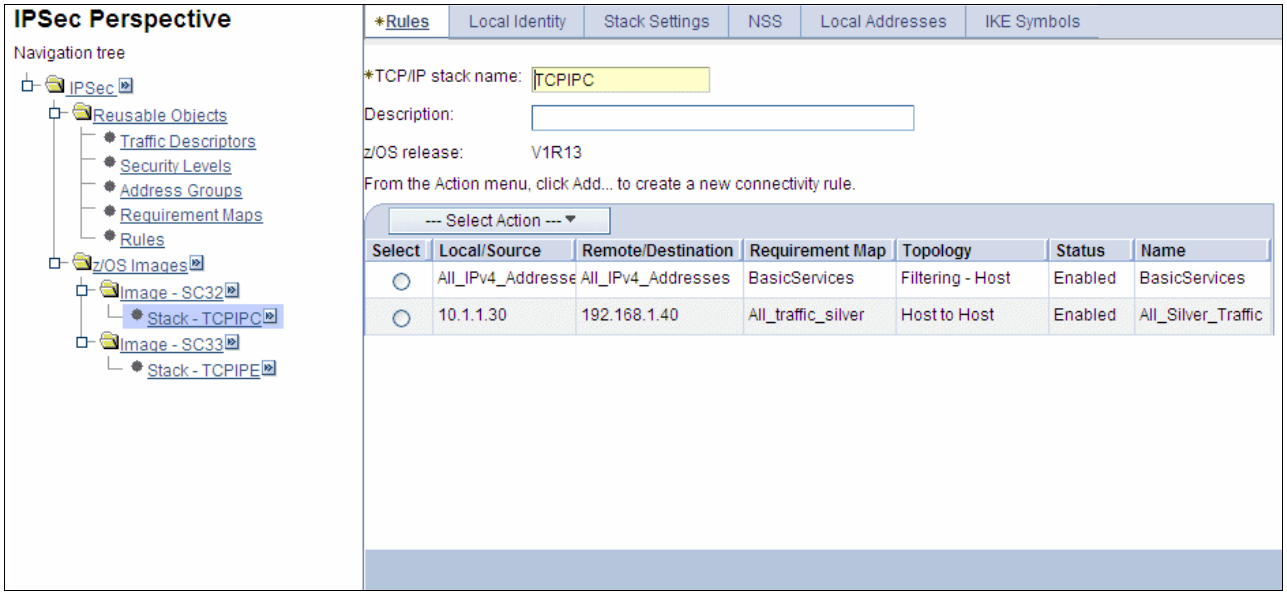


Figure 8-33 Stack TCPIPE on SC33 settings

At this point, you have completed the configuration process for our scenario. Next, you can use the IBM Configuration Assistant to install the configuration files, as described in the next section.

Tip: Click **Health Check** from the IPSec perspective panel before you install the configuration file to the z/OS image. The Health Check reports on errors in the policy. We found it helpful.

8.6.2 Installing the configuration files

After saving the generated configuration for all stacks, install these configuration files on each z/OS image, either using FTP to send the policy files to the z/OS system, or if z/OSMF is running on the same image, by saving to disk. In our environment, we used both methods as described in this section.

FTP policy files

To send the policy files to SC32 we use FTP because z/OSMF is running on SC33:

1. Click the small symbol to the right of **Image-SC32**, and click **Install Configuration Files** in the IBM Configuration Assistant Navigation Tree panel to install the configuration file as shown in Figure 8-34

Figure 8-34 Install Configuration for SC32 by clicking this icon

2. In the List of Configuration Files for Image SC32 panel, select **TCPIPC - IPSecurity Policy** in the configuration files list. To install the selected configuration file in the z/OS image, click **Select Action** → **Install**.

> **Note:** The IKED configuration file created by z/OSMF Configuration Assistant can also be installed by using this method, but in our scenario, we manually created the file as shown in “Create the IKE daemon configuration file” on page 275

3. In the Install File panel (Figure 8-35), select FTP and enter the necessary FTP login information. Give the install file name a meaningful name suffix, or use your preferred file naming convention. We named our file as follows:

   `/etc/cfgasst/v1r13/SC32/TCPIPC/ipsPol_zos2zos_psk`

   Then, click **Go**.

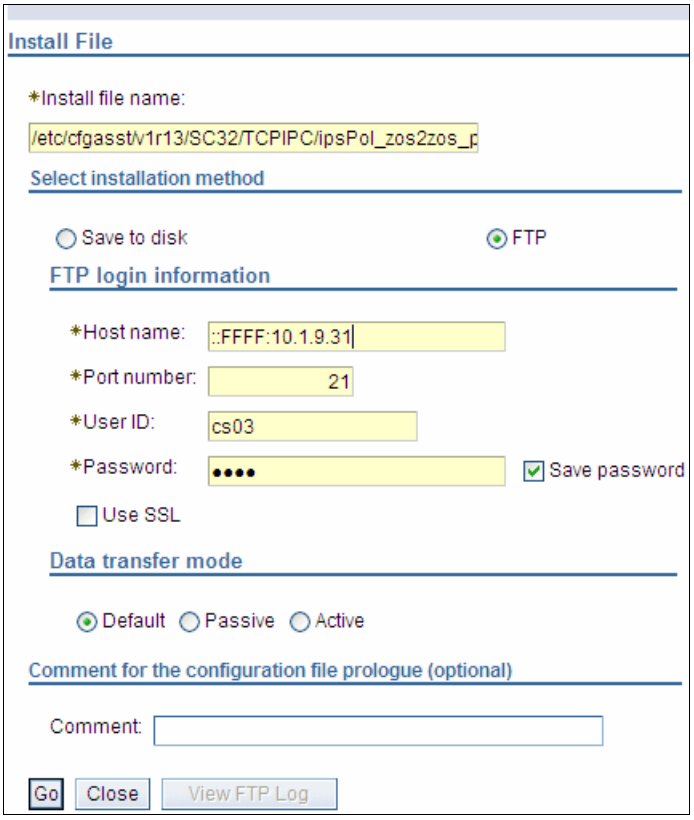


Figure 8-35 Install configuration file through FTP

4. When the FTP is complete click **Close** on the Install File panel, enter optional save history log information and click **OK**. Click **Close** on the List of Configuration Files for Image SC32 panel.

Save policy files to disk

To install the policy files to SC33, we use **Save to disk** because z/OSMF is running on SC33:

1. Click the small symbol to the right of **Image-SC33**, and click **Install Configuration Files** in the IBM Configuration Assistant Navigation Tree panel to install the configuration file as shown in Figure 8-36

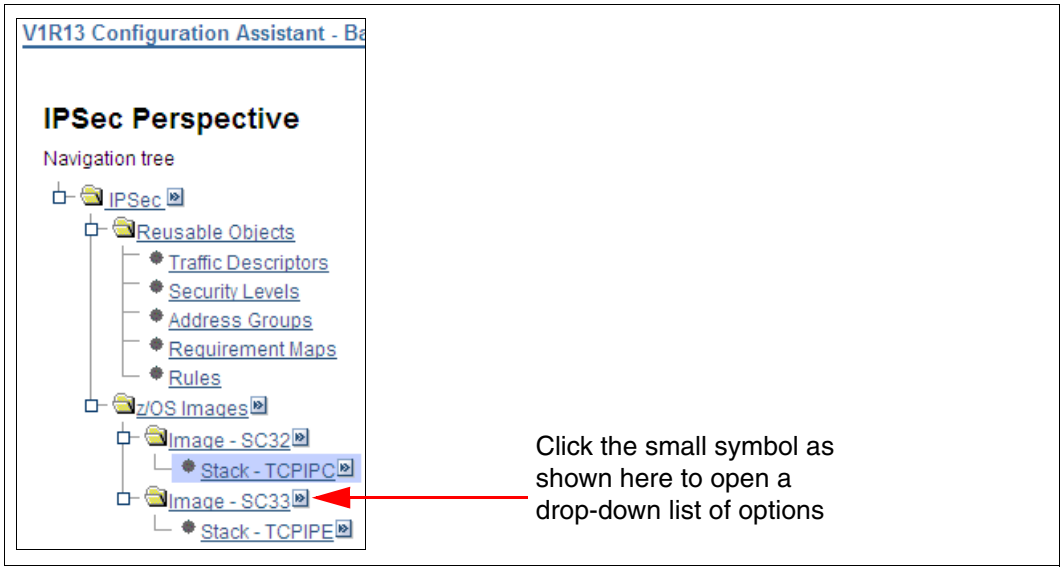


Figure 8-36 Install Configuration for SC33 by clicking this icon

2. In the List of Configuration Files for Image SC33 panel, select **TCPIPE - IPSecurity Policy** in the configuration files list. To install the selected configuration file in the z/OS image, click **Select Action** → **Install**.
3. In the Install File panel, select **Save** to disk. Give the install file name a meaningful name suffix, or use your preferred file naming convention. We gave our file the following name:

   `/etc/cfgasst/v1r13/SC33/TCPIPE/ipsPol_zos2zos_psk`

   Then, click **Go**.

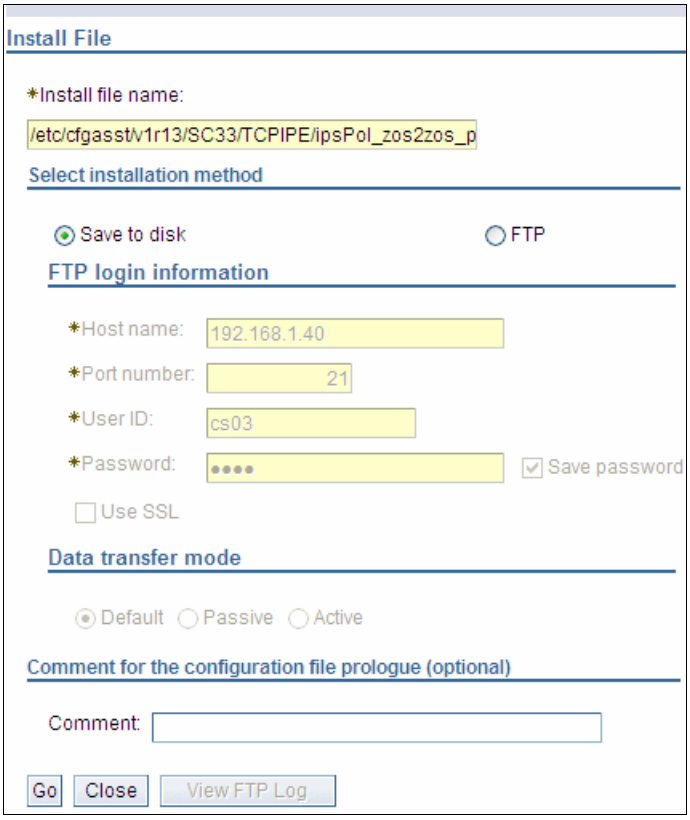


Figure 8-37 Install configuration file through Save to disk

4. When the Save to disk is complete click **Close** on the Install File panel, enter optional save history log information and click **OK**. Click **Close** on the List of Configuration Files for Image SC33 panel.

Updating PAGENT to use the new policy files

After transmitting, or saving, the files to the z/OS image, customize PAGENT to point to the new IP security policy file. In our scenario, we updated the PAGENT configuration file `/SC32/etc/pagent.sc32.tcpipc.conf` to point to our IPSec policy file. Example 8-23 shows part of that file.

Example 8-23 /SC32/etc/pagent.sc32.tcpipc.conf contents with new IPSec policy

```
IPSecConfig /etc/cfgasst/v1r13/SC32/TCPIPC/ipsPol_zos2zos_psk
```

Update the policy agent by using the **`modify pagent,update`** command as shown in Example 8-24. The IPSec tunnel becomes active the next time traffic is generated between the two endpoints.

Example 8-24 The modify pagent,update command results

```
F PAGENT,UPDATE
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPC : IPSEC
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPC
```

We perform the same steps on image SC33. After these commands executed, our scenario was implemented and ready to be used.

8.6.3 Verifying 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:

1. Check that the TCP/IP stack is configured for IP security.
2. Check that the policy agent is active with IP security policy.
3. Display the active IP security policy.
4. Display IKED and dynamic tunnels.

Check that the TCP/IP stack is configured for IP security

To check that the TCP/IP stack is configured for IP security, run the following command from the UNIX shell:

```
netstat -p TCPIPC -f
```

In this command, the **-p** parameter allows a specific stack to be queried. Check that the `IpSecurity` field under the IP Configuration Table is equal to `Yes`. Example 8-25 shows part of the output that is generated by the **netstat** command.

Example 8-25 The netstat -f command for stack TCPIPC on SC32 z/OS image

```
MVS TCP/IP NETSTAT CS V1R13       TCPIP Name: TCPIPC          11:27:39
TCP Configuration Table:
DefaultRcvBufSize:  00065536  DefaultSndBufSize: 00065536
DefltMaxRcvBufSize: 00262144  SoMaxConn:         0000000010
...
IP Configuration Table:
Forwarding: Yes    TimeToLive: 00064  RsmTimeOut:  00060
IpSecurity: Yes
```

Check that the policy agent is active with IP security policy

You need to check that the policy agent has read in the current policies. After you use z/OSMF Configuration Assistant to copy the configuration file to the z/OS image, use the following command 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 receive the following message:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPC : NONE
```

If changes have been made, then you receive the following message instead:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPC : IPSEC
```

Display the active IP security policy

With TRMD running, `syslogd` is the repository for all policy agent and IKE daemon messages. Start an OMVS session and run this command to display the active IP security filters:

```
ipsec -p TCPIPC -f display -a YO
```

The command uses the following parameters:

`-p` Allows specific stacks policies to be queried.
`-f` Indicates filters are to be displayed.
`-a` Indicates to show only a list of all dynamic anchor filters.

In the resulting display, look at the `Source:` field. It shows `Stack Policy`, meaning that the IP security policy is installed and active.

In our scenario, the command shows two policy filters with the type `dynamic anchor` from a total of 34, as shown in Example 8-26.

Example 8-26 The ipsec -f display command for stack TCPIPA, on SC32 z/OS image

```
CS V1R13 ipsec  Stack Name: TCPIPC  Thu Jul 21 11:35:00 2011
Primary:  Filter          Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    Current           TotAvail: 36
Logging:  On              Predecap: Off               DVIPSec:  No
NatKeepAlive:  20         FIPS140:  No
Defensive Mode: Inactive

FilterName:                   ALL_Traffic_Silver~7
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
DefensiveType:                n/a
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Outbound
OnDemand:                     Yes
...
SourceAddress:                10.1.1.30
...
DestAddress:                  192.168.1.40
...

*******************************************************************************
FilterName:                   ALL_Traffic_Silver~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
DefensiveType:                n/a
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
...
SourceAddress:                192.168.1.40
...
DestAddress:                  10.1.1.30
...
*******************************************************************************
2 entries selected
```

In the output listed in Example 8-26, 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. You can display more complete information using the following command:

```
pasearch -p TCPIPC -s DynamicVpn
```

Example 8-27 shows the resulting display.

Example 8-27 pasearch -p TCPIPC -s DynamicVpn display

```
CSO3 @ SC32:/u/csO3>pasearch -p TCPIPC -s DynamicVpn

TCP/IP pasearch CS V1R13                  Image Name: TCPIPC
  Date:                 07/21/2011        Time:  11:43:16
  IPSec Instance Id:    1311261936

policyRule:             All_Silver_Traffic~7
  Rule Type:            IpFilter
  Version:              3                 Status:            Active
  Weight:               105               ForLoadDist:       False
  Priority:             5                 Sequence Actions:  Don't Care
  No. Policy Action:    2                 ConditionListType: CNF
  IpSecType:            policyIpFilter
  policyAction:         IpSec__LogNo
   ActionType:          IpFilter GenericFilter
   Action Sequence:     0
  policyAction:         IPSec__Silver
   ActionType:          IpFilter DynamicVpn
```

You can use the **-t** parameter of the **ipsec** command for traffic test. The following command returns all the rules in the current filter table that match the given traffic type:

```
ipsec -p TCPIPC -t 10.1.1.30 192.168.1.40 tcp 21 0 out
```

Running this command from SC32 (IP: 10.1.1.30) results in the output of all matching rules in the current filter table of SC32 that match FTP traffic, as shown in Example 8-28.

Example 8-28 The output of ipsec -t command on the SC32 z/OS image

```
CS V1R13 ipsec  Stack Name: TCPIPC  Thu Jul 21 11:48:37 2011
Primary:  IP Traffic Test Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    n/a               TotAvail: 2
TestData: 10.1.1.30  192.168.1.40  tcp 21 0 out
Defensive Mode: Inactive

FilterName:                  All_Silver_Traffic~7
FilterNameExtension:         1
GroupName:                   n/a
LocalStartActionName:        n/a
VpnActionName:               IPSec__Silver
TunnelID:                    Y0
Type:                        Dynamic Anchor
DefensiveType:               n/a
State:                       Active
Action:                      Permit
Scope:                       Local
Direction:                   Outbound
OnDemand:                    Yes
...
SourceAddress:               10.1.1.30
...
DestAddress:                 192.168.1.40
...
```

In the output shown in Example 8-28 on page 303, the first matching rule is a rule with action type `IPSec_Silver` (which means the tunnel is secured), the Tunnel ID is `Y0` (which means a dynamic tunnel is created and that the traffic direction is `Outbound`).

Display IKED and dynamic tunnels

To check that communication is working under IP security, use **ftp**. If you have more than one TCP/IP stack in your z/OS image, run **ftp** in the following format:

```
ftp -p TCPIPC 192.168.1.40
```

In this example, the command is run from SC32, using stack TCPIPC. After the **ftp** command completes successfully, use the following command to display the IKE tunnel:

```
ipsec -p TCPIPC -k display
```

Example 8-29 shows the output from this command.

Example 8-29 IPSec active IKE tunnels display in SC32, after ftp to SC33

```
CSO3 @ SC32:/u/csO3>ipsec -p TCPIPC -k display
CS V1R13 ipsec  Stack Name: TCPIPC  Thu Jul 21 O8:28:26 2011
Primary:  IKE tunnel      Function: Display           Format:   Detail
Source:   IKED            Scope:    Current           TotAvail: n/a

TunnelID:                     K1
Generation:                   1
IKEVersion:                   2.0
KeyExchangeRuleName:          All_Silver_Traffic~5
KeyExchangeActionName:        All_Silver_Traffic
LocalEndPoint:                1O.1.1.30
LocalIDType:                  ID_IPV4_ADDR
LocalID:                      1O.1.1.30
RemoteEndPoint:               192.168.1.40
RemoteIDType:                 ID_IPV4_ADDR
RemoteID:                     192.168.1.40
ExchangeMode:                 n/a
State:                        DONE
AuthenticationAlgorithm:      HMAC-SHA1-96
EncryptionAlgorithm:          DES-CBC
 KeyLength:                   n/a
PseudoRandomFunction:         HMAC-SHA1
DiffieHellmanGroup:           1
LocalAuthenticationMethod:    PresharedKey
RemoteAuthenticationMethod:   PresharedKey
InitiatorCookie:              0xFCF33ACBE0D09BF8
ResponderCookie:              0x52BAFAD309F3122F
Lifesize:                     OK
CurrentByteCount:             416b
Lifetime:                     480m
LifetimeRefresh:              2011/07/21 16:11:56
LifetimeExpires:              2011/07/21 16:17:12
ReauthInterval:               Om
ReauthTime:                   n/a
Role:                         Initiator
AssociatedDynamicTunnels:     1
NATTSupportLevel:             None
NATInFrntLclScEndPnt:         No
```

```
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     No
RmtNAPTDetected:              No
RmtUdpEncapPort:              n/a
```

Tip: The **ipsec** command has three main display functions that you can use 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 IKE 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, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Use the following command to display the active dynamic tunnels:

```
ipsec -p TCPIPC -y display
```

Example 8-30 shows the output from this command.

Example 8-30 IPSec active dynamic tunnels display on SC32, after ftp to SC33

```
CSO3 @ SC32:/u/csO3>ipsec -p TCPIPC -y display
CS V1R13 ipsec  Stack Name: TCPIPC  Thu Jul 21 08:28:36 2011
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y2
Generation:                   1
IKEVersion:                   2.0
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-SHA1
 AuthInboundSpi:              850964131  (0x32B8AEA3)
 AuthOutboundSpi:             1831121463 (0x6D24B237)
HowToEncrypt:                 DES-CBC
 KeyLength:                   n/a
 EncryptInboundSpi:           850964131  (0x32B8AEA3)
 EncryptOutboundSpi:          1831121463 (0x6D24B237)
Protocol:                     ALL(0)
```

```
LocalPort:                    n/a
LocalPortRange:               n/a
RemotePort:                   n/a
RemotePortRange:              n/a
Type:                         n/a
TypeRange:                    n/a
Code:                         n/a
CodeRange:                    n/a
OutboundPackets:              5
OutboundBytes:                168
InboundPackets:               5
InboundBytes:                 306
Lifesize:                     0K
LifesizeRefresh:              0K
CurrentByteCount:             0b
LifetimeRefresh:              2011/07/21 12:06:52
LifetimeExpires:              2011/07/21 12:17:12
CurrentTime:                  2011/07/21 08:28:36
VPNLifeExpires:               2011/07/22 08:17:12
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
PassthroughDF:                n/a
PassthroughDSCP:              n/a
```

8.7 Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1

Tip: Also see Appendix C, “Configuring IPSec between z/OS and Windows” on page 859.

In 8.6, “Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2” on page 285, we described how to configure IKE with pre-shared keys only. In this section, we describe how to change the IKE configuration using pre-shared key (Figure 8-38 on page 307) to a configuration that builds the encrypted tunnel using RSA signature mode.

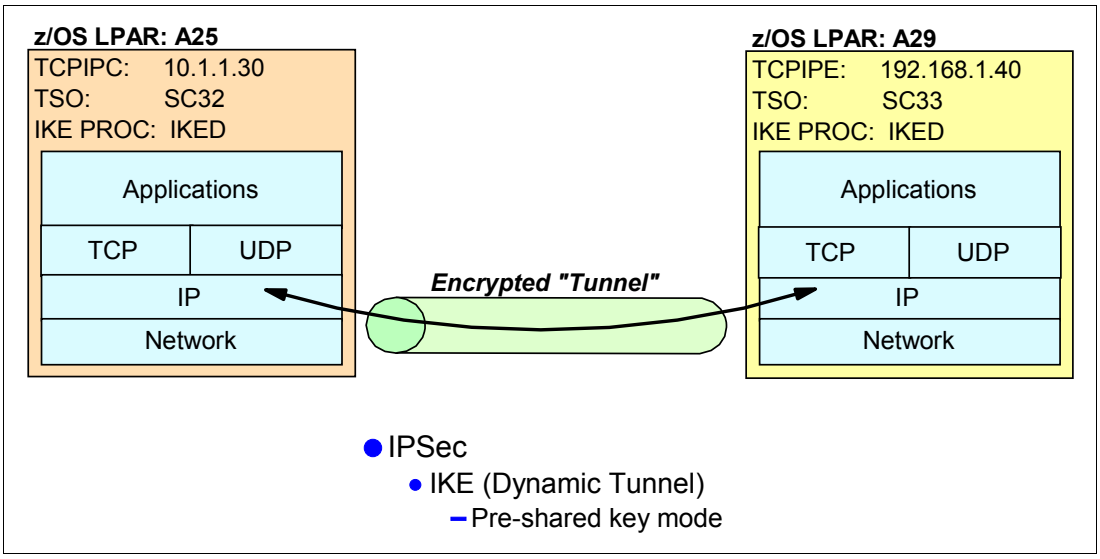


Figure 8-38 VPN tunnel between TCPIPA on SC32 and TCPIPE on SC33 using pre-shared key mode

Tip: The Network Security Services (NSS) solution described in Chapter 9, “Network Security Services for IPSec clients” on page 325 relies on IKE with digital signature authentication. If you decide to implement NSS with IKED as an NSS client, you will want to refer to the RSA signature mode examples in this section. With IKEv2, using NSS for digital certificate management is required.

8.7.1 Generating certificates for IKEv1 RSA signature mode

Figure 8-39 illustrates the new scenario.

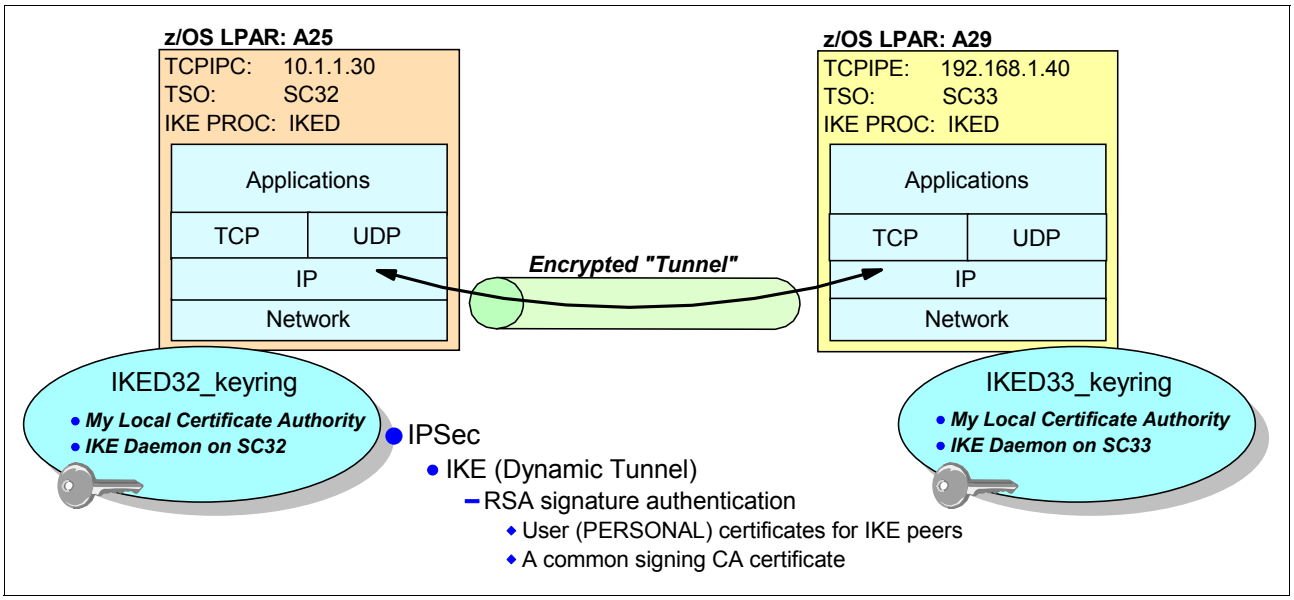


Figure 8-39 VPN tunnel between TCPIPC on SC32 and TCPIPE on SC33 using RSA mode

Tip: For IKEv1, the term RSA signature mode is the same as digital signature mode. The Configuration Assistant windows refers to it as digital signature mode. In this book it is referred as RSA mode in many places because our examples use IKEv1.

IKEv1 with RSA signature mode requires key ring access, as shown in Figure 8-39 on page 307. For this scenario, we use two key rings, but you can also have a shared key ring between SC32 and SC33. Using two key rings for this scenario, which does not rely on NSS, simplifies the explanation of how you can migrate to a scenario that does rely on NSS.

Tip: For details about creating key rings, certificate authority certificates, and personal certificates, see 8.5.9, “RACF certificate definitions for IKED” on page 279

The IKED32_keyring is populated with the following certificates on SC32:

- A certificate authority certificate that has signed the certificate that resides on IKED’s key ring. That is, the CA certificate is the one that has signed IKED’s user (PERSONAL) certificate. The label on the CA certificate is `My Local Certificate Authority`. A description of how to create this CA certificate is shown in “Generating a self-signed certificate to represent the local certificate authority” on page 280.
- A user (PERSONAL) certificate representing the IKED peer on SC32. The label on the certificate is IKE Daemon on SC32. This is used by peer IKEs to authenticate SC32. A description of how to create this personal certificate is shown in “Creating a certificate for the server” on page 281.

The IKED33_keyring is populated with the following certificates on SC33:

- A certificate authority certificate that has signed the certificate that resides on IKED’s key ring. That is, the CA certificate is the one that has signed IKED’s user (PERSONAL) certificate. The label on the CA certificate is `My Local Certificate Authority`. A description of how to create a CA certificate is shown in “Generating a self-signed certificate to represent the local certificate authority” on page 280.
- A user (PERSONAL) certificate representing the IKED peer. The label on the certificate is IKE Daemon on SC33. This is used by peer IKEs to authenticate SC33. A description of how to create a personal certificate is shown in “Creating a certificate for the server” on page 281.

Tip: Our scenario uses the same signing certificate authority for both of the PERSONAL certificates. You might encounter situations in which there are two separate CAs for the two user certificates. You need to ensure that the correct CA certificate which signed the IKE peer’s user certificate is connected to the correct key ring.

8.7.2 Creating the IPSec filters and policies for the IPSec tunnel

We require IKE processes for the two IPSec partners:

- One partner on SC32
- One partner on SC33

We need to prepare the IPSec filter definitions and policies for the two systems. For this scenario, we use the same policies that were configured and shown in the following sections:

- 8.5.3, “Updating the TCP/IP stack to activate IPSec” on page 266
- 8.5.4, “Restricting the use of the ipsec command” on page 267

- 8.5.5, “Installing the IBM Configuration Assistant for z/OS Communications Server” on page 267
- 8.5.7, “Defining the IPSec policies to PAGENT” on page 269
- 8.5.8, “Setting up the IKE daemon” on page 274

In our scenario for RSA signature mode, we need to make a change to the IBM Configuration Assistant files we created earlier, as explained in the next section.

8.7.3 Modifying existing policies to use RSA signature mode

If you build a scenario similar to that described in this chapter, you can modify that policy by editing the existing IPSec Policy File, or you can change the policy with the help of the IBM Configuration Assistant GUI. We chose to save a copy of the existing IBM Configuration Assistant backing store file and to make the necessary changes there.

The IPSec implementation between SC32_TCPIPC and SC33_TCPIPE (described in 8.6, “Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2” on page 285) is defined in the backing store file named `IPSec_zos2zos_psk`. Our plan was to keep that file as our backup and to create a new backing store file that includes RSA mode.

Thus, to create a new backing store file in the IBM Configuration Assistant, perform the following actions:

1. From the IPSec Perspective main panel, click the small icon to the right of Action, select **Save As**, and provide the name `IPSec_zos2zos_RSA`.
2. Click **OK,** then **OK** again to verify you are now working with the newly saved backing store file.

Next, change the IPSec policy for TCPIPC on SC32 and TCPIPE on SC33 to a policy that supports RSA mode and IKEv1. Complete the following steps:

1. In the IPSec Perspective main panel, select **Stack-TCPIPC** in the navigation tree left menu and click the Stack Settings tab. For the default initiator mode, choose IKEv1 main. Then, click **OK**, as shown in Figure 8-40 on page 310.

> **Note:** We show an example using IKEv1 here, which does not require NSS. To see examples of using IKEv2 with NSS see Chapter 9, “Network Security Services for IPSec clients” on page 325.

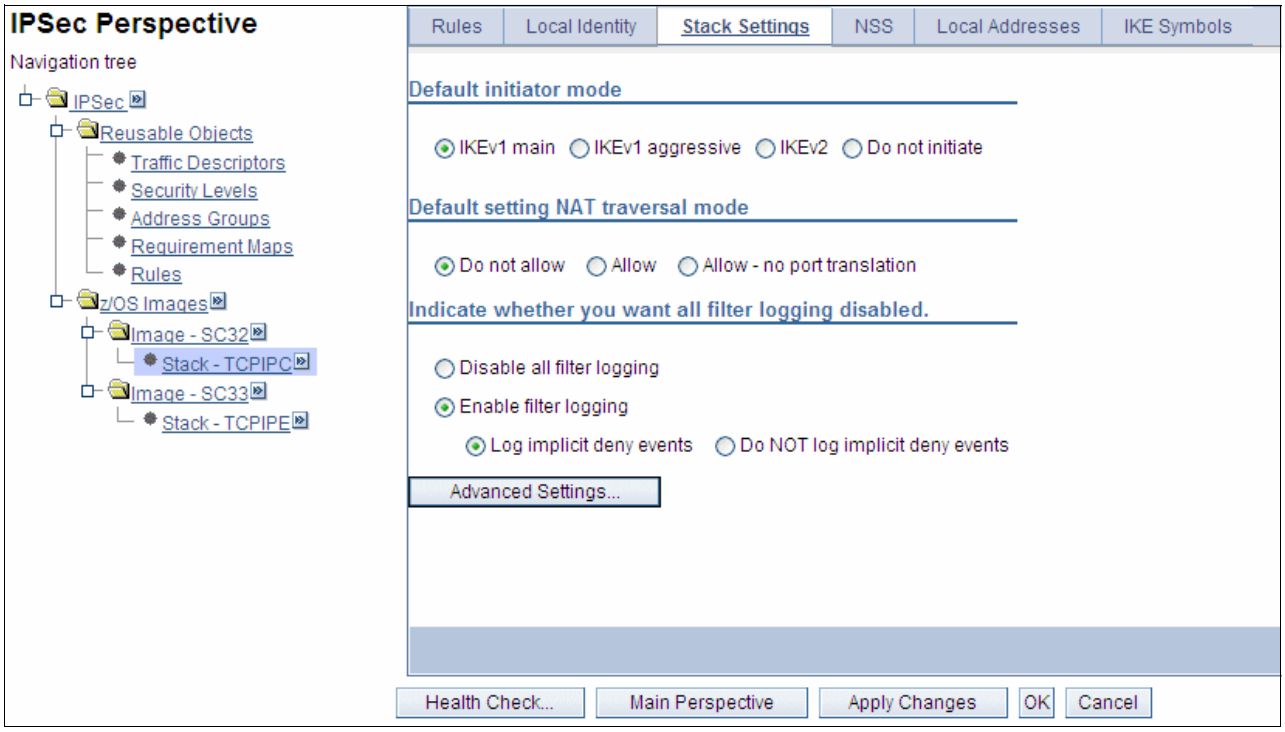


Figure 8-40 Stack settings for IKEv1 with RSA

2. Click **Stack-TCPIPC**, select the connectivity rule that uses IPSec, in our case All_Silver_Traffic, and then click **Select Action** → **Modify** as shown in Figure 8-41.

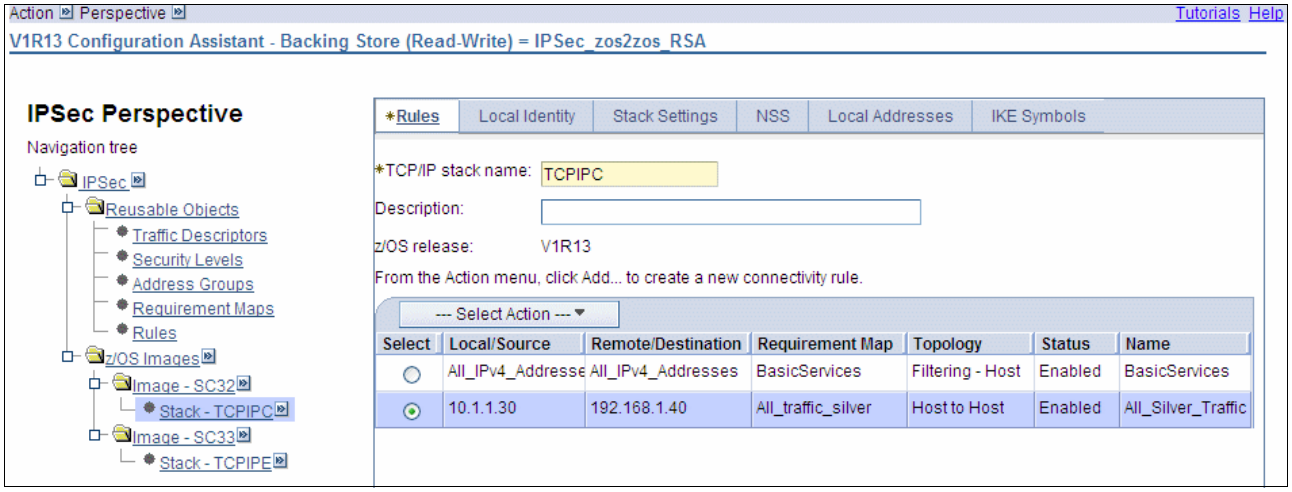


Figure 8-41 TCPIPC All_Silver_Traffic rule

3. Click the **Remote Security Endpoint** tab. Note that the authentication of remote IKE peers in the copied configuration uses pre-shared keys. See Figure 8-28 on page 292. You need to change this selection to Digital mode for this scenario. For IKEv1, this implies the RSA mode.

   Also note that the remote identity is depicted by means of an IP address in the previous, pre-shared key scenario. RSA mode requires authentication through the exchange of identities that have been coded in an x.509 certificate. Our certificate uses the fields `CN`, `OU`, and `C`. Together, these fields comprise the *x.500 distinguished name*. The x.500

distinguished name field also represents a valid identity for an IKE connection. Unless the distinguished name is used, the identity must appear in the Subject Alternative Name field of the personal certificate used in RSA authentication.

You can select any one of the following identities:

- IP address: Equivalent to the RACF Subject ALTNAME field, IP.
- Fully qualified domain name (FQDN): Equivalent to the RACF Subject ALTNAME field, DOMAIN.
- User id @ FQDN: Equivalent to the RACF Subject ALTNAME field, EMAIL.
- x.500 distinguished name: The full name of the Subject, not an alternate name.

In our certificate definition, we defined an ALTNAME for the IP address, and also the other possible alternate name types, so we can leave the remote identity as an IP address. Click the Local Security Endpoint tab.

Note: If your certificates do not have ALTNAME coded, use the distinguished name as described in “Alternative IKE endpoint identities using x.500 distinguished name” on page 312

4. In the Local Security Endpoint tab, note that the local identity is depicted by means of an IP address in the previous, pre-shared key scenario. Again, because we have defined an ALTNAME for the IP address in our certificate, we can leave this as it is. Click **OK**.
5. Select **Apply Changes** and click **OK** to return to the IPSec Perspective main panel.

Next, you perform the same steps to change the Connectivity Rules for TCPIPE:

1. In the IPSec Perspective panel, select the stack **TCPIPE**, and select the All_Silver_Traffic Connectivity Rule. Click **Select Action** → **Modify**, and proceed through the same steps that we executed for TCPIPC on SC32, remembering to use IKEv1 to match what is set for TCPIPC on SC32.
2. In the Remote Security Endpoint tab, select Digital Signature, and leave the remote identity as an IP address.
3. Select the Local Security Endpoint tab and note the identity type is still set to IP address. This is fine, because we have the IP address defined as an ALTNAME in the RACF certificate. Click **OK**. Select **Apply Changes**.

Alternative IKE endpoint identities using x.500 distinguished name

If you do not code alternate names in your RACF certificate you must identify the security endpoint by the x.500 distinguished name from the certificate for SC32. To do so, complete the Local IKE identity field with a type of `X.500 distinguished name` and the following value as shown in Figure 8-42:

```
CN=IKE Daemon on SC32,OU=ITSO,C=US
```

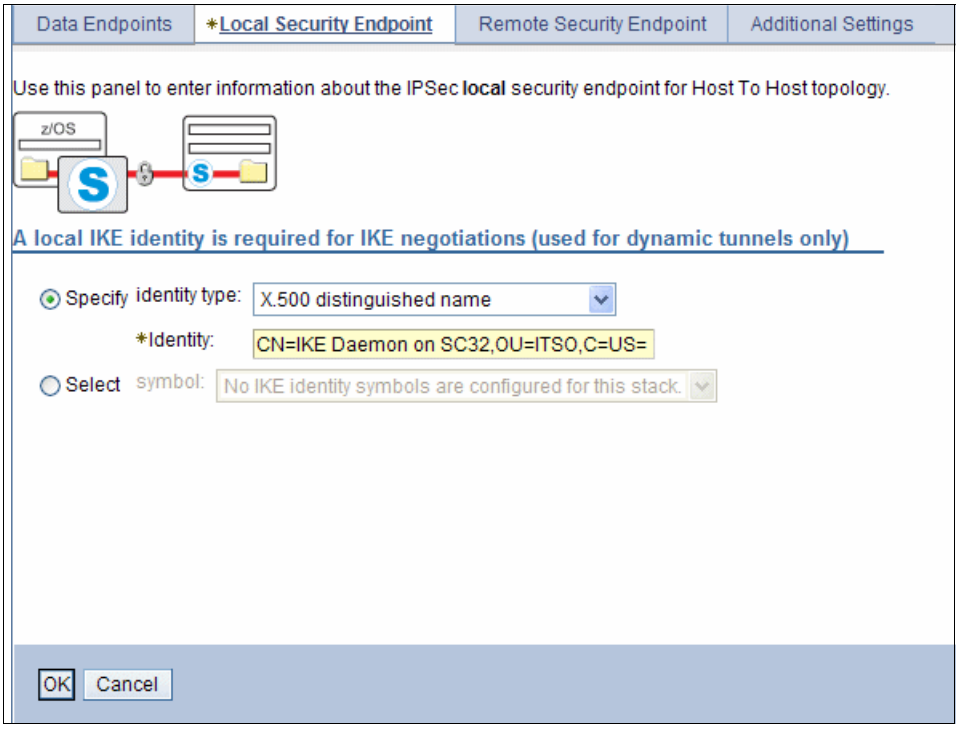


Figure 8-42 Using X.500 distinguished name as local IKE identity

This value was coded in our certificate definition, and we used the sequence that is displayed in the certificate subject name when we issued the following command:

```
racdcert id(IKED) list(label('IKE Daemon on SC32'))
```

The output at **1** in Example 8-31 shows the certificate Subject's Name. It is in the order that was entered in the `X.500 Distinguished Name` field of Figure 8-42 and this order is important. The RACF definition can 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.

Example 8-31 Display of certificate for SC32

```
Digital certificate information for user IKED:

  Label: IKE Daemon on SC32
  Certificate ID: 2QTJOsXEydLFQMSBhZSWlUCWlUDiw/Py
  Status: TRUST
  Start Date: 2010/11/11 00:00:00
  End Date:   2012/11/11 23:59:59
  Serial Number:
       >0F<
  Issuer's Name:
       >OU=ITSO z/OS CS.O=I.B.M Corporation.C=US<
```

```
Subject's Name:
     >CN=IKE Daemon on SC32.OU=ITSO.C=US<          1
Subject's AltNames:
  IP: 10.1.1.30
  EMail: sc32a at sc32a.itso.raleigh.ibm.com
  Domain: sc32a.itso.raleigh.ibm.com
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
  Ring Owner: IKED
  Ring:

     >IKED32_keyring<
```

Select the remote security endpoint tab and complete the remote IKE identity field with a type of `X.500 distinguished name` and the following value as shown in Figure 8-43.

```
CN=IKE Daemon on SC33,OU=ITSO,C=US
```

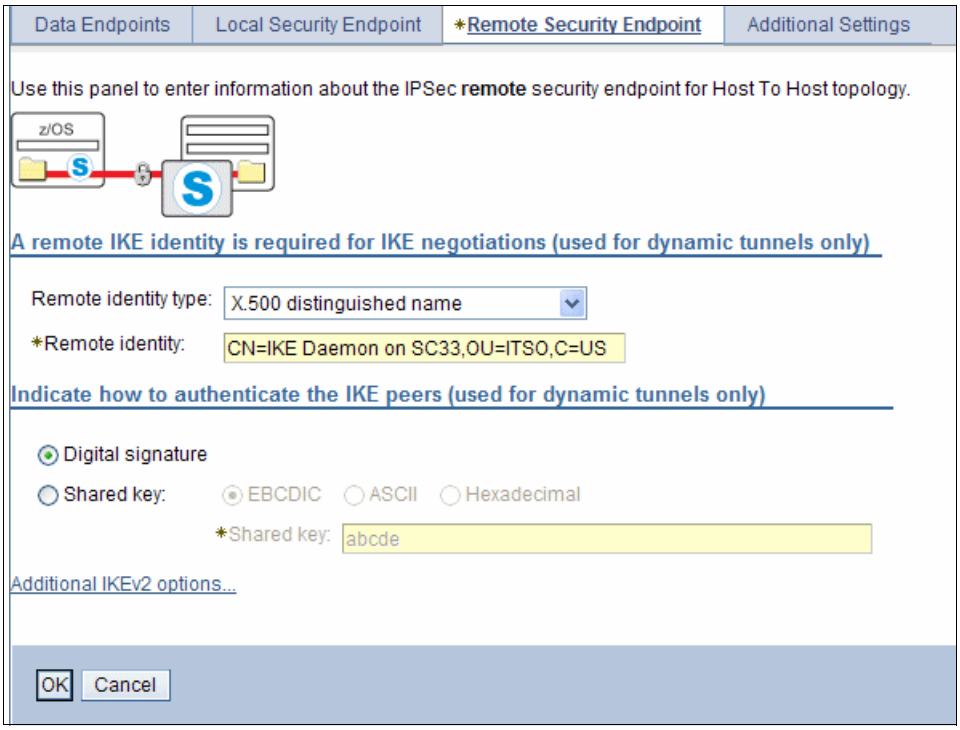


Figure 8-43 Using X.500 distinguished name as remote IKE identity

Important: The racdcert display separates the individual fields (also known as RDN) of the Distinguished Name (DN) with periods as delimiters. The display pattern is as follows:

```
RDN<period>RDN<period>RDN<period>RDN
```

However, you see that the syntax required in the IKE definition panel (Figure 8-42) requires a comma (,) as a delimiter. The coding pattern in this case is as follows:

```
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, which builds the `iked.conf` file from these definitions.

There are many RDNs that can be used to build the x.500 distinguished name. Review Table 9-1 on page 370 for a list of other valid RDNs.

Building new IPSec policies to use RSA signature mode

We used the existing IPSec policies that we saved from the previous backing store file. We simply changed the IKE process in our new backing store file to use IKEv1 and from pre-shared key mode to **digital signature mode**. If you need to build new IPSec policies because you have no file to save as and modify, see the description of IPSec policies that we describe in 8.5, “How IPSec is implemented” on page 264.

Building the IKE Daemon files for RSA mode

In our case, we had already created the IKE Daemon configuration files manually as described in “Create the IKE daemon configuration file” on page 275. You can also create the IKE Daemon configuration files with the help of the IBM Configuration Assistant. To create them, complete the following steps:

1. Return to the IPSec Perspective of the IBM Configuration Assistant and select **MVS Image-SC32**. In the panel that opens, click the IKE tab.
2. Change the key ring on the panel to reflect the new SC32 key ring that is used for the IKED procedure. We replaced the name of the key ring with `IKED/IKED32_keyring`, which is the key ring that we built to contain the x.509 certificates for RSA signature mode. Click **OK**.
3. Next select the SC33 image from the IPSec Perspective. Move to the IKE settings tab and again, change the name of the key ring to reflect the ring for RSA signature authentication at IKED on SC33. We replaced the name of the key ring with `IKED/IKED33_keyring`, which is the key ring that we built to contain the x.509 certificates for RSA mode at SC33. Click **OK**.

Sending IPSec and IKE configuration files to the mainframe

In this section, we describe how to send IPSec and IKE configuration files to the mainframe.

Working with IPSec policies

To install the changed IPSec policies so PAGENT can access them, follow the steps described in 8.6.2, “Installing the configuration files” on page 297.

Use a meaningful name suffix if you choose to use the naming convention used by z/OSMF Configuration Assistant. In our case we use the following names:

- `/etc/cfgasst/v1r13/SC32/TCPIPC/ipsPol`**`_zos2zos_RSA`** for image SC32
- `/etc/cfgasst/v1r13/SC33/TCPIPE/ipsPol`**`_zos2zos_RSA`** for image SC33

Alternatively, use your preferred file naming convention.

At the point where you see the List of Configuration Files for Image SC32 panel, you can view the configuration file by selecting the IP Security Policy and clicking **Select Action → Show Configuration File**. Example 8-32 shows that in one portion of the configuration file for the IPSec policy for TCPIPC there is a `KeyExchangeOffer` named `KEO~10` that designates `RsaSignature` as the method for IKE peer authentication.

Example 8-32 TCPIPC: New KeyExchangeOffer KEO~10: Authenticate Peers with Rsa

```
KeyExchangeOffer KEO~10
{
HowToAuthMsgs SHA1
HowToAuthPeers RsaSignature
DHGroup Group1
RefreshLifetimeProposed 480
RefreshLifetimeAccepted 240 1440
RefreshLifesizeProposed None
RefreshLifesizeAccepted None
}
```

Continue looking at the policy file to find another change to the original policies for TCPIPC, as Example 8-33 shows.

Example 8-33 Noting the changed addresses

```
LocalSecurityEndpoint All_Silver_Traffic~LSE~4
{
Identity IpAddr 10.1.1.30
LocationRef All_Silver_Traffic~ADR~1
}

RemoteSecurityEndpoint All_Silver_Traffic~RSE~3
{
Identity IpAddr 192.168.1.40
LocationRef All_Silver_Traffic~ADR~2
}
```

Observe how the `LocalSecurityEndpoint` and the `RemoteSecurityEndpoint` reflect the IP addresses that are defined as ALTNAME IP in our RACF certificates. The files related to SC33 reflect the same information. These fields are part of the certificates that each IKE peer will present to the other.

- At TCPIPC on SC32, change the PAGENT image file to point to the new IPSec policy sent to SC32 in a previous step, as Example 8-34 shows.

  Example 8-34 /SC32/etc/pagent.sc32.tcpipc.conf

  ```
  IPSecConfig /etc/cfgasst/v1r13/SC32/TCPIPC/ipsPol_zos2zos_RSA
  ```

- At TCPIPE on SC33, also change the PAGENT image file to point to the RSA policy file as Example 8-35 shows.

  Example 8-35 /SC33/etc/pagent.sc33.tcpipe.conf

  ```
  IPSecConfig /etc/cfgasst/v1r13/SC33/TCPIPE/ipsPol_zos2zos_RSA
  ```

Working with IKE Daemon configuration files

Remember that you can use two methods to create the IKE Daemon configuration files:

- Use the IBM Configuration Assistant to update the files automatically as described in "Building the IKE Daemon files for RSA mode" on page 314.
- Edit the files manually at the mainframe as described in "Create the IKE daemon configuration file" on page 275.

So, keep in mind that you need to modify the IKE Daemon file when you convert to RSA mode. Specifically, you must at least point to the correct key ring in which the x.509 certificates for the RSA process are stored.

8.7.4 Verifying IKE with RSA signature mode

This section shows the areas that you need to verify to ensure that IPSec is operational when using RSA mode.

Initializing TRMD, IKED, and PAGENT at SC32 and SC33

The IKED and Pagent policies at SC32 and SC33 are already revised. To verify IKED with the new RSA mode, stop both IKED and policy agent on each stack and start them again with the new definitions.

Reviewing SYSLOG daemon logs for information about the initialization

Review the IKE messages in syslogd to determine whether the user (PRIVATE) certificates for SC32 and SC33 have been found during IKE initialization as shown in Example 8-36.

Example 8-36 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      1
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 RSA   certificate with label IKE Daemon on SC32  2
IKE: End display of Cert Cache
IKE: Initialization of Cert Cache Complete
```

The numbers in this example correspond to the following information:

1. Shows the beginning of the display of the user certificates that are used to identify local and remote endpoints.

2. Shows 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 8-37.

Example 8-37 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 RSA   certificate with label IKE Daemon on SC33  1
IKE: End display of Cert Cache
IKE: Initialization of Cert Cache Complete
```

At **1**, the IKE daemon reveals that it has processed the certificate that identifies the IKE Daemon on SC33 successfully.

Tip: If `SupportedCertAuth` was coded in the IKE configuration file, the display of the CA cache for IKE lists `My Local Certificate Authority` in the output shown in Examples Example 8-36 on page 316 and Example 8-37 on page 317.

Issue a **pasearch** command to determine if the proper IPSec policy Key Exchange rule was loaded:

```
pasearch -p TCPIPC -v k
```

Example 8-38 shows the output for this command.

Example 8-38 IPSec policy Key Exchange rule on SC32

```
TCP/IP pasearch CS V1R13                  Image Name: TCPIPC
  Date:                 07/22/2011        Time:  13:08:59
  IPSec Instance Id:    1311343601

policyRule:             All_Silver_Traffic~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_Silver_Traffic
   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
```

```
IpSec Condition Summary:                NegativeIndicator: Off
 KeyExchange Condition:
  LocalSecurityEndPoint:
   Location:
    FromAddr:          10.1.1.30 1
    ToAddr:            10.1.1.30
   Identity:
    IpAddr:
     FromAddr:         10.1.1.30 2
     ToAddr:           10.1.1.30
  RemoteSecurityEndPoint:
   Location:
    FromAddr:          192.168.1.40 3
    ToAddr:            192.168.1.40
   Identity:
    IpAddr:
     FromAddr:         192.168.1.40 4
     ToAddr:           192.168.1.40
```

When you examine the **pasearch** output, check that the IPSec policies accurately reflect the intended environment as follows:

1. The local security endpoint address is correct.
2. The Identity that was extracted from the certificate is the alternate name IP address.
3. The remote security endpoint IP address is also correct.
4. The identity that was extracted from the certificate of the remote partner is also its alternate name IP address.

At SC33's OMVS shell, use the **ftp** command to the TCPIPC stack from TCPIPE to cause a VPN to be created on demand:

```
ftp -p TCPIPE 10.1.1.30
```

Display the dynamic IKE tunnel from SC32:

```
ipsec -p TCPIPC -k display
```

Example 8-39 shows the output for this command.

Example 8-39 Display of the IKE tunnel between SC32 and SC33

```
CS V1R13 ipsec  Stack Name: TCPIPC  Fri Jul 22 13:34:54 2011
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                     K1          1
Generation:                   1
IKEVersion:                   1.0
KeyExchangeRuleName:          All_Silver_Traffic~5
KeyExchangeActionName:        All_Silver_Traffic
LocalEndPoint:                10.1.1.30
LocalIDType:                  ID_IPV4_ADDR
LocalID:                      10.1.1.30    2
RemoteEndPoint:               192.168.1.40
RemoteIDType:                 ID_IPV4_ADDR
RemoteID:                     192.168.1.40 3
ExchangeMode:                 Main
```

```
State:                          DONE
AuthenticationAlgorithm:        HMAC-SHA1
EncryptionAlgorithm:            DES-CBC
 KeyLength:                     n/a
PseudoRandomFunction:           HMAC-SHA1
DiffieHellmanGroup:             1
LocalAuthenticationMethod:      RsaSignature 4
RemoteAuthenticationMethod:     RsaSignature 4
InitiatorCookie:                0x4C67C55DBBF9B5B0
ResponderCookie:                0xDE62A6172669A766
Lifesize:                       0K
CurrentByteCount:               376b
Lifetime:                       480m
LifetimeRefresh:                2011/07/22 21:22:56
LifetimeExpires:                2011/07/22 21:31:36
ReauthInterval:                 480m
ReauthTime:                     2011/07/22 21:22:56
Role:                           Responder    5
AssociatedDynamicTunnels:       1
NATTSupportLevel:               None
NATInFrntLclScEndPnt:           No
NATInFrntRmtScEndPnt:           No
zOSCanInitiateP1SA:             Yes
AllowNat:                       No
RmtNAPTDetected:                No
RmtUdpEncapPort:                n/a
*******************************************************************************
```

Example 8-39 on page 318 shows several important pieces of information:

1. K1 is the identifier for the IKE tunnel.

2. The local IKE ID.

3. The remote IKE ID.

4. The authentication method is RSA signature.

5. The SC33 side of the IKE tunnel initiated the tunnel. SC32 is the Responder.

We display the dynamic tunnel over which the FTP traffic flowed:

```
ipsec -p TCPIPC -y display
```

Example 8-40 shows the output for this command.

Example 8-40 Display the dynamic tunnel between SC32 and SC33

```
CS V1R13 ipsec  Stack Name: TCPIPC  Fri Jul 22 13:34:58 2011
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                       Y2 1
Generation:                     1
IKEVersion:                     1.0
ParentIKETunnelID:              K1 2
VpnActionName:                  IPSec__Silver
LocalDynVpnRule:                n/a
State:                          Active
```

```
HowToEncap:                   Transport 3
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 4
 AuthAlgorithm:               HMAC-SHA1
 AuthInboundSpi:              1289215922 (0x4CD7DFB2)
 AuthOutboundSpi:             1070565066 (0x3FCF86CA)
HowToEncrypt:                 DES-CBC 5
 KeyLength:                   n/a
 EncryptInboundSpi:           1289215922 (0x4CD7DFB2)
 EncryptOutboundSpi:          1070565066 (0x3FCF86CA)
Protocol:                     ALL(0) 6
...
```

Example 8-40 on page 319 shows multiple important pieces of information:

1. Y2 is the identifier for the dynamic tunnel that carried the ftp traffic.
2. K1 is the IKE tunnel over which was negotiated the phase 2 security association.
3. Transport mode is in use for the host-to-host connection.
4. Encapsulating Security Protocol (ESP) is in use, because we are authenticating and encrypting the data.
5. The data on this IPSec connection is subject to DES encryption.
6. All protocols (TCP, UDP, ICMP, and so on) are supported.

> **Note:** Although all protocols are supported, because the BasicServices connectivity rule exists, and is above the All_Silver_Traffic rule in the rules table, ping, resolver, DNS, OSPF, and PAGENT traffic are permitted without using encryption. For an example of what would happen if ping traffic was also required to traverse a dynamic tunnel, see Figure on page 321

Next, check the logs at SC33 to determine whether you have IKED and TRMD syslogd messages relating to successful IKE and phase 2 tunnel activation. An example of our messages is shown in Example 8-41.

Example 8-41 IKED and TRMD messages for SC33

```
*** SA Context Information ***
 Phase 1 tunnel ID : K0 Generation : 0
 Stackname : TCPIPE
 Local IKE ID info  : ID_IPV4_ADDR 192.168.1.40
 Remote IKE ID info : ID_IPV4_ADDR 10.1.1.30
 Local IKE IP  : 192.168.1.40 port 500
 Remote IKE IP : 10.1.1.30 port 500
 KeyExchangeRuleName : All_Silver_Traffic~5
 Icookie/Rcookie : xC79B2E8E6ACFD5C5 / x0000000000000000
 IKE Version : 1
 Pending phase 2 info:
```

```
 Local IPSec upper-layer info  : N/A N/A
 Remote IPSec upper-layer info : N/A N/A
 Local IPSec IP info : 192.168.1.40
 Remote IPSec IP info : 10.1.1.30
 Protocol : ALL(0)
 *** UREQ Context Information ***
 Type : 2    Delayed : 0    Stack Name : TCPIPE
IKE DEBUGSA : Sending message 1 of Main mode (INIT state) from SA x2BEE0550

*** SA Context Information ***
 Phase 2 tunnel ID : Y0 Generation : 0
 Parent  tunnel ID : K1 Generation : 1
 Stackname : TCPIPE
 Local IPSec IP info : 192.168.1.40
 Remote IPSec IP info : 10.1.1.30
 Protocol : ALL(0)
 IpFilterRuleName : All_Silver_Traffic~7
 AH SPIs in/out : 0 / 0
 ESP SPIs in/out : 0 / 0
 IKE Version : 1
 *** SA Context Information ***
 Phase 1 tunnel ID : K1 Generation : 1
 Stackname : TCPIPE
 Local IKE ID info  : ID_IPV4_ADDR 192.168.1.40
 Remote IKE ID info : ID_IPV4_ADDR 10.1.1.30
 Local IKE IP  : 192.168.1.40 port 500
 Remote IKE IP : 10.1.1.30 port 500
 KeyExchangeRuleName : All_Silver_Traffic~5
 Icookie/Rcookie : xC79B2E8E6ACFD5C5 / xE1360B814CB9CD94
 IKE Version : 1
IKE DEBUGSA : Sending message 1 of Quick mode (INIT state) from SA x2BEE7838

Jul 22 14:22:51 wtsc33/TRMD     TRMDE1   TRMD.TCPIPEÝ50593948¨: EZD0818I Tunnel
added: 07/22/2011 18:22:42.41 vpnaction= IPSec__Silver tunnelID= Y2  AHSPI= 0
ESPSPI=854029929
```

Example of initial ping failing

If ping traffic was also required to traverse a dynamic tunnel because of IPSec policy, the initial ping would fail. Example 8-42 shows the output for this example.

Example 8-42 Bringing up a dynamic tunnel between TCPIPE and TCPIPC

```
CS02 @ SC33:/u/cs02>ping -p TCPIPE 10.1.1.30
CS: Pinging host 10.1.1.30
sendto(): EDC5111I Permission denied. (errno2=0x74420291)1
CS02 @ SC33:/u/cs02>ping -p TCPIPE 10.1.1.30
CS: Pinging host 10.1.1.30
Ping #1 response took 0.000 seconds. 2
CS02 @ SC33:/u/cs02>
```

Note that in Example 8-42 on page 321, the first ping (1) does not succeed.

Why the connection fails at the first ping command: Our IPSec policy was specified with “OnDemand.” ICMP commands (such as **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 turn on 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 might need the tunnel available).

The second **ping** (2) does succeed, which indicates that we have established a connection across the VPN.

8.7.5 Diagnosing IKE with RSA signature mode

The following two resources are the most relevant for diagnosing problems in an IKE implementation:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781

The information in these two resources 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 that might occur. The most important messages for diagnosing problems are:

- IKE daemon messages
- TRMD messages
- PAGENT messages

Your syslog daemon implementation can place the pertinent messages in separate logs or in one large log. However, many of the messages that you might need for advanced diagnosis are only available if you raise the logging levels.

Tip: 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 can 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.

8.8 Additional information

For more information, see the following resources:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding IPSec configuration
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787, for additional information about authorization

9

Network Security Services for IPSec clients

In this chapter we explain how you can use Network Security Services (NSS) by implementing a Network Security Services Daemon (NSSD) and Internet Key Exchange Daemon (IKED) as a Network Security Services client.

We discuss the following topics in this chapter.

Section	Topic
9.1, “Basic concepts” on page 326	Basic concepts behind an NSS implementation
9.2, “Configuring NSS for the IPSec discipline” on page 338	The preliminary and integral steps needed to configure the NSS environment
9.3, “Verifying the NSS environment for the IKED Client” on page 404	The commands to verify the NSS working environment
9.4, “Diagnosing the NSSD environment” on page 423	The references, commands, and log entries that are helpful in diagnosing NSS problems
9.5, “Worksheet questions for NSSD implementation (IKED Client)” on page 427	Worksheet to help design the NSS environment on your installation, to simplify configuration of the implementation
9.6, “Additional information” on page 428	References to additional information that are helpful in implementing NSS

9.1 Basic concepts

To understand Network Security Services (NSS), you must first understand the disciplines that NSS can support:

- The IPSec discipline
- The XMLAppliance discipline

In z/OS V1R9, NSS was introduced to centralize the certificate and key management for IKED clients. In z/OS V1R10, the role of NSS was expanded to include function for IBM DataPower® appliances as NSS clients.

This chapter discusses only the IKED client as part of the IPSec discipline. It assumes that you are familiar with and know how to implement IPSec and Internet Key Exchange (IKE) using digital signature mode as explained in Chapter 8, “IP Security” on page 245.

9.1.1 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 over a Virtual Private Network (VPN). It is protocol-independent, allowing security to be applied to TCP packets, and 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:

- *Manual tunnels* are built when security parameters and encryption keys are configured statically at each end of the tunnel. The tunnels are managed manually by a security administrator. Keys are refreshed when the tunnels are recycled manually.
- *Dynamic tunnels* use the IKE protocol to negotiate security parameters and to generate encryption keys dynamically to lower the risk of compromise on the secret key. The keys are refreshed dynamically during the lifetime of the tunnel.

With dynamic tunnels the IKE protocol can work with *pre-shared keys* or with *digital signature mode* during phase I.

With pre-shared keys, IKE authenticates the peers by means of a secret key that is shared between the IKE peers, which is called *pre-shared key mode*. With *digital signature mode*, the IKE protocol uses X.509 certificates to authenticate the IKE peers. Therefore, in digital 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 (CA) certificate to authenticate the peer.

Figure 9-1 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*.

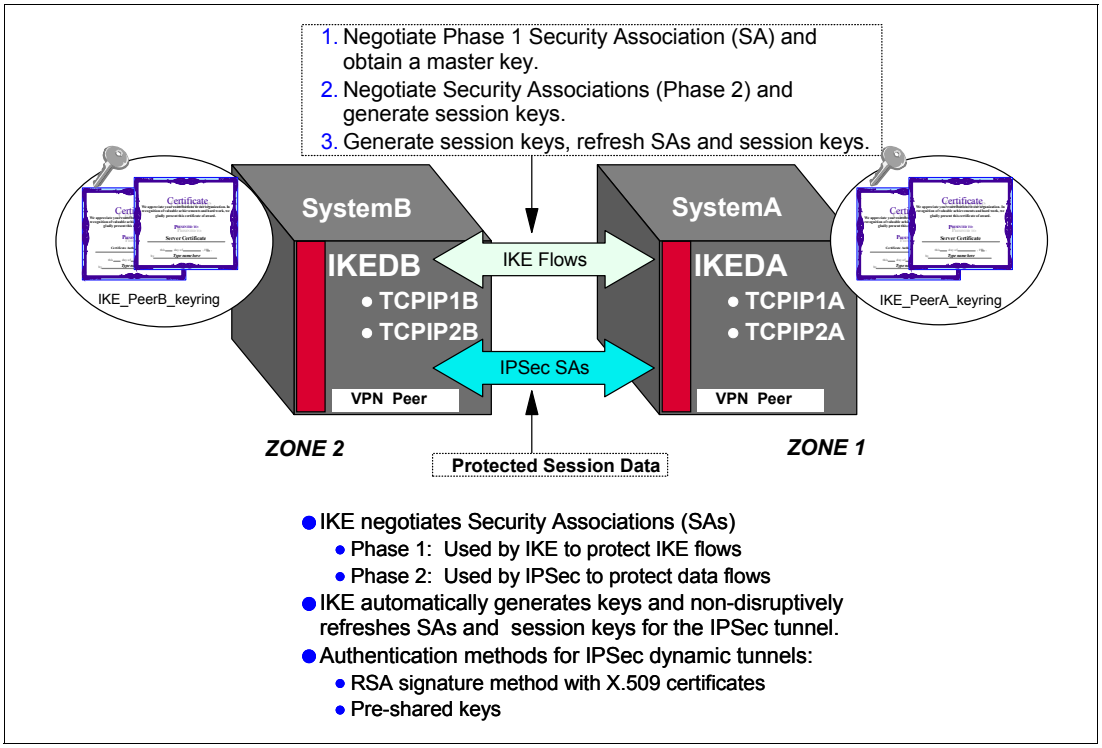


Figure 9-1 IKE concepts

Note that NSS plays a role in digital signature mode authentication only on behalf of an IKED client. NSS is not concerned at all with pre-shared key mode because digital 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, can reside in a separate zone of the IP network. Some zones are more trusted than others. For example, in Figure 9-1, Zone 1 can be within a highly secure enterprise data center. Zone 2 can 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 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 refresh again requires RSA signature operations.

Managing certificates and signatures in this environment can become onerous if the network is large. The administration of certificates across many security endpoints is cumbersome and error-prone, and the configuration and deployment of the many certificates is subject to the same burdens. Also, they are the security vulnerabilities themselves. Sensitive data, such as private keys, can be compromised when stored in less trusted zones.

The NSS solution can help to overcome these challenges.

9.1.2 The NSS solution for IKED Clients: IPSec discipline

The NSS server supports the *IPSec discipline*, which provides a set of network security services for IPSec. These services 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 9-2 by the Internet Key Exchange Daemon (IKED) in SystemB. IKED is enhanced with NSS client functionality on SystemA.

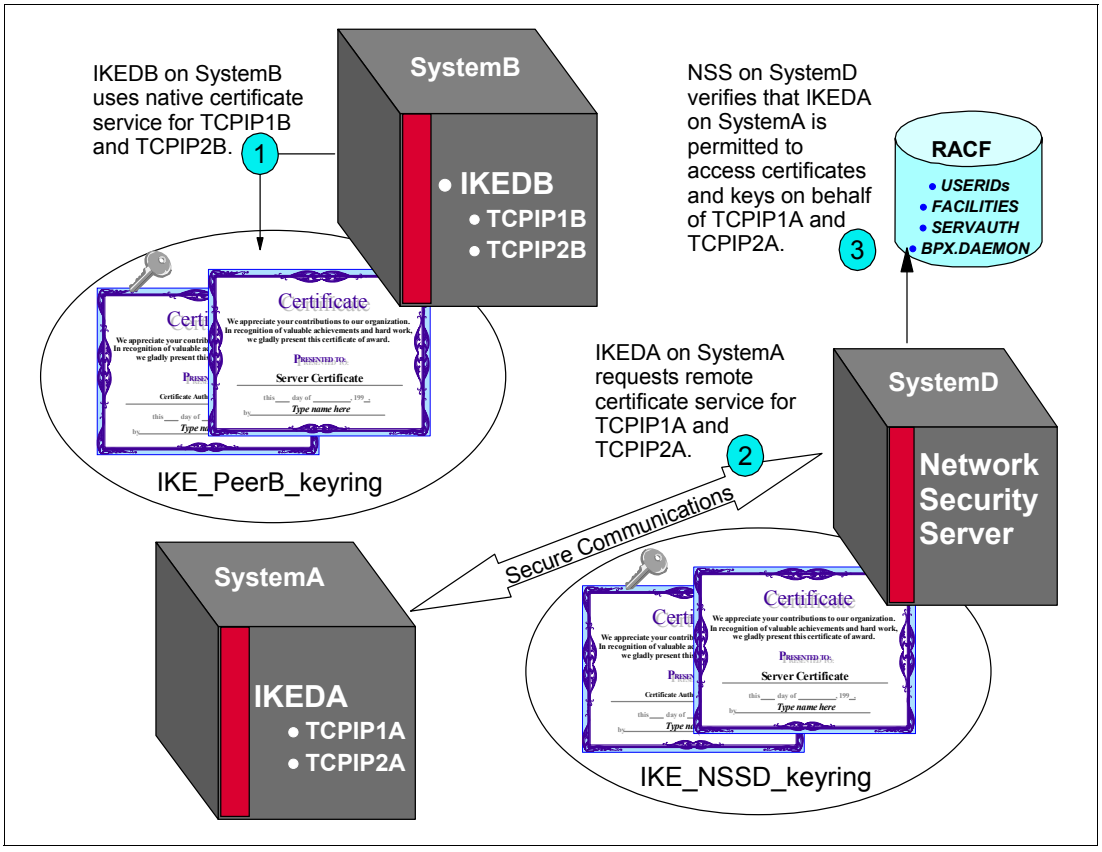


Figure 9-2 Overview of Network Security Services: Certificate management

In Figure 9-2, IKEDB on SystemB at (1) uses the X.509 certificates and private keys that are stored natively on the local key ring (IKE_PeerB_keyring). Alternatively, 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 certificates and private keys that are stored on the NSS server's key ring (IKE_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 SSL/TLS.

Tip: 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 SystemA that is not NSS-enabled (TCPIP3A is not depicted in Figure 9-2). In such a case, TCPIP3A still needs access to a local key ring. This type of configuration might 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 IKE_NSSD_keyring and its contents on behalf of TCPIP1A and TCPIP2A. After that is accomplished, the NSS creates and verifies digital signatures on behalf of the NSS client using private keys and digital certificates that are stored only at the NSS server on the remote ring (IKE_NSSD_keyring). Thus, although the NSSD_keyring is owned by NSS, it contains the same certificates that formerly populated the IKE_PeerA_keyring to which we referred in Figure 9-1 on page 327.

Figure 9-3 illustrates several of these points.

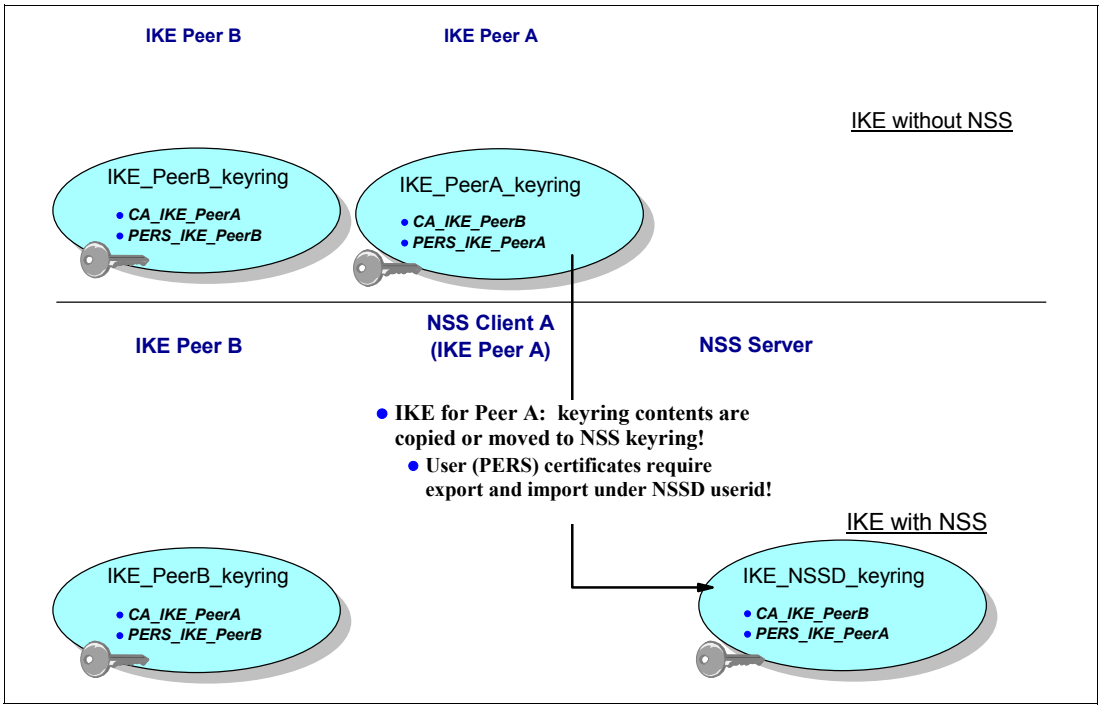


Figure 9-3 Contrasting key ring usage: IKE without NSS and IKE with NSS

Because the NSS key ring is functionally equivalent to the IKE key ring, you see in Figure 9-3 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.

Tip: 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 definitions in 9.2.6, "Configuring authorizations for NSS" on page 351, when we describe the implementation steps for NSS.

NSS also provides network management services for NSS clients as depicted in Figure 9-4.

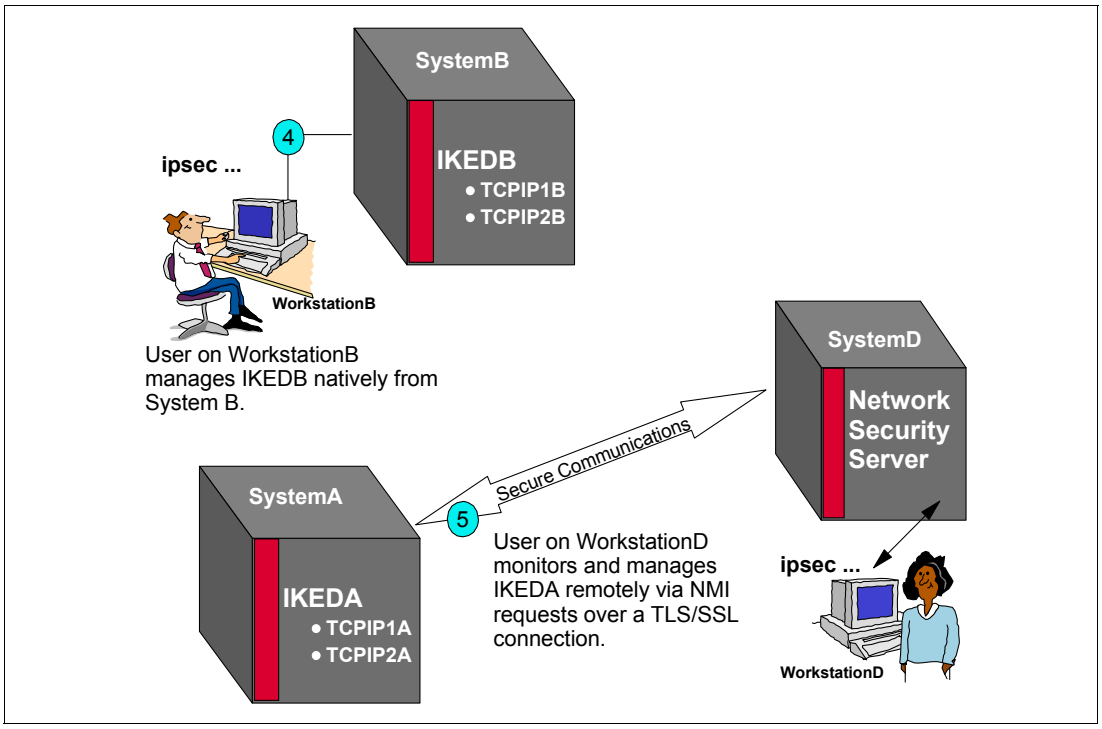


Figure 9-4 Overview of Network Security Services: Network Management Service

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 can use NSS remote management services to monitor and control remote IPsec endpoints. The user on Workstation D can 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 9-4 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 because of the connection between the NSS daemon (NSSD) and the NSS client, IKEDA on SystemA. These communications take place across a connection secured with SSL/TLS.

The IKE daemon can be configured to act as an NSS client on behalf of multiple TCP/IP stacks. In Example 9-52 on page 416, 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 with the IPSec Discipline

NSS centralizes the sensitive key ring material that might 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 digital certificates along with signature services within the most trusted zones.

Other advantages inherent in the NSS solution allow 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 9-5 expands on the view of the NSS solution shown in earlier figures.

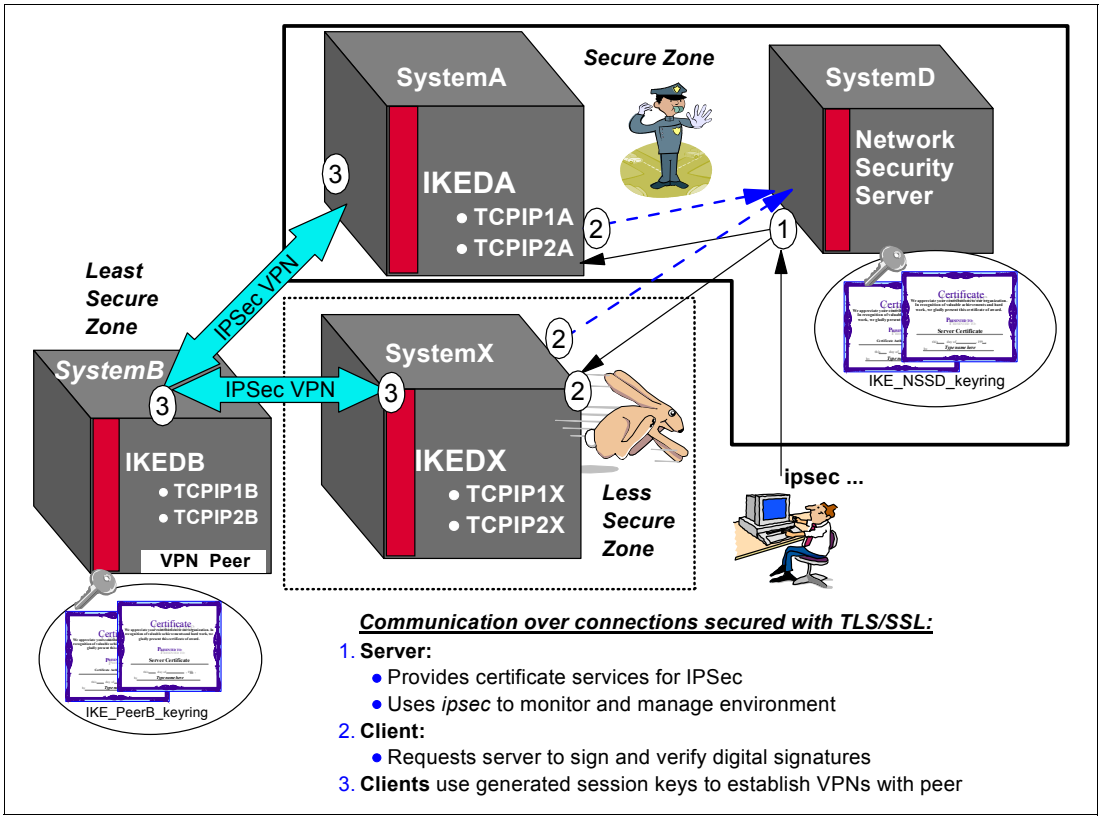


Figure 9-5 Benefits of NSS solution

Figure 9-5 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 eight TCP/IP stacks running on that system. Communication between client and server systems is secured by SSL/TLS connections.

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

Also note that multiple IKE daemons can access network security services simultaneously. These IKE daemons do not have to reside within the same sysplex as the NSS server. In Figure 9-5 on page 331, we have two 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 the system SystemX. This is possibly a test sysplex in the data center.

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

Key rings for the NSS implementation

As shown in Figure 9-2 on page 328, IKE daemons and the NSS server require key rings to hold the private keys and certificates for digital authentication during IKE negotiations.

Tip: The NSSD key ring contains the certificates that normally reside on the IKE key ring. The NSSD key ring takes the place of the IKE key ring for any NSS clients.

Also shown in Figure 9-2 on page 328 and Figure 9-4 on page 330, the communications between the NSS clients and the NSS server are secured by SSL/TLS protocols. Thus, the NSS solution also requires SSL/TLS policies that point to key rings that hold SSL/TLS certificates and private keys.

Figure 9-6 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 key ring 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 separate set of key rings (A_R and A_L).

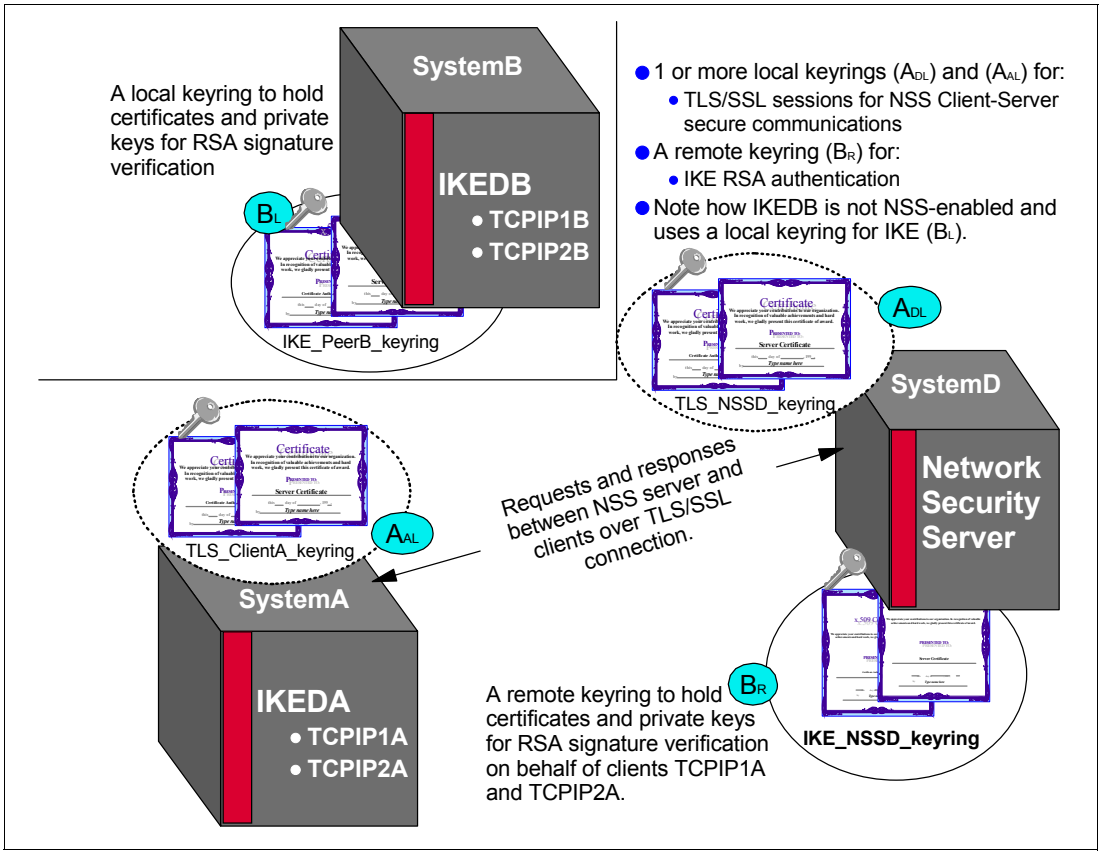


Figure 9-6 NSS using key rings for IKE processes and secure client/server communication

As discussed in Chapter 3, “Certificate management in z/OS” on page 39, you can store certificates on key rings in many different ways. So, here we look at strategies for managing key rings, as illustrated in Figure 9-7.

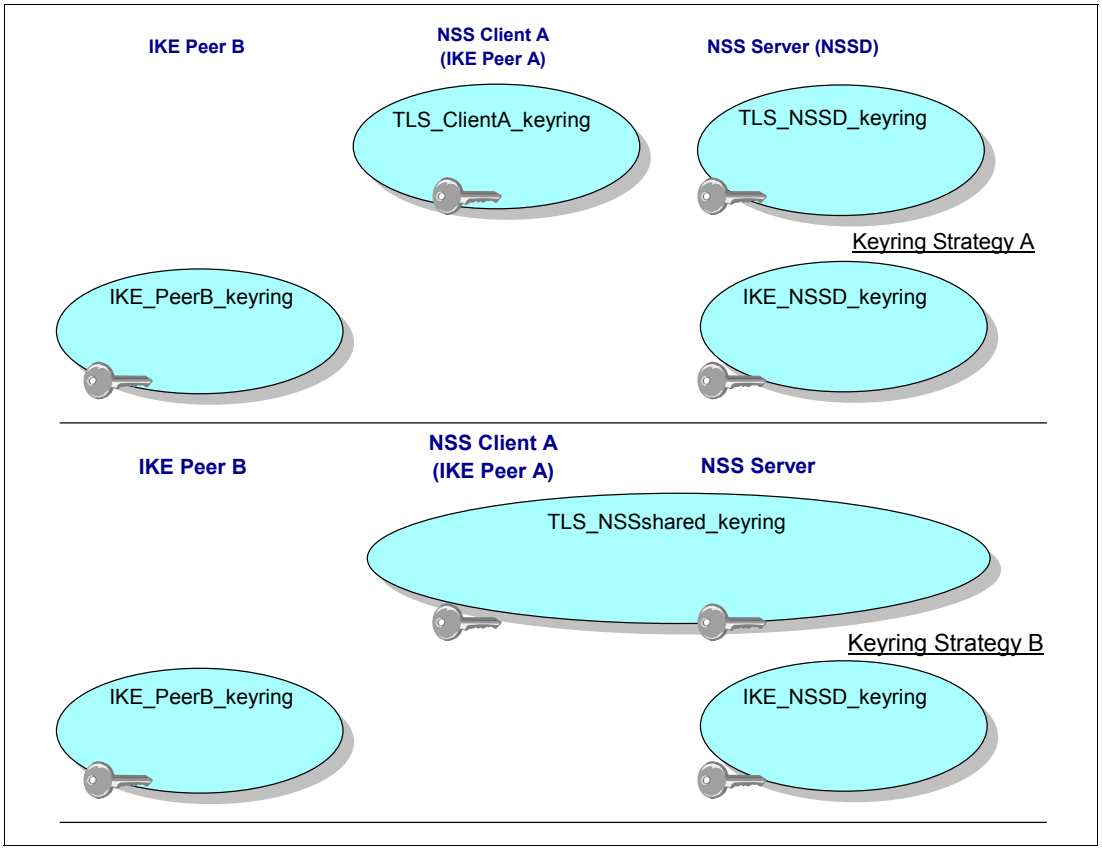


Figure 9-7 Key ring strategies for an NSS implementation

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

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

- Key ring Strategy A, shown in Figure 9-7, consists of the following strategies:
  - A key ring solely for SSL/TLS at the NSS server side (TLS_NSSD_keyring).
  - A separate key ring for SSL/TLS 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.

- Key ring Strategy B, which is also shown in Figure 9-7 on page 334, consists of the following strategies:
  - 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 SSL/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.

  Tip: In our digital 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.

- Figure 9-8 shows the contents of the key rings illustrated in Key ring Strategy A.

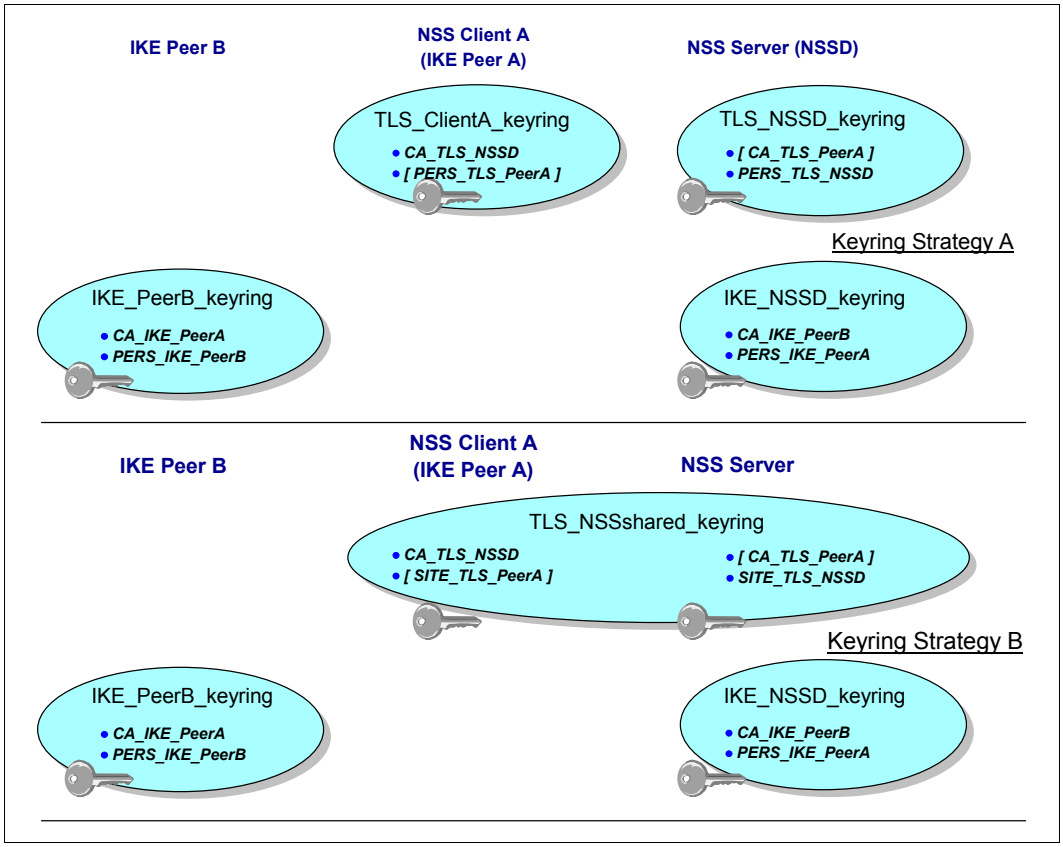


Figure 9-8 Key ring contents for an NSS implementation

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 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 might optionally see the CA certificate, CA_TLS_PeerA, that has signed the TLS client's personal certificate if client authentication is necessary.

  Tip: 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 can 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 CA certificate, CA_TLS_NSSD, that has signed NSSD's server certificate, which is PERS_TLS_NSSD. You might 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 might 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.

- In Figure 9-8 on page 335, 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 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 might optionally see the client's SITE certificate, SITE_TLS_PeerA, and the CA Certificate, CA_TLS_PeerA, that has signed the TLS client's SITE certificate if client authentication is necessary.

    Tip: 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 might 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.

General Rule: At the end of this chapter, we discuss how to implement the complete secured environment for both digital signature verification and for TLS. We assume that 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 general rule for the implementation of NSSD.

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 SSL/TLS greatly simplifies the implementation of NSSD, because your only concern at that point is to create the NSS client and NSS server relationship.

Therefore, reduce your learning curve with NSSD by migrating an existing IPSec and IKE scenario to one using NSSD. You should also already have experience with the Configuration Assistant for building the AT-TLS environment. This is the approach we took in our NSSD scenario.

9.2 Configuring NSS for the IPSec discipline

The network topology that we use to configure our NSS scenario is based on the digital signature mode scenario that we built in 8.7, “Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1” on page 306. Figure 9-9 shows again the network diagram that illustrates a successful IPSec and IKE implementation.

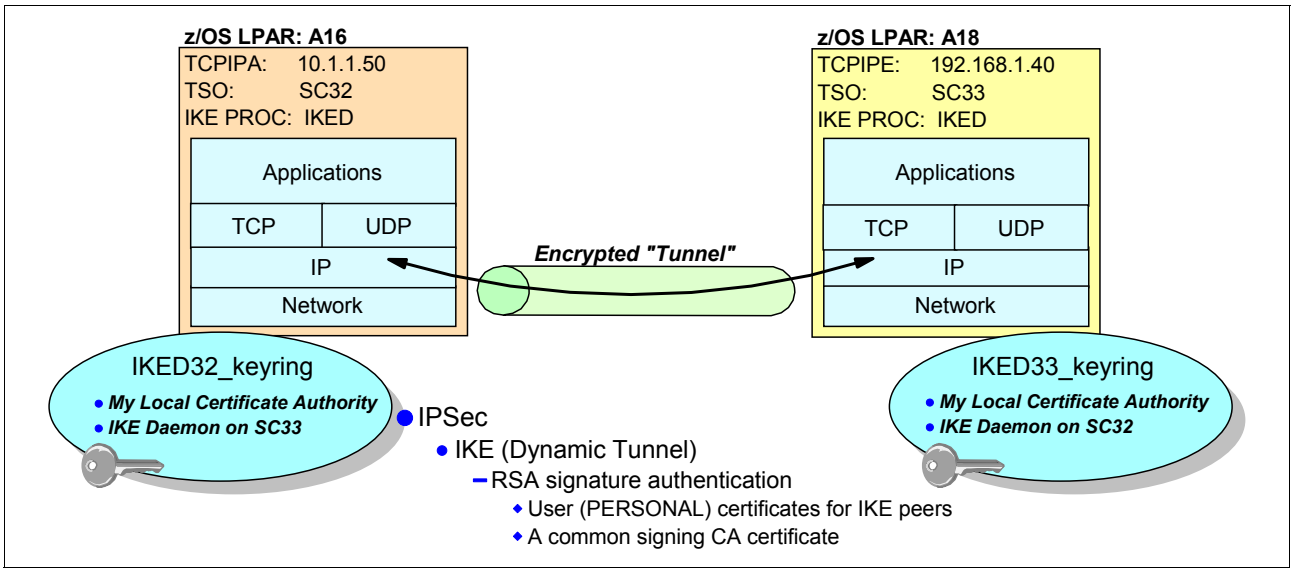


Figure 9-9 VPN tunnel between TCPIPA on SC32 and TCPIPE on SC33

Using IKE with digital 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 NSS on behalf of SC32’s TCPIPA and the local certificate services of SC33’s TCPIPE.

Tip: 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.

9.2.1 Overview of preliminary tasks

Before you begin the configuration, complete these tasks:

1. Select or build an IPSec environment that is using IKE in digital signature mode so that you can migrate one of the IKE peers to be an NSS client. We built this environment in 8.7, “Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1” on page 306.
   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 use.

   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 about PAGENT and AT-TLS.
3. 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/syslog.conf` file:

     ```
     *.IKED*.*.*     /var/syslog/%Y/%m/%d/iked.log -F 640 -D 770
     ```

     Note that IKED logging can be accomplished with another variation of the definition:

     ```
     local4.*        /var/syslog/%Y/%m/%d/local4.log -F 640 -D 770
     ```

   – NSSD logging

     At the NSS server node, add a statement similar to the following statement to the `/etc/syslog.conf` file to capture NSSD messages:

     ```
     *.NSSD*.*.*     /var/syslog/%Y/%m/%d/nssd.log -F 640 -D 770
     ```

     However, many NSSD messages are also captured in the `local4.*` facility name and priority. Therefore, at the server add:

     ```
     local4.*        /var/syslog/%Y/%m/%d/local4.log -F 640 -D 770
     ```

   – TRMD logging

     At all the IKED nodes on the mainframe, ensure that the TRMD messages flow to a log file. They can flow to the IKED log or elsewhere, as you choose. For example, you might implement the following line:

     ```
     *.TRMD*.*.*     /var/syslog/%Y/%m/%d/trmd.log -F 640 -D 770
     ```
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. See Figure 9-10 on page 340 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 9.5, “Worksheet questions for NSSD implementation (IKED Client)” on page 427. Also consult our worksheet, which has already been filled out for our implementation.

9.2.2 NSS client and NSS server

With this scenario, an NSS client (IKED) on SC32 uses the services of NSSD on SC33. The NSS client retrieves certificates and keys from the NSS server to establish an IPSec tunnel between TCPIPA on SC32 and TCPIPE on SC33. See Figure 9-10.

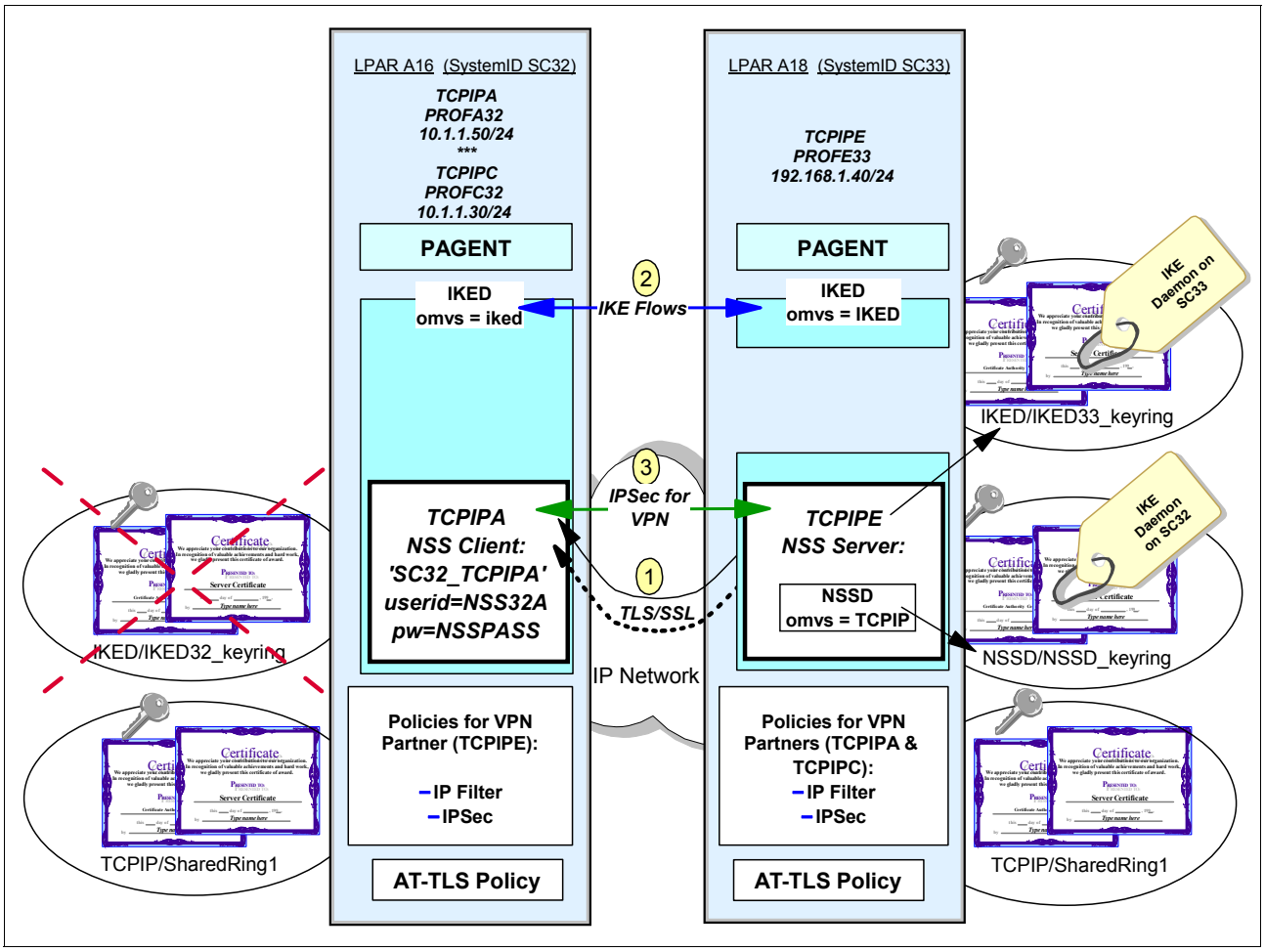


Figure 9-10 TCPIPA is NSS-enabled

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 digital 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 SSL/TLS 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 digital signature verification.

At (2), you see the flows between IKED (SC32) and IKED (SC33) to manage the IPSec session keys. IKED (SC33) natively accesses a key ring named IKED/IKED33_keyring that contains IKE certificates and private keys. The NSS server accesses its key ring, NSSD/NSSD_keyring, when it provides NSS Certificate Management services to IKED on behalf of TCPIPA on SC32. IKED (SC32) 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 you can see, TCPIPA on SC32 in Figure 9-10 on page 340 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 the `iked.conf` file
- The client user ID, defined as a RACF user ID at the NSS server node

Therefore, in Figure 9-10 on page 340, 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, IKED on SC32)
- A shared SSL/TLS 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 IKED 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 you should already have experience in IPSec, IKE, and AT-TLS prior to working with NSS so that you can focus entirely on NSS itself.

9.2.3 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. We provide a blank worksheet for you to complete in 9.5, “Worksheet questions for NSSD implementation (IKED Client)” on page 427. Here, we provide a sample worksheet that we completed for our implementation as illustrated in Figure 9-10 on page 340:

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon?
   - SC33
2. What is the superuser user ID associated with the OMVS segment under which the NSS daemon is to run?
   - TCPIP
3. With which TCP/IP stack (or stacks) does the NSS daemon establish affinity?
   - TCPIPE
4. What is the address or the DNS name that the NSS client will use to connect to the NSS server?
   - 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?
   - 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?
   - 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?
   - NSS32A
8. What type of authentication should our client user ID present: a password or a passticket?
   - password of NSSPASS
9. What is the user ID associated with the OMVS Segment of the client’s IKED procedure?
   - IKED
10. What is the name of the key ring 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 digital certificate for each TCP/IP stack? Display with **`racdcert listring(IKED_keyring) id(IKED)`**.
    - 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?)
    - NSSD
13. What is the name of the key ring on the NSS node that is used for the SSL/TLS secure connection between the NSS server and client?
    - TCPIP/SharedRing1
14. What is the name of the key ring on the NSS client node that is used for the SSL/TLS secure connection between the NSS client and server?
    - TCPIP/SharedRing1
15. What are the IP addresses of the NSS server and the NSS client for which you might have to establish IP filters on the NSS client node?
    - 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 might have to establish IP filters on the NSS server node?
    - 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 z/OSMF Configuration Assistant

The z/OSMF Configuration Assistant allows you to configure your NSS server (and clients) through a graphical user interface (GUI). It then generates the NSSD configuration file and the necessary AT-TLS policy and IP filter rules for NSS client-server traffic. Note that the z/OSMF Configuration Assistant also generates the NSS client configuration in the IKED configuration file.

Tip: The TCP/IP stack under which the NSS server executes is not required to have IKED running unless it is also to be the endpoint of an IPSec tunnel. In our scenario, we used the TCP/IP stack named TCPIPE as an IPSec endpoint. TCPIPE also happens to be the stack with which the NSS server establishes affinity.

9.2.4 Configuring the NSS environment

As we have described, you need to complete many tasks prior to defining the NSS server and client environment. Figure 9-11 illustrates many of the prerequisite components.

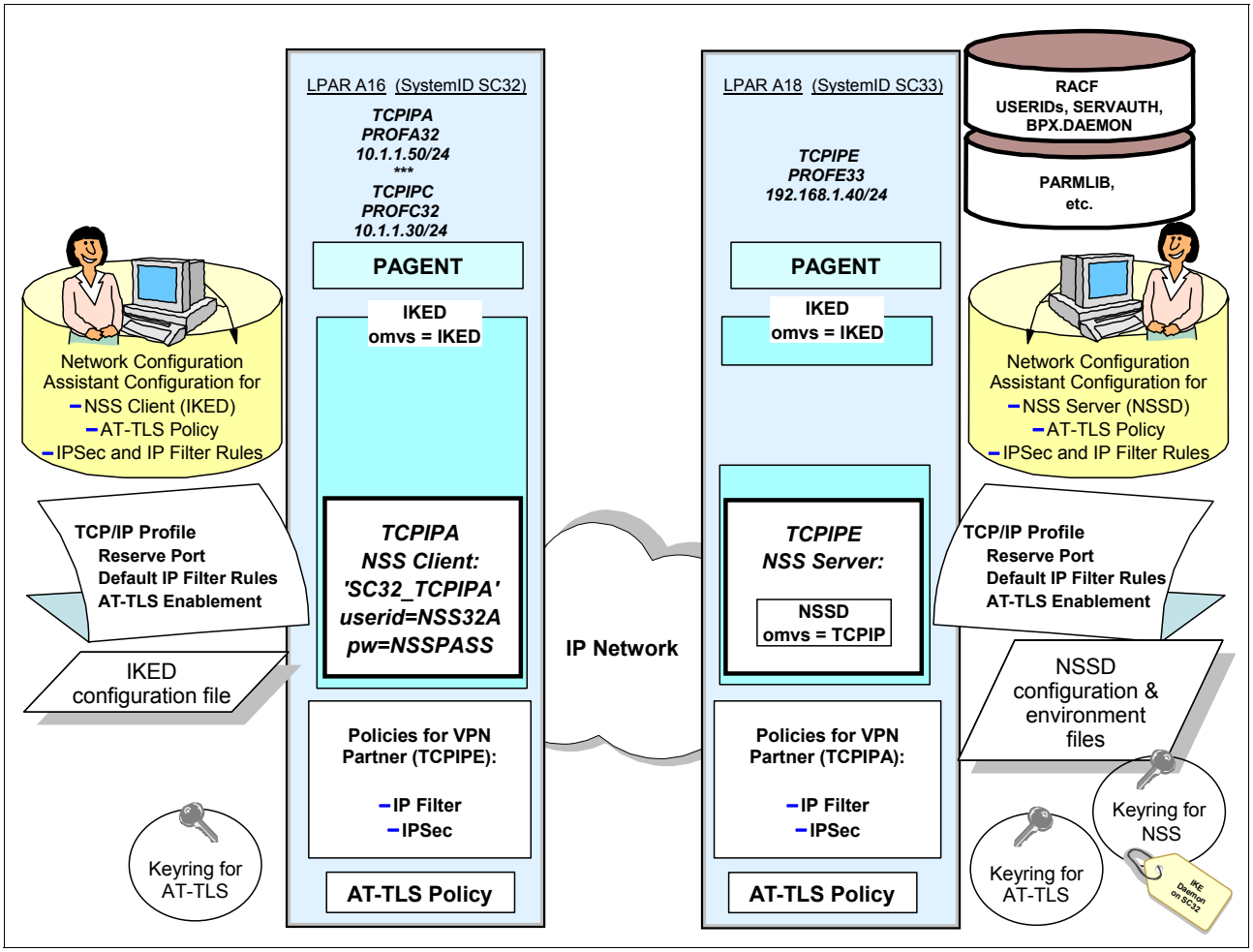


Figure 9-11 Components of an NSS configuration for our scenario

The NSS server or daemon (NSSD) prerequisites

The base environment for NSS has several prerequisites at the server side:

- 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.
- 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.
- A Network Security Services Daemon (NSSD) with an NSS daemon configuration file that has been generated either manually or with the Network IBM Configuration Assistant.
- A small set of environment variables that define the location configuration resources.
- 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.
- A RACF key ring that contains all of the certificates and private keys that will be used for digital signature creation and signature verification on behalf of any of the NSS clients during IKE processing.

  A certificate authority certificate for IPSec and IKE that can be used to authenticate the VPN partners of the NSS client is also required.

  Tip: You can use a single key ring for the certificates required for both AT-TLS and IKE. Alternatively, you can use more than one key ring. Consult Chapter 3, "Certificate management in z/OS" on page 39 for more information about this subject. Also see "Key rings for the NSS implementation" on page 332 for a discussion of several strategies that you can use to manage key rings.

- RACF SERVAUTH profiles to control client access to NSS services, including both remote management and certificate services.
- 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:

- 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.

- 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.
- 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 5, “Central Policy Server” on page 161
  - Chapter 7, “IP filtering” on page 217
  - Chapter 8, “IP Security” on page 245
- 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.

  Tip: The digital 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.

- A small set of environment variables that define the location configuration resources.

9.2.5 Configuring prerequisites for NSS for an IKED Client

As already noted, the easiest way to build an NSS environment is to have the prerequisite technologies, such as SSL/TLS and IPSec, already functioning so that working certificates are in place and that IPSec is already implemented successfully. With these technologies firmly in place, it is then easy to set up the NSSD environment.

General rule: Implement NSSD after you are comfortable with the concepts of certificates, AT-TLS, IPSec, and the use of the Network Configuration Assistant. Even better, implement NSSD after you have worked in these environments so that you can minimize risks and reduce the learning curve, because all you need to do is implement the NSS daemon and its client.

For this scenario, we assume that you are an expert in creating key rings and certificates. and that you are familiar with the Network Configuration Assistant. You can see previous chapters for a review for many of the steps in the following sections.

Creating and managing certificates

You must have the appropriate RACF key ring environment with the necessary CA Certificates and server certificates available to implement AT-TLS and NSS. You need to prepare these key rings ahead of time. For information about creating the various types of certificates, consult Chapter 3, “Certificate management in z/OS” on page 39.

For working examples of how these certificates were used for AT-TLS with TN3270 and FTP, consult the following chapters:

- Chapter 12, “Application Transparent Transport Layer Security” on page 551
- Chapter 16, “Telnet security” on page 677
- Chapter 17, “Secure File Transfer Protocol” on page 719

For working examples of how these certificates and key rings were used for IPSec and with IKED, consult Chapter 8, “IP Security” on page 245.

Tip: In the examples provided, we have assumed that all the necessary RACF certificate classes are defined and activated. If not, then consult the chapters listed before this note, 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

Building the SSL/TLS 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 9-1. This is the self-signed CA certificate with which we sign our SITE certificate for the NSS daemon.

Example 9-1 CA 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(2008-11-10))                    -
    NOTAFTER(DATE(2012-11-11))                     -
    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 used for several servers, such as FTP and TN3270, on behalf of SSL/TLS. We decided to use the same SITE certificate to represent the NSS server during the SSL/TLS flows.

Example 9-2 shows the JCL that we used to create the NSS server certificate.

Example 9-2 SITE certificate for SSL/TLS 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(2008-11-11))                        -
     NOTAFTER(DATE(2012-11-11))                         -
     withlabel('CS19 ITSO SharedSite1')                -
     signwith(certauth label('CS19 ITSO CA1')
  racdcert site list(label('CS19 ITSO SharedSite1'))
/*
```

Example 9-3 shows the JCL that we used to create the key ring for AT-TLS and to attach the certificates to that key ring.

Example 9-3 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 user ID '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)
/*
```

Enabling the TCP/IP stack for AT-TLS

Consult Chapter 12, “Application Transparent Transport Layer Security” on page 551 for information about enabling AT-TLS in the TCP/IP profile for both the NSS client (IKED) on SC32 and the NSS server (NSSD) on SC33.

We perform this step later in “Update the TCP/IP profile” on page 364.

Building the AT-TLS policy for NSS server and client

We use the Network Configuration Assistant later (in “Update the policy files at the NSS server and client nodes” on page 365) to build the AT-TLS policy for NSS server and client.

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 that key ring. However, you must execute a critical set of steps to ensure that the NSS server can process the NSS client certificates properly. In fact, the user (PERSONAL) certificates on the NSSD key ring must be owned by the user ID under which the NSSD server is executing. They *cannot* be associated with the old user ID associated with the IKED server.

You cannot reconnect the existing certificate to the new key ring. Instead, you must move certificates from the old IKE key ring to the new NSS key ring as follows:

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 be reconnected to the NSSD key ring.

IKED certificate definition for NSS client

We already built the IKED certificates for the NSS client when we created the IKE scenario with digital signature authentication in 8.7.1, “Generating certificates for IKEv1 RSA signature mode” on page 307. However, those certificates reside on a key ring named IKED/IKED32_keyring. We no longer need this key ring 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.

To create the key ring for the NSS server and connect the required certificates to NSSD's key ring, complete the following steps:

1. Review the JCL in Example 9-4, which shows how we exported the NSS client's certificate to a data set. Remember to name a password for this operation. It is used again later when you import the certificate.

Example 9-4 Export a certificate to a data set

```
//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 user ID 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('CS02.CERT.EXPORT32') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
/*
```

2. Examine the JCL that we used to delete the certificate from the RACF database, shown in Example 9-5.

Example 9-5 Delete the certificate under its previous owner from the RACF database

```
//CERTDE32  JOB MSGCLASS=X,NOTIFY=CS02
//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 user ID.                                       *
//*********************************************************************
//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 as shown in Example 9-6.

Example 9-6 Import the certificate and its key and 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 user ID 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 9-7.

Example 9-7 Create new key ring for NSSD and add the NSS Client certificates 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 user ID 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, add the CA Certificates that have signed the certificates of the NSS Client's VPN partners to the NSSD key ring as shown in Example 9-8.

Example 9-8 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 user ID 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)
/*
```

Enabling TCP/IP stack for IPSec

Ensure that the NSS client stacks are already implemented with IPSec and IKED. All the VPN peers to the NSS client stacks also must be enabled for IPSec. Consult Chapter 8, "IP Security" on page 245 for more information about these topics.

9.2.6 Configuring authorizations for NSS

We used RACF as our external security manager for NSS. To authorize resources to NSS, follow these steps:

1. Create a user ID for the NSSD started task, and associate it with a required UID of *0*. Example 9-9 shows the RACF commands that we executed from TSO to create this user ID. See also *z/OS Communications Server: IP Configuration Guide*, SC31-8775, which suggests a user ID of *NSSD*.

Example 9-9 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
SETROPTS GENERIC(STARTED) REFRESH
```

2. Permit the NSSD user ID to SYS1.PARMLIB:

```
PERMIT SYS1.PARMLIB ID(NSSD) ACCESS(READ)
```

3. Define 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. Permit the administrator user IDs that will administer key ring access to the following facility classes. The users CS01 through CS06 belong to the RACF group SYS1.

```
PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(SYS1) ACC(UPDATE)
PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(SYS1) ACC(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

5. Permit the NSSD user ID to have READ access to the key rings that it owns and to the key rings that it does not own:

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(NSSD) ACC(CONTROL)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(NSSD) ACC(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

Tip: In our scenarios, NSSD owns the NSSD key ring for the IPSec (IKED) clients. For this key ring, NSSD requires only READ access. However, our scenarios also require access to the AT-TLS key rings, which are owned by the user ID of TCPIP. Therefore, the NSSD user ID requires CONTROL and UPDATE access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING facility classes respectively, as shown in our examples.

6. Permit the NSSD user ID to the BPX.DAEMON facility class.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(NSSD) ACCESS(READ)
setropts raclist(facility) refresh
```

7. If you have chosen to implement secured signon for the NSS client's user ID with passtickets, follow the instructions for passticket authorization as described in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. In our scenario, we elected to use a simple password authentication for our NSS client user IDs and, therefore, skipped this step.
8. If you have chosen to implement RACF certificate name filtering for an NSS XML client, follow the instructions for certificate name filtering as described in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. In our scenario, we chose not to map x.500 distinguished names to a RACF ID and, therefore, skipped this step.

Tip: The next implementation steps require that you authorize SERVAUTH access to the NSS client. Each NSS discipline and its service requires a different set of SERVAUTH authorizations:

- IPSec Certificate Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.IPSEC.CERT`

- IPSec Network Management Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.IPSEC.NETMGMT`

- XML Appliance SAF Access Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.XMLAPPLIANCE.SAFACCESS`

The NSS IPSec (IKED) client

To authorize an NSS IPSEC (IKED) client, complete the following steps:

1. Create a user ID for the NSS client. In our scenario, we decided named the IPSec client *NSS32A*. The client's password is *NSSPASS*. Example 9-10 shows the JCL that we used to create this user ID.

Example 9-10 IPSec (IKED) client user ID and password generation

```
//ADDNSSx 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
//* requires 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   NSSXML GROUP(SYS1) OWNER(HAIMO) AUTHORITY(JOIN) +
            UACC(ALTER)
  ALU NSSXML 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 NSSXML -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
//*
```

2. Permit this user ID to a new class that is created with the commands shown in Example 9-11. This class is used for the NSS Certificate Services.

Example 9-11 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

Design and diagram this scenario carefully and assign meaningful naming conventions to it.

You need a class for each client. Alternatively, you can wildcards, but you lose some granularity and control when using wildcards.

3. NSS has a second capability beyond Certificate Services called *IPSec management*. Therefore, next permit the NSS client user ID and the user ID of the NSS administrator to a new SAF class that is created with the commands shown in Example 9-12. This class is used for the NSS Network Management Services.

Example 9-12 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 syntax for Certificate Services. However, there is an extra IPSEC segment. The syntax of this SERVAUTH class is:

EZB.NSS.*<NSSD MVS System ID>*.*<NSS Client Symbolic Name>*.**IPSEC**.NETMGMT

4. Define an unique SERVAUTH class resource profile for each NSS client. One of the fields in it is composed 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.

   You need to verify the label on that certificate, so use a RACF command to list the contents of the IKE key ring that contains the certificate assigned to the TCPIPA NSS client as shown in Example 9-13.

Example 9-13 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, you can verify that the label assigned to the client certificate is *IKE Daemon on SC32*.

Now, you can define the new class resource with one of the following formats:

- `EZB.NSSCERT.<NSSD MVS System ID>.<NSS Client IKE Certificate Label>.HOST`
- `EZB.NSSCERT.<NSSD MVS System ID>.<mappedlabelname>.HOST`

Example 9-14 shows our definition.

Example 9-14 SERVAUTH Resource profile for NSS client's CA 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 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 following characters are affected by this rule:

- Asterisk (*)
- Percent sign (%)
- Ampersand (&)
- Blank

Tip: We executed the RDEFINE and the PERMIT statements from the TSO ISPF environment. As a result, even if we entered the label in lowercase, our definitions were successful because TSO converts the alphabetics to uppercase automatically.

For more information about mapped label names, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

5. Now, build another profile for the CA certificate that has signed the NSS client's certificate. We know from a previous display that this label is `My Local Certificate Authority`, as shown in Example 9-15.

Example 9-15 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, which represent the blanks in the certificate label of the CA certificate. The syntax is similar to previous syntax:

- EZB.NSSCERT.<*NSSD MVS System ID*>.<*NSS Client CA Certificate Label*>.CERTAUTH
- 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:

- Asterisk (*)
- Percent sign (%)
- Ampersand (&)
- Blank

To avoid the uppercase conversion requirement imposed by SERVAUTH definitions, always create RACF certificates in uppercase.

6. Define additional SERVAUTH resource to allow remote users to monitor (DISPLAY) or manage (CONTROL) NSS Clients, as shown in Example 9-16.

Example 9-16 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(CSO1,
CSO2,CSO3,CSO4,CSO5,CSO6,CSO7,CSO9,CS10) ACC(READ)

PERMIT EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.DISPLAY CLASS(SERVAUTH) ID(CSO1,
CSO2,CSO3,CSO4,CSO5,CSO6,CSO7,CSO9,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
- EZB.NETMGMT.<*NSSD MVS System ID*>.<*NSS Client Symbolic Name*>.IPSEC.DISPLAY

7. In our scenario, we had a few remaining SERVAUTH resources to define. If your user, CS02, wants to execute the **ipsec** command with the **-x** option, the user can display information about the NSS server and its IPSec clients. Likewise, if your user wants to display information about any type of client (IPSec or XML) supported by the NSS server, that use requires authorization to the **nssctl** command. Therefore, on the server system, you need the resource profile and permissions for CS02 shown in Example 9-17.

Example 9-17 SERVAUTH profile to display NSS information with nssctl or ipsec -x

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.NSS.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.NSS.DISPLAY CLASS(SERVAUTH) ID(CS01,CS02,CS03,
CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

This syntax in Example 9-17 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.

Important: There are two display commands for NSS management:

- The **ipsec** command with its various options
- The **nssctl** command

Authorization for both commands is provided through the resource profile

```
EZB.NETMGMT.<NSS Server SYSID>.<NSS Server SYSID>.NSS.DISPLAY
```

Although both commands are used for the NSS IPSec (IKED) discipline, the XML discipline takes advantage of only the **nssctl** command.

If user CS02 wants to display information about the IKE daemon at the NSS server node with the **ipsec -w display** command, the resource profile shown in Example 9-18 must be defined.

Example 9-18 SERVAUTH profile to display IKE at server with ipsec -w display

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.IKED.DISPLAY CLASS(SERVAUTH) ID(CS01,CS02,CS03,
CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Tip: 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.

8. If user CS02 wants to display information about the IKE daemon's NSS clients, then the resource profile shown in Example 9-19 must be defined at the NSS client node. The command for this type of display is again **ipsec -w**.

Example 9-19 SERVAUTH profile to display IKE at NSS client with ipsec -w

```
rdefine SERVAUTH EZB.NETMGMT.SC32.SC32.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC32.SC32.IKED.DISPLAY CLASS(SERVAUTH) ID(CS01,CS02,CS03,
CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Tip: Although the numerous authorizations that we describe here might seem quite tedious, we show them in detail to emphasize several critical points:

- The numerous individual SERVAUTH profiles and authorizations provide a great deal of control and security. To understand them well, it is necessary to provide this level of detail. However, after you understand the concepts, you can use wildcards in many of the fields to reduce the amount of definition that is required.
- The definitions require detailed planning. As mentioned in 9.2.1, "Overview of preliminary tasks" on page 339, two of the planning steps are to prepare a logical network diagram and to complete a worksheet that contains the necessary information for these definitions. This preliminary work is useful to expediting the steps that we cover in 9.2.6, "Configuring authorizations for NSS" on page 351.
- If you have already worked with IPSec and IP Filtering, the task of setting up NSSD for IPSec (IKED) clients becomes quite simple.

The NSS XML client RACF user ID

The authorization tasks in this section apply only to the NSS XML client:

1. Create a user ID for the NSS client. In our scenario, we named the XML client *NSSXML*. The client's password is to be *XMLPASS*. Example 9-20 shows the JCL that we used to create this user ID.

Example 9-20 XML Client user ID and password generation

```
//ADDXML 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 NSSXML PASSWORD(NEW2DAY) +
    NAME('ID for Comm Server') +
    OWNER(HAIMO) UACC(READ) DFLTGRP(SYS1) +
    AUTHORITY(JOIN) GRPACC +
    OMVS(AUTOUID HOME(/u/nssxml) PROGRAM(/bin/sh))
  CONNECT   NSSXML GROUP(SYS1) OWNER(HAIMO) AUTHORITY(JOIN) +
            UACC(ALTER)
  ALU NSSXML PASSWORD(XMLPASS) NOEXPIRED
```

```
  PASSWORD USER(NSSXML) NOINTERVAL
  ADDSD     'NSSXML.**' UACC(NONE) OWNER(NSSXML)
  SETROPTS REFRESH RACLIST(TSOPROC ACCTNUM TSOAUTH)
  DEFINE ALIAS   (NAME('NSSXML') RELATE('UCAT.COMCAT'))
/*
//DEFINEZ  EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(NSSXML.HFS) -
            LINEAR CYLINDERS(1 1) SHAREOPTIONS(3)    -
            VOLUMES(COMST4))
/*
//CREATEZ  EXEC   PGM=IOEAGFMT,
//           PARM=('-aggregate NSSXML.HFS -compat
//             -owner NSSXML -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/nssxml/.profile'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  PROF MSGID WTPMSG
  OCOPY INDD(PDS1) OUTDD(HFS1) TEXT
/*
//CHOWN1     EXEC PGM=BPXBATCH,
//   PARM='SH chown nssxml /u/nssxml/.profile'
//STDOUT     DD  PATH='/tmp/stdout',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//STDERR     DD  PATH='/tmp/stderr',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//*
```

2. Permit this user ID to a new class that is created with the commands shown in Example 9-21. This class is used for the NSS SAF Access Service.

Example 9-21 Authorizing XML client user IDs to SERVAUTH class for NSS

```
RDEFINE SERVAUTH EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.SAFACCESS UACC(NONE)

PERMIT EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.SAFACCESS CLASS(SERVAUTH) ID(NSSXML)
ACCESS(READ)

SETROPTS RACLIST(SERVAUTH) REFRESH

SETROPTS GENERIC(SERVAUTH) REFRESH
```

Note the following syntax for this class:

```
EZB.NSS.<NSSD MVS System ID>.<NSS Client Symbolic Name>.XMLAPPLIANCE.SAFACCESS
```

You need a class for each symbolic ClientID. Alternatively, you can use wildcards for the definition, but you lose some security granularity and control when using wildcards.

3. We had one remaining SERVAUTH resource to define for our NSS XML discipline scenario. If a user, CS02, wants to display information about the NSS server and the clients it is supporting, the user needs authorization to the **nssctl** command. Such authorization is provided through the resource profile and permissions for CS02's RACF group shown in Example 9-22.

Example 9-22 SERVAUTH profile to display NSS information with ipsec -x or nssctl -d

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.NSS.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.NSS.DISPLAY CLASS(SERVAUTH) ID(SYS1) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

This syntax in Example 9-22 is quite different from previous syntax:

```
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.

Important: There are two display commands for NSS management:

- The **ipsec** command with its various options
- The **nssctl** command

Authorization for both commands is provided through the resource profile

```
EZB.NETMGMT.<NSS Server SYSID>.<NSS Server SYSID>.NSS.DISPLAY
```

Although both commands are used for the NSS IPSec (IKED) discipline, the XML discipline takes advantage of only the **nssctl** command.

Tip: Although the numerous authorizations that we describe here might seem quite tedious, we show them in detail to emphasize several critical points:

- The numerous individual SERVAUTH profiles and authorizations provide a great deal of control and security. To understand them well, it is necessary to show this level of detail. However, after you understand the concepts, you can use wildcards in many of the fields to reduce the amount of definition that is required.
- The definitions require detailed planning. As mentioned in 9.2.1, "Overview of preliminary tasks" on page 339, two of the planning steps are to prepare a logical network diagram and to complete a worksheet that contains the information that you need for these definitions. This preliminary work is useful to expediting the steps that we cover in 9.2.6, "Configuring authorizations for NSS" on page 351.
- If you have already worked with IPSec and IP Filtering, the task of setting up NSSD for the IPSec discipline becomes quite simple.

9.2.7 Configuring the NSS server for an IKED Client

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, and TLS and IPSec certificates. This section explains these elements.

The NSS configuration file

The NSS server requires a configuration file, which you can create using either of these methods:

- Copy `nssd.conf` from `/usr/lpp/tcpip/samples` into `/etc` and then modify the copied file.
- Create the `nssd.conf` file with the Network Configuration Assistant.

Initially, for our scenario, we chose to create the 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 9-23 shows the pertinent parts of our NSS configuration file, which we stored in `/etc` as `nssd.sc33.conf`.

Example 9-23 The /etc/nssd.sc33.conf file 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 1
#
# Sys 0-255                   (dynamically modifiable)
# Default: 1
  SyslogLevel 255 2
#
# KeyRing userid/ringname (dynamically modifiable)
# This is the keyring holding the IKE certificates for IKED Server/Clients
  KeyRing NSSD/NSSD_keyring 3
#
# Discipline disciplineName Enable | Disable    (dynamically modifiable)
# Default: IPSec Enable
# Default: XMLAppliance Enable
  Discipline IPSec Enable 4
  Discipline XMLAppliance Enable 5
}
```

In this example, we specify the port to which the NSS server is to bind and on which it is to listen when it is initialized at 1. Port 4159 is the default port.

We specify a SYSLOG level of 255 at 2, which can help with problem determination. The default SYSLOG level is 4. We set the level quite high to ensure that we trap all possible messages.

We identify the NSSD key ring that holds the client certificates that are to be used for IKE digital signature processing, which is the NSSD/NSSD_keyring at 3. 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.

In certain versions, the instructions in the sample file and in *z/OS Communications Server: IP Configuration Guide*, SC31-8775, mistakenly 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. However, if the key ring was set up to be a *shared key ring* and it is populated with SITE certificates for the servers, then the owner of the key ring can be a USERID different from that under which NSSD or IKED is running.

At 4 and 5, we coded the default values for the disciplines that we support with this NSS server. Technically, we could disabled the XMLAppliance discipline for this section because we are illustrating only the IKE client here. However, we also support the XMLAppliance discipline in another scenario. Therefore, we decided to leave both discipline defaults in place.

Setting BPX_JOBNAME and BPX_USERID

We decided to start the NSS server with JCL, although you can choose to start it from `/etc/rc`. Therefore, in our case, we omitted the steps that we show in this section, which are documented in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Tip: If you are starting the NSS server from the UNIX shell or from within `/etc/rc`, you need to set the following environment variables:

```
# Start the NSS daemon
export NSSD_FILE=/etc/nssd.sc33.conf
export _BPX_JOBNAME='NSSD' /usr/lpp/tcpip/sbin/nssd -f /etc/nssd.sc33.conf &
export _BPX_USERID='NSSD'
unset _BPX_USERID
unset _BPX_JOBNAME
```

If you fail to set these environment variables, 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.

General Rule: Start NSSD as a procedure during or after TCP/IP startup because its prerequisite processes are not usually initialized prior to this time.

NSS server authorization to RACF

We performed the various NSS server authorization tasks earlier in 9.2.6, “Configuring authorizations for NSS” on page 351.

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 9.2.1, “Overview of preliminary tasks” on page 339.

NSS server environment variables

This is an optional step. You can specify the following variables in the environment variables:

- NSSD_CTRACE_MEMBER
- NSSD_FILE
- NSSD_PIDFILE
- NSSD_CODEPAGE

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 data set, as shown in Example 9-25 on page 364.

Establishing the NSS server’s IKE key ring: NSSD_keyring

We created this key ring earlier and populated it with the IKE client certificates.

Tip: 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

You can use several methods to initialize the NSS daemon:

- From an MVS procedure

  Example 9-24 shows the modified sample that we copied from `hlq.SEZAINST(NSSD)` into our MVS procedure library.

  Example 9-24 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"',    1
//         '"_CEE_ENVFILE=DD:STDENV")/')
//*
//* NSSD_FILE=/etc/security/nssd.conf
//* NSSD_CTRACE_MEMBER=CTINSS01
//*
//*STDENV DD DUMMY
//* Sample MVS data set containing environment variables:
//*STDENV DD DSN=TCPIP.NSSD.ENV(NSSD),DISP=SHR
//* Sample file containing environment variables:
//STDENV DD PATH='/etc/nssd.sc33.env',PATHOPTS=(ORDONLY) 2
//*
//* 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 the stack TCPIPE. Therefore, we established stack affinity to TCPIPE with the variable _BPXK_SETIBMOPT_TRANSPORT=TCPIPE as shown at **1** in Example 9-24 on page 363. Example 9-25 shows our environment file shown at **2** in Example 9-24 on page 363.

Example 9-25 Our environment file /etc/nssd.sc33.env for Network Security Services server

```
NSSD_FILE=/etc/nssd.sc33.conf
NSSD_CTRACE_MEMBER=CTINSS00
NSSD_CODEPAGE=IBM-1047
```

IBM Language Environment® variables: We executed our NSS server with PGM=NSSD. There are restrictions regarding Language Environment variables that might require you to run the NSS server under BPXBATCH. For more information these restrictions, see *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`

Tip: 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

To update the TCP/IP profile, complete the tasks in this section:

1. Reserve the NSS port in the TCP/IP profile. We used the default of port 4159 as shown in Example 9-26 at **1**.

   Example 9-26 Reserving the NSS port

   ```
   PORT
   ....
      4159 TCP NSSD; 1
      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 Chapter 16, “Telnet security” on page 677 or for FTP in Chapter 17, “Secure File Transfer Protocol” on page 719. Example 9-27 shows the appropriate TCPCONFIG parameter for AT-TLS.

   Example 9-27 AT-TLS stack enablement

   ```
   TCPCONFIG TTLS
   ```

Tip: You might 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, you might also want to update the IPSec default rules in the TCP/IP profile. See Example 9-28. Consult 8.5.3, “Updating the TCP/IP stack to activate IPSec” on page 266 for information about this task.

Example 9-28 Updating the TCP/IP profile for IP Security on behalf of NSS

```
IPCONFIG ... IPSECURITY ... 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  SECCLASS 100
;  ;;  Administrative access
 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 * 2 ; Network Security
;IPSEC6RULE * * LOG  PROTOCOL TCP SRCPORT 4159 DESTPORT * 3 ; Network Security
;
ENDIPSEC
```

In this example, the numbers correspond to the following information:

1. We enabled IPSECURITY on the IPCONFIG statement.
2. We included default IPSEC rules on behalf of the NSS server for IPv4.
3. We commented out the IPSEC6RULE because we had not enabled IPv6 security in this stack.

Tip: You might need to perform this task for more than one stack, depending on whether you allow 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 might need to consider the following types of policy for NSS communications:

- You must have an AT-TLS policy for the NSS connections between server and client.
- You might 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 that are required in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

9.2.8 Enabling an IKED NSS client to use NSS

The NSS client on SC32 is *IKED*. It interoperates with NSS on SC33 on behalf of TCP/IP stack TCPIPA on SC32 as illustrated in Figure 9-12.

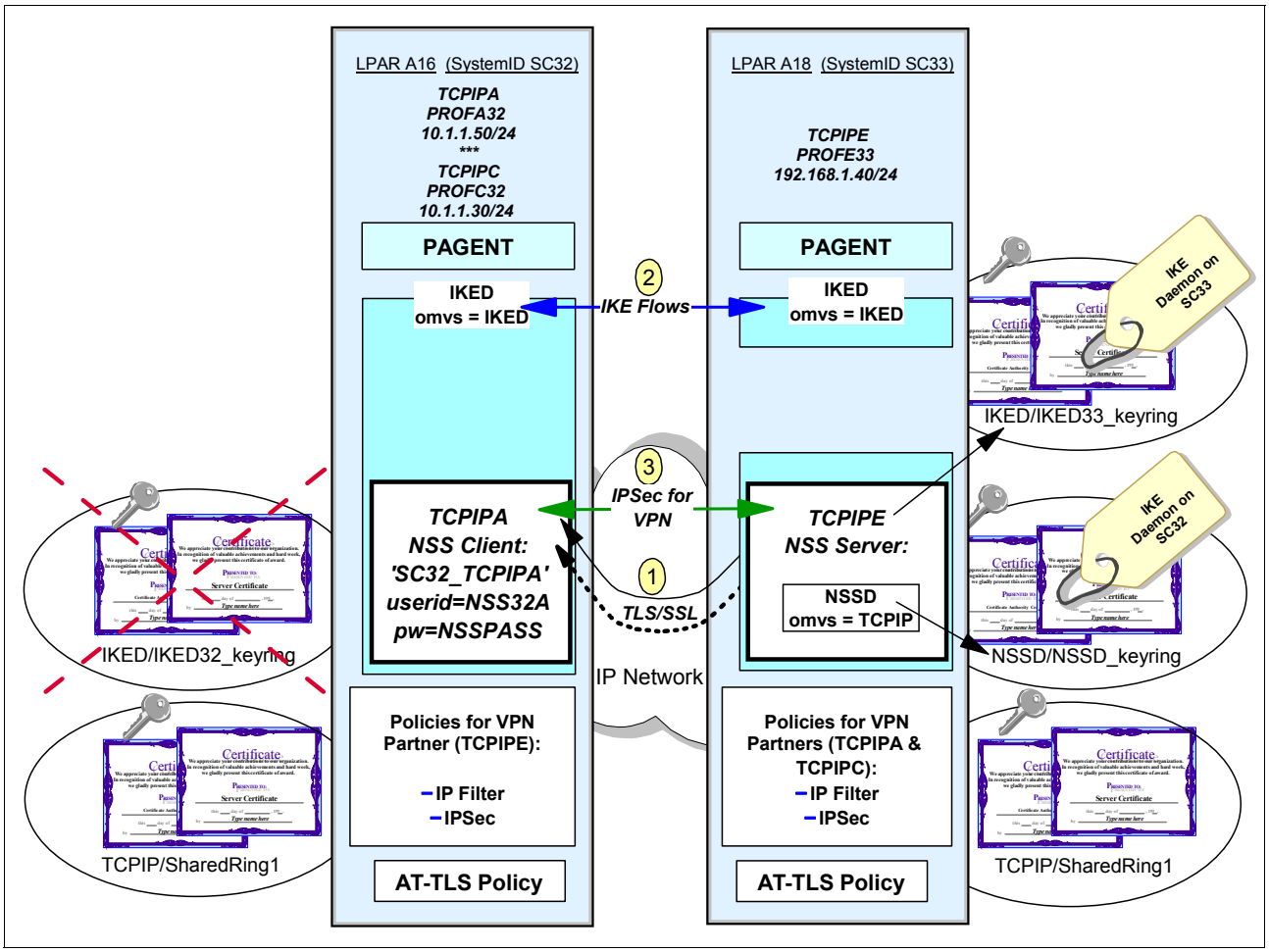


Figure 9-12 TCPIPA is NSS-enabled at system SC32

You can use two methods to create the IKED (SC32) definition file to reflect its usage of NSS for stack TCPIPA:

- You can use the IBM Configuration Assistant.
- You can edit the IKED configuration file directly.

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, Example 9-29 shows the text version of the configuration file.

Example 9-29 The iked.sc32.nss.conf file: 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 1
...
```

```
IkeConfig
{
# IkeSyslogLevel    0-255             (dynamically modifiable)
# Specifies the level of logging to obtain from the IKE daemon.
# Default:          1
IkeSyslogLevel      255 2

# PagentSyslogLevel 0-255             (dynamically modifiable)
# Specifies the level of logging to obtain from pagent through the PAPI
# Default:          0
PagentSyslogLevel   255 3

# Keyring           userid/ringname   (not dynamically modifiable)
# The owning userid and ringname used by the IKE server when performing
# Digital Signature Mode of authentication. The userid must be the userid of
# the process under which IKE will run.
KeyRing IKED/IKED32_keyring 4

# IkeRetries        1-8               (dynamically modifiable)
# Specifies the number of times that an unanswered IKE negotiation
# message will be retransmitted before the negotiation is cancelled.
# Default:          6

# IkeInitWait           1-15          (dynamically modifiable)
# Specifies the number of seconds to wait before the
# first retransmission of an unanswered IKE message.
# Default:          2

# FIPS140    yes, no          (not dynamically modifiable)
# Specifies whether the IKE daemon should perform cryptograhic
# operations by invoking cyptographic modules that are compliant with
# Federal Information Processing Standards (FIPS) publication 140-2's
# Level 1 security requirements.
# Default:          no

# Echo              yes,no            (dynamically modifiable)
# Echoes all IKE daemon log messages to the job output file,
# specified by the IKEDOUT DD (JCL) statement.
# Default:          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.
# Default:          0

# SMF119   IKEAll,IKETunnel,DynTunnel,None (dynamically modifiable)
# Specifies the level of SMF logging.
# Default:          None

# SupportedCertAuth label             (dynamically modifiable)
# 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" 6

}

# NssStackConfig    stackname         (dynamically modifiable)
# Used to configure a stack as a Network Security client.
NssStackConfig TCPIPA 7
{

# ClientName        clientname        (dynamically modifiable)
# Specifies the Network Security client name for the stack.
  ClientName  SC32_TCPIPA 8

# ServiceType       Cert,RemoteMgmt   (dynamically modifiable)
# Specifies the types of centralized services requested
# from the Network Security server.
  ServiceType Cert 9

# ServiceType       Cert,RemoteMgmt   (dynamically modifiable)
# Specifies the types of centralized services requested
# from the Network Security server.
  ServiceType RemoteMgmt 9

# UserId            userid            (dynamically modifiable)
# Specifies the RACF userid that will be used to verify access
# for this stack to the services provided by the
# Network Security server.
  UserId      NSS32A 10

# AuthBy      Password pw, Passticket (dynamically modifiable)
# Specifies the mechanism by which the Network Security server
# should authenticate the client TCPIP stack.
  AuthBy      Password NSSPASS 11
}
```

Example 9-29 on page 366 shows that we will provide IKED services for all TCP/IP stacks on this system (MVS SC32) that are IPSec-enabled through the IPCONFIG parameter IPSecurity. However, for stack TCPIPA at 8, we will use NSS.

In this example, the numbers correspond to the following information:

1. This IKED configuration file was built with the IBM Configuration Assistant.
2. The initial level of IKED logging is high. After testing is complete, we will revert to a lower level of IKED logging.
3. The initial level of PAGENT logging is high. After testing is complete, we will revert to a lower level of PAGENT logging.
4. The key ring that contains the CA certificate and the IKE certificates for digital signature authentication is named *IKED_keyring* and is owned by the superuser *IKED*. This key ring is used by all stacks except for stack TCPIPA. TCPIPA, the NSS client, obtains key ring services from the NSS server.
5. We 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 can be used, regardless of what is in the `SupportedCertAuth` list. The certificates with the named CAs are processed first.

6. Up to this point, all the definitions apply to IKED in general. With the statement `NetworkSecurityServer`, we begin the definitions that apply to NSS.

`NetworkSecurityServer` specifies the IP connection address and port of the NSS server that are 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 **6** under which the certificate was stored in RACF. x500dn is also known as the *Subject’s Name* in a **racdcert** output display. You see the `Subject's Name` in Example 9-30 at **1** with the command **racdcert site list**.

Example 9-30 Output of racdcert site list for TLS Server Certificate of NSS Server

```
Label: CS19 ITSO SharedSite1
Certificate ID: 2QiJmZmiiaOFg8Pi8flAyePi1kDiiIGZhYTiiaOF8UBA
Status: TRUST
Start Date: 2008/11/10 01:00:00
End Date:   2012/11/11 00:59:59
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 that is required in the `iked.conf` file depicted in Example 9-30 requires a comma as a delimiter. Thus, the coding pattern in this case is as follows:

```
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 provided an x500dn identity in the IKED client configuration file, shown in Example 9-29 on page 366, that is composed of three RDNs. However, there are other RDNs that can comprise an x500dn identity. Table 9-1 lists other RDN attributed that are valid entries in a certificate definition.

Table 9-1 Table of attributes recognized by System SSL for certificate processing

Abbreviation	Meaning
C	Country
CN	Common name
DC	Domain component
E	Email address
EMAIL	Email address (preferred)
EMAILADDRESS	Email 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

7. 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 name is *TCPIPA*.

8. The symbolic client name of the TCP/IP stack is SC32_TCPIPA. This symbolic name is used by the server for SERVAUTH processing. Recall that you defined numerous SERVAUTH definitions and permissions in 9.2.6, “Configuring authorizations for NSS” on page 351.

9. Two service types are valid for this NSS client:

- Certificate management
- Remote management using ipsec or an NMI application

Note that we commented out the second `NetworkSecurityServer` definition in Example 9-29 on page 366.

10. The user ID that is presented to the NSS server for either password or pass token authentication is *NSS32A*. You defined this user ID in 9.2.6, “Configuring authorizations for NSS” on page 351.

11. The password that was assigned to this user ID is *NSSPASS*.

9.2.9 Creating NSS files for an IKED Client with z/OSMF Configuration Assistant

In this section, we describe how to use the z/OSMF Configuration Assistant to build the following NSS files:

- For the NSS server:
  - The NSSD configuration file
  - The AT-TLS policy for secure communication between the NSS server and its clients
  - Optionally, IPSec policy to permit traffic to and from the NSS server if the NSS server has been enabled for IPSec
- For the NSS client:
  - The IKED configuration file
  - The AT-TLS policy for secure communication between the NSS client and the server
  - The IPSec policy that permits communication with the NSS server

You can read an overview of NSS by clicking **Help** from the z/OSMF Configuration Assistant Main Perspective panel and then expanding the items on the left navigation panel **Configuration** → **Configuration Assistant Task** → **Overviews** → **NSS Overview** as shown in Figure 9-13.

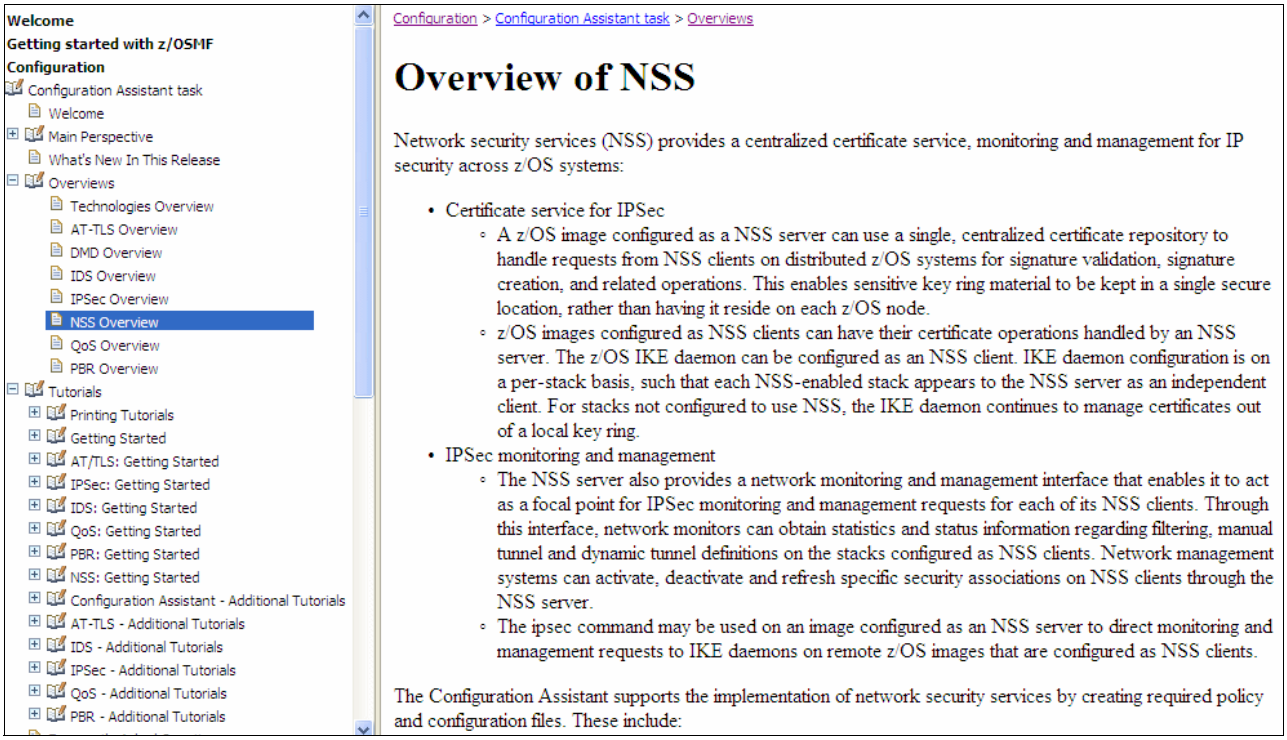


Figure 9-13 Help windows for NSS Overview in IBM Configuration Assistant

From this window, you can read through the overview or click other selections in the left navigation pane to learn how to code for NSS server and client images as shown in Figure 9-14.

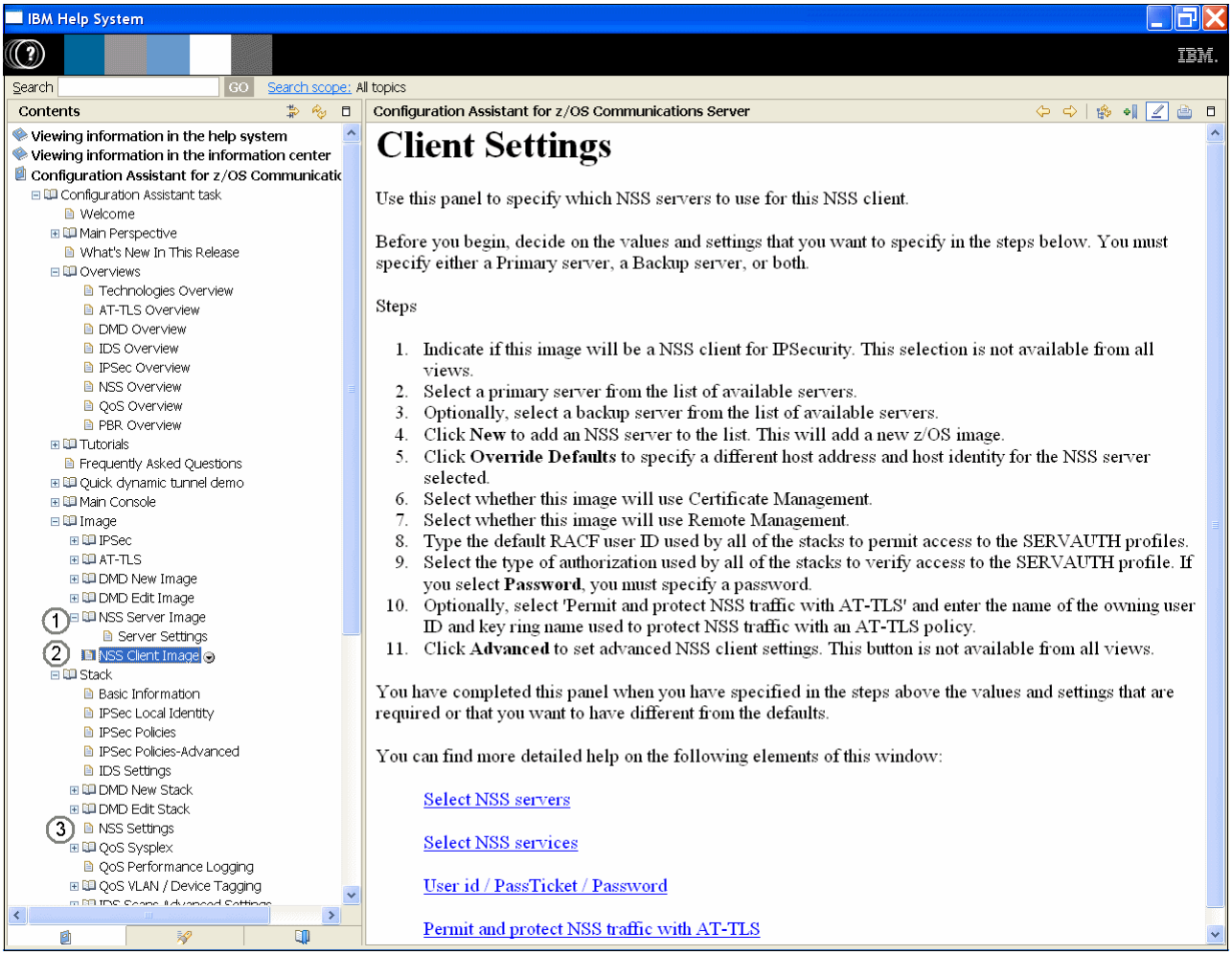


Figure 9-14 Additional Help windows for NSS in IBM Configuration Assistant

The line items indicated with 1, 2, and 3 in Figure 9-14 represent other areas that can be defined for the NSS client and server. After perusing these windows, you can close them to return to the Main Perspective window of the GUI.

Creating a new backing store file and merging the old IPSec file into it

We already have IBM 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-2z0S-RSA.backingstore`, and it is the file that we created in 8.7.3, “Modifying existing policies to use RSA signature mode” on page 309. Our plan is to keep that file as our backup and to create a new backing store file that includes NSS.

Therefore, we create a new backing store in the IBM Configuration Assistant. To create a new backing store file, follow these steps:

1. From the Main Perspective, click **Action** → **Open** → **Create a New Backing Store**. For our scenario, we provided the name `Between-2zOS-RSA-NSS.backingstore` as shown in Figure 9-15.

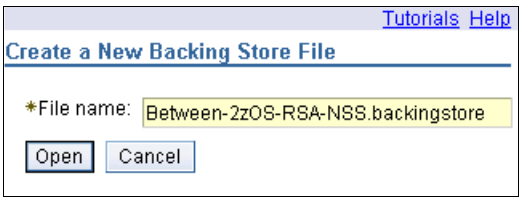


Figure 9-15 Creating a new backing store file for the NSS configuration

2. Click **Open** to return to the Main Perspective for the new backing store file as shown in Figure 9-16.

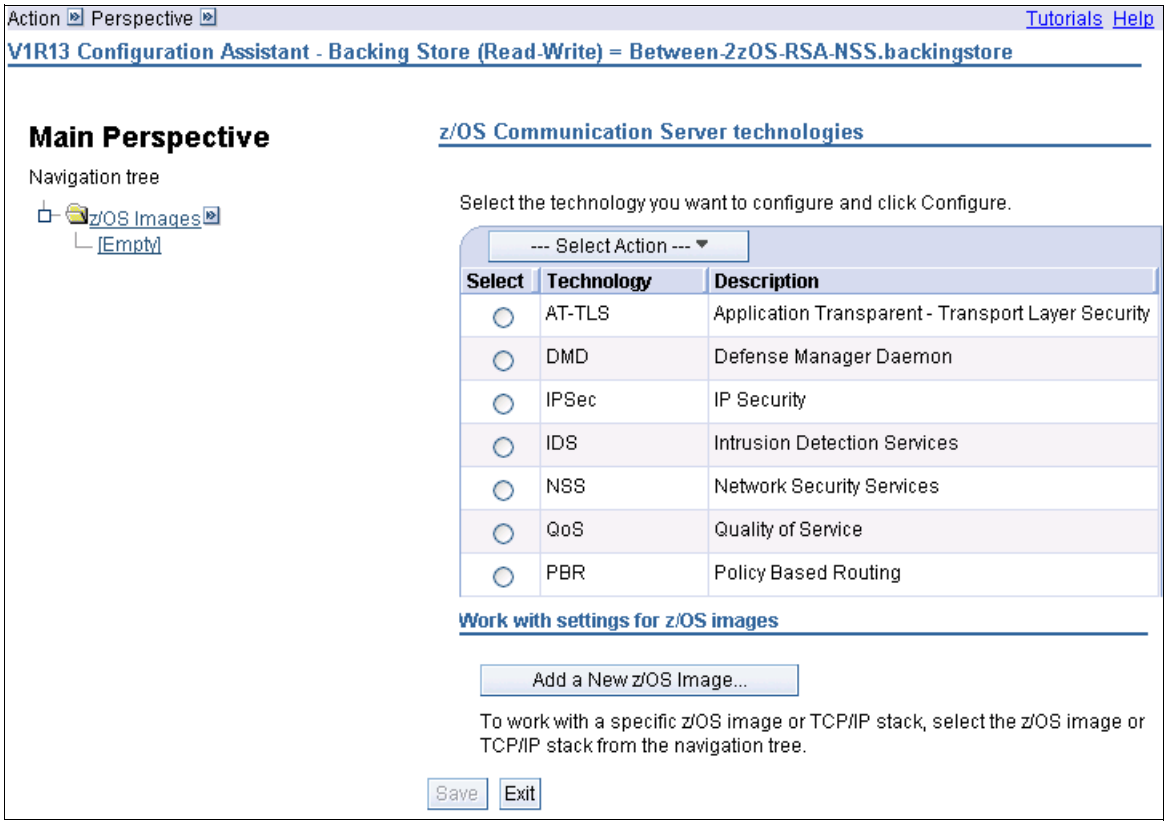


Figure 9-16 Beginning with a new backing store file

In Figure 9-16, notice that the top line of the window shows that you are working on the new backing store that was just created (`Between-2zOS-RSA-NSS.backingstore`).

3. Import the working IPSec backing store that was used in 8.7.3, “Modifying existing policies to use RSA signature mode” on page 309. Click **Action** → **Import** → **Import Backing Store**. Select **Local or shared file** and import the backup of the existing file `Between-2zOS-RSA.backingstore`, as shown in Figure 9-17. Click **OK**.

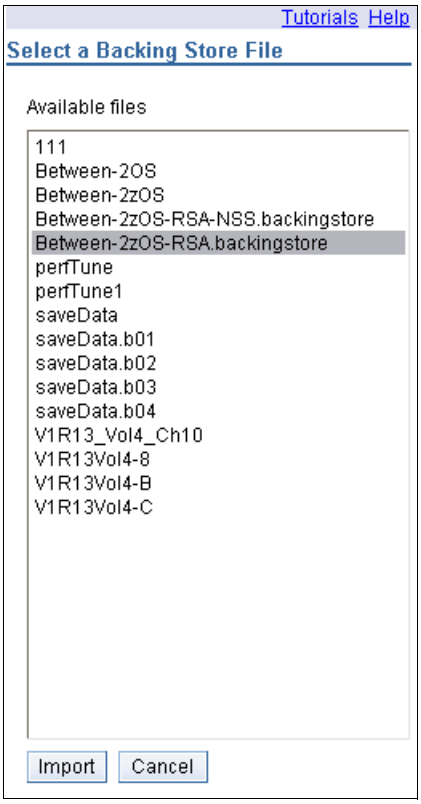


Figure 9-17 Importing an existing backingstore file into a new one

4. When a message appears indicating that the import is successful, click **OK**. A window opens that displays a summary of the import activity. Click **Close**.

5. (Optional) In our scenario, we used another TCP/IP stack named TCPIPC as a non-NSS Client. Before adding NSS, therefore, we created TCPIPC (`10.1.1.30`) on `Image SC32` and implemented the same IPSec connectivity rules as TCPIPA except for IP address, as shown in Figure 9-18. These connectivity rule names are `BasicServices` and `TCPIPE_Traffic_Silver`. Also, we added a connectivity rule for TCPIPC to TCPIPE named `TCPIPC_Traffic_Silver`. All additional policies use digital signature mode. To complete this step, you can see Chapter 8, “IP Security” on page 245.

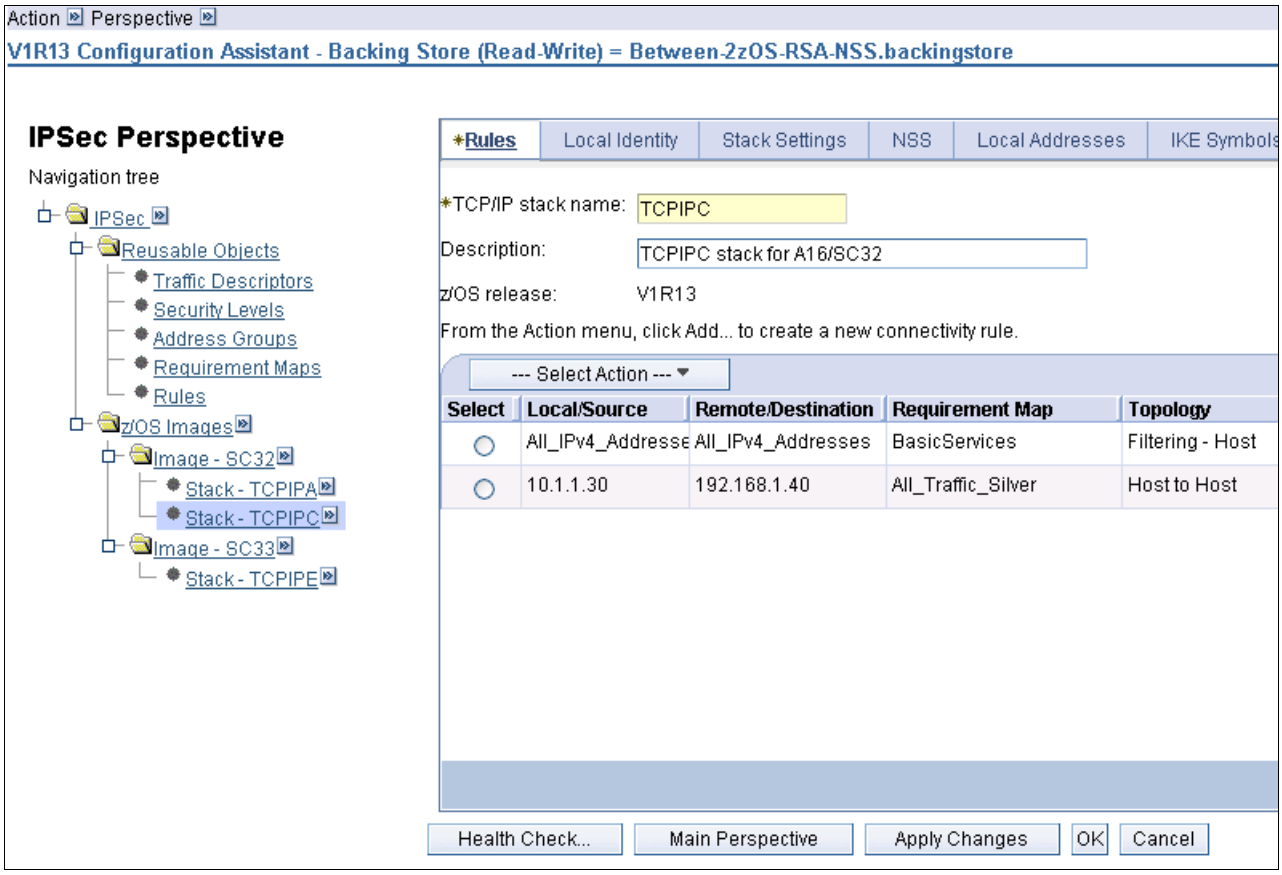


Figure 9-18 Stack TCPIPC on SC32 settings

Adding NSS to the new backing store file

To add NSS to the new backing store, complete the following steps:

1. In the Main Perspective window, select **Image SC33**. Then, highlight **NSS** and Select Action **Enable**. The NSS configuration status shows as `Incomplete` as shown in Figure 9-19. Select Action **Configure**.

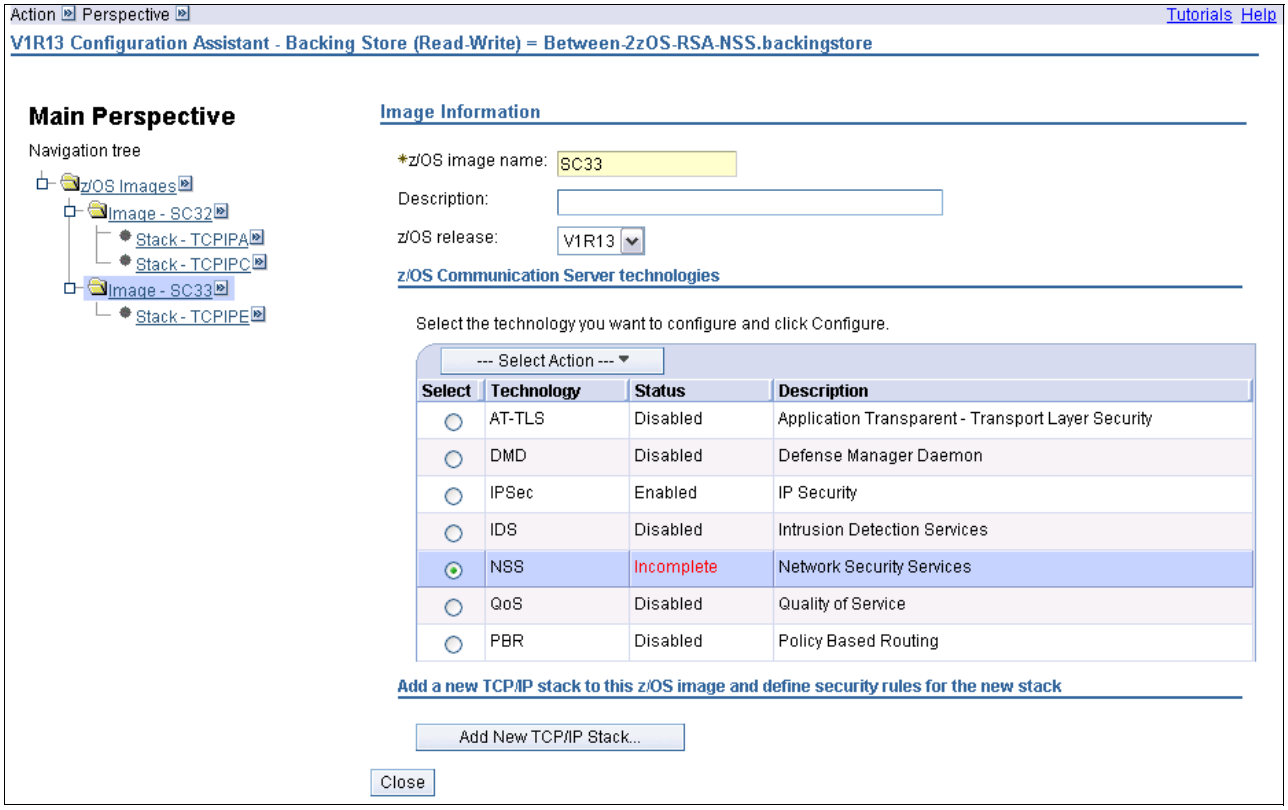


Figure 9-19 Begin to work on NSS for the NSS server at SC33

2. In the next window that opens (Figure 9-20), select the **Server** role for SC33, and click **Next**.

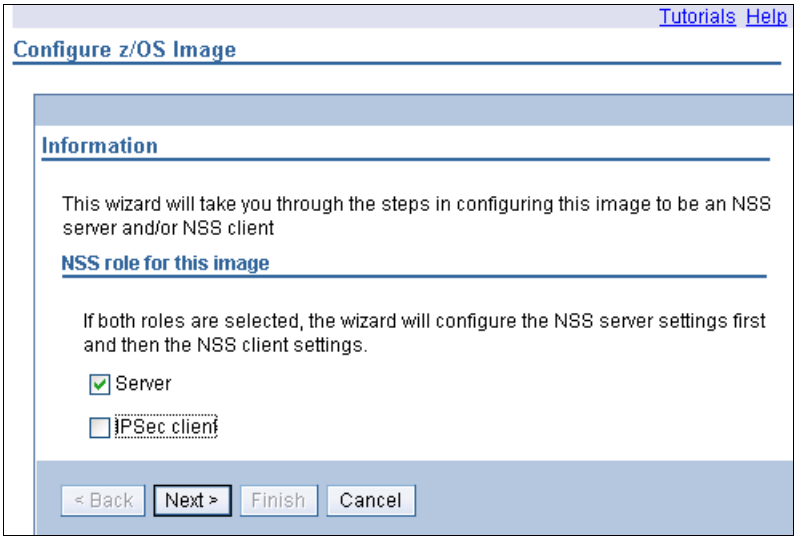


Figure 9-20 NSS role or roles for this MVS image

3. Complete the values in the Configure z/OS Image-Server window as shown in Figure 9-21. For our scenario, we entered the name of the key ring as `NSSD_keyring`. The owning user ID for this ring is NSSD. We entered 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 x.509 server certificate into the X500dn field.

> **Important:** The CN value is from the TLS x.509 certificate and is *not* the CN value from the IKE x.509 certificate. Our TLS certificate was created as a SITE certificate. Therefore, we used the **`racdcert site list`** command to obtain the x500dn value. We then copied it from the command output and pasted it into the field in this window.
>
> Remember that the **`racdcer`**t output, shown in Example 9-30 on page 369, delimits the individual fields of the x500dn with periods. Notice that in Figure 9-21 we changed the periods to commas.

Figure 9-21 also shows that we included the name of the TLS key ring that the NSS server will use during secure communications with the client.

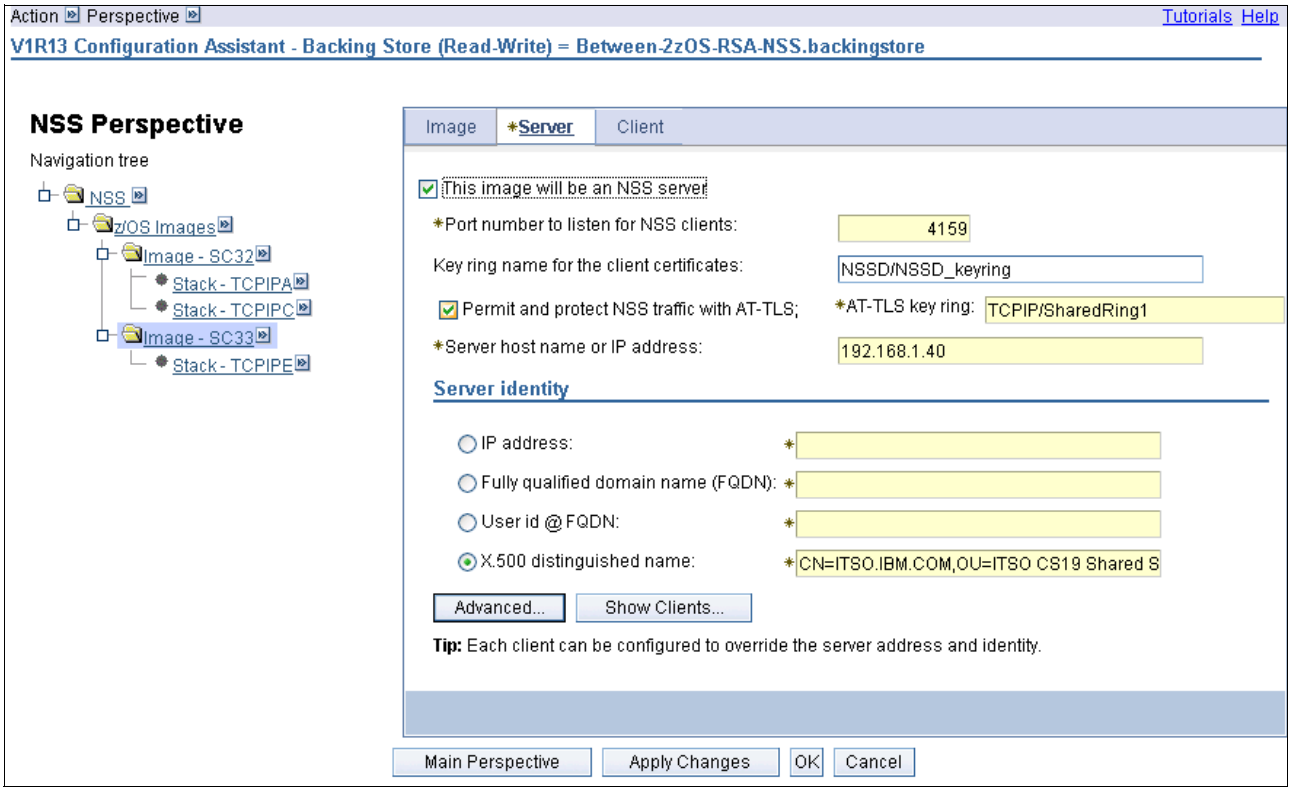


Figure 9-21 NSS definitions of the server

> **Common mistake:** The window in Figure 9-21 calls for two key ring names for the NSS Server node:
>
> - The NSS or IKE key ring that the server uses
> - The AT-TLS key ring that the server uses.
>
> Do not confuse the two key rings.

4. Select **Image** to return to the NSS Perspective window as shown in Figure 9-22.

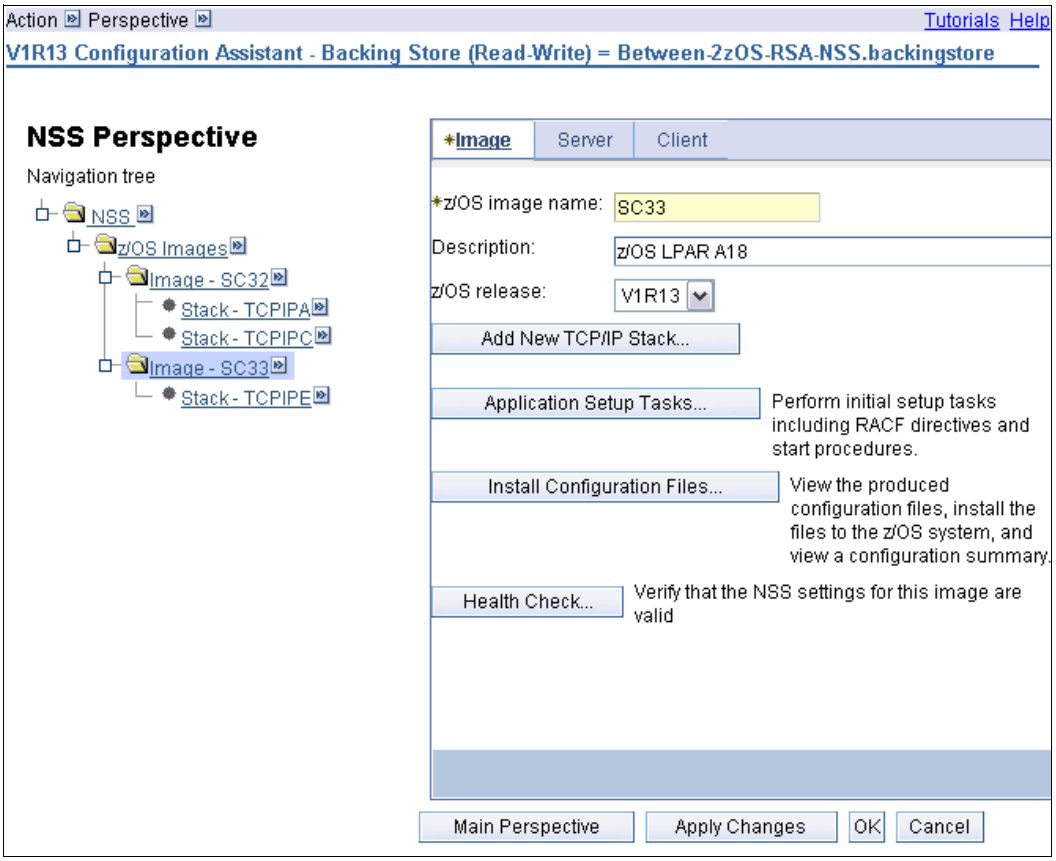


Figure 9-22 Completed NSS perspective for the SC33 image

5. Click **Health Check**. The report provides informational messages that explain that there are no NSS clients. Close the Health Check window and return to the Main Perspective.
6. Select the TCPIPE stack, and the status shows that it is disabled for NSS. You might need to click **Main Perspective**. Select the **NSS** radio button and Select Action **Enable**.

7. Select **Image - SC32**, the client image, where you enable NSS at the client, as shown in Figure 9-23. Click the **NSS** radio button and Select Action **Enable**. The Status is listed as `Incomplete`.

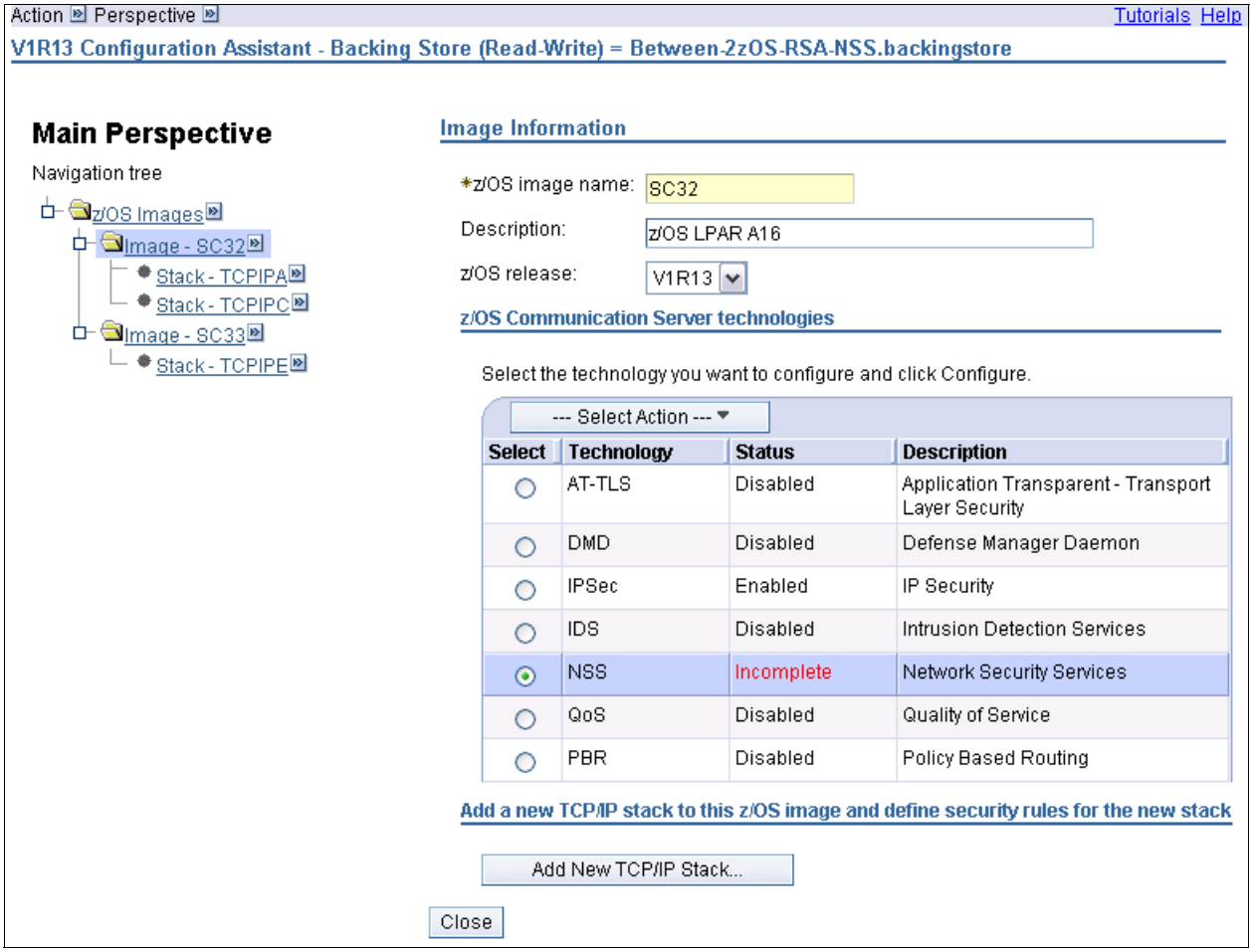


Figure 9-23 Enable Client image for NSS

8. Select **Configure**. If prompted, select **Apply Changes**.

9. Select the **client** tab as shown in Figure 9-24, and click **Next**.

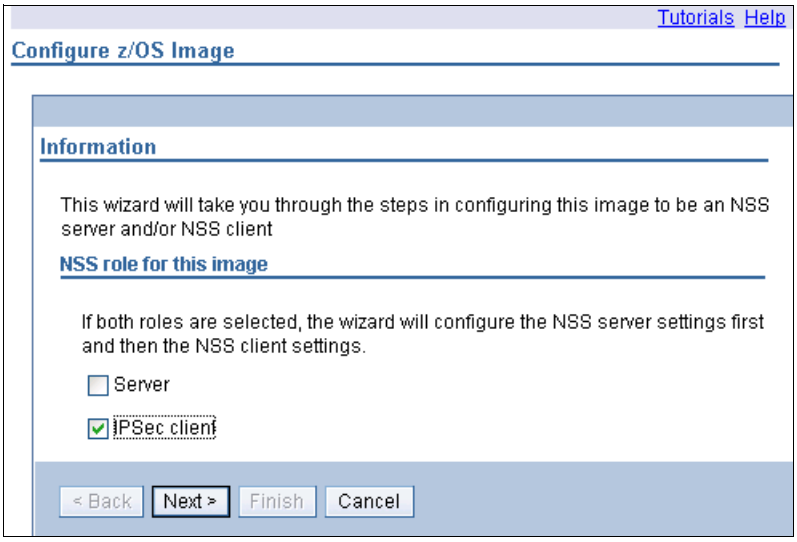


Figure 9-24 Identifying the NSS client role of SC32

10. Select **SC33** from the Primary Server menu as the NSS server. Select both types of Network Security Services (**Certificate Management** and **Remote Management**). Enter the user ID that we want to assign to the NSS client (in our case, `NSS32A`) and request password authentication with a password (in our case, `NSSPASS`). Finally, enter the name of the client's AT-TLS certificate key ring (TCPIP/`SharedRing1`), and click **OK**.

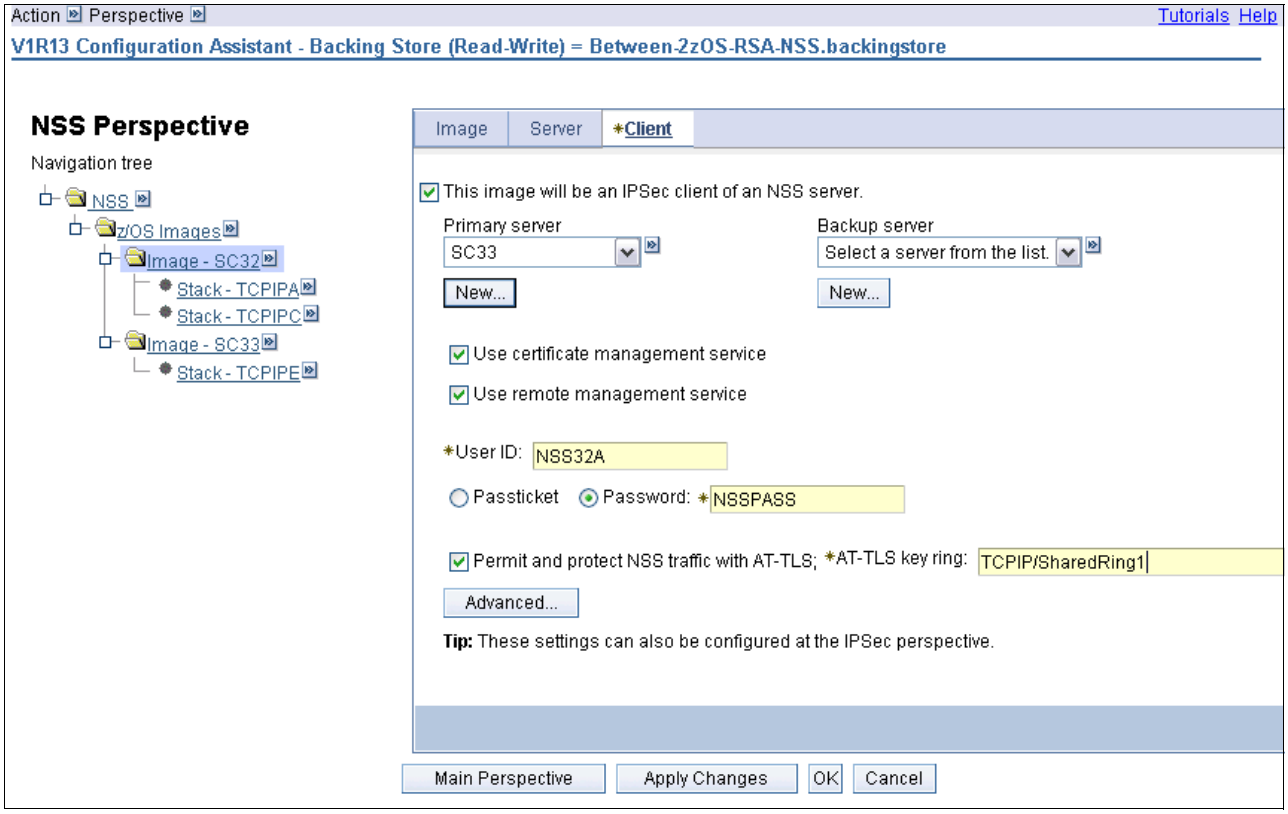


Figure 9-25 Identifying NSS server, NSS client user ID, and client AT-TLS key ring

Tip: We planned the selections that we made in Figure 9-25 when we created the prerequisite environment for NSS and its clients. See 9.2.6, “Configuring authorizations for NSS” on page 351 for more detailed information about this topic.

11. Click the Image tab. Click **Health Check**. The resulting messages explain that no client TCP/IP stack is enabled (as shown in Figure 9-26). Close the Help window and return to the Main Perspective.

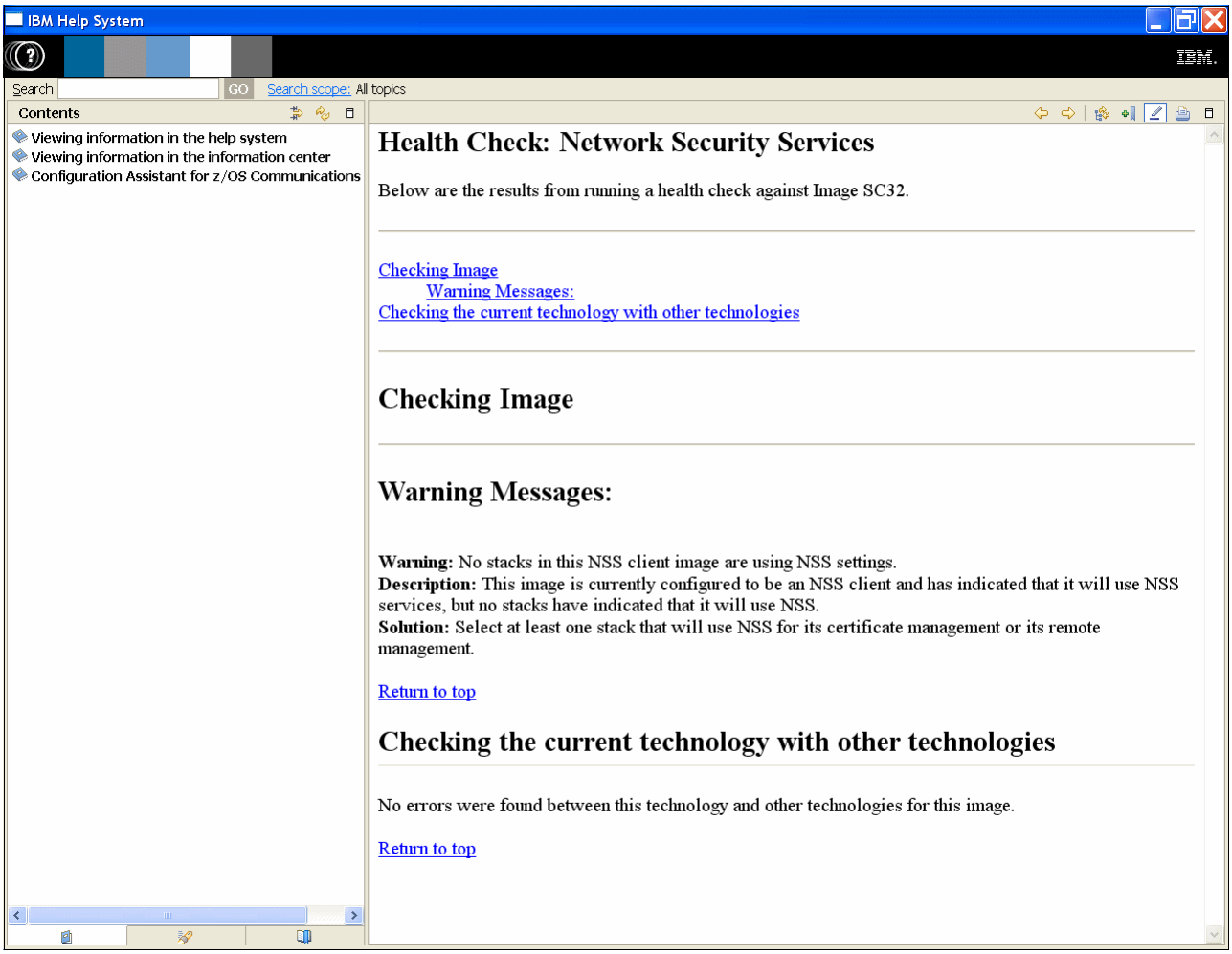


Figure 9-26 Warning messages: No NSS client TCPIP stacks

12. Select the TCPIPA stack, and the status shows that it is disabled for NSS. Select **NSS** from the window and press **Enable** in the top right corner.

13. In the NSS Perspective window, click the IPSec Client Settings tab, select **IPSec will use NSS**, and then complete the fields as shown in Figure 9-27. In our scenario, we leave the default name of `SC32_TCPIPA` for the NSS symbolic client name. Click **Apply Changes** and then **OK**.

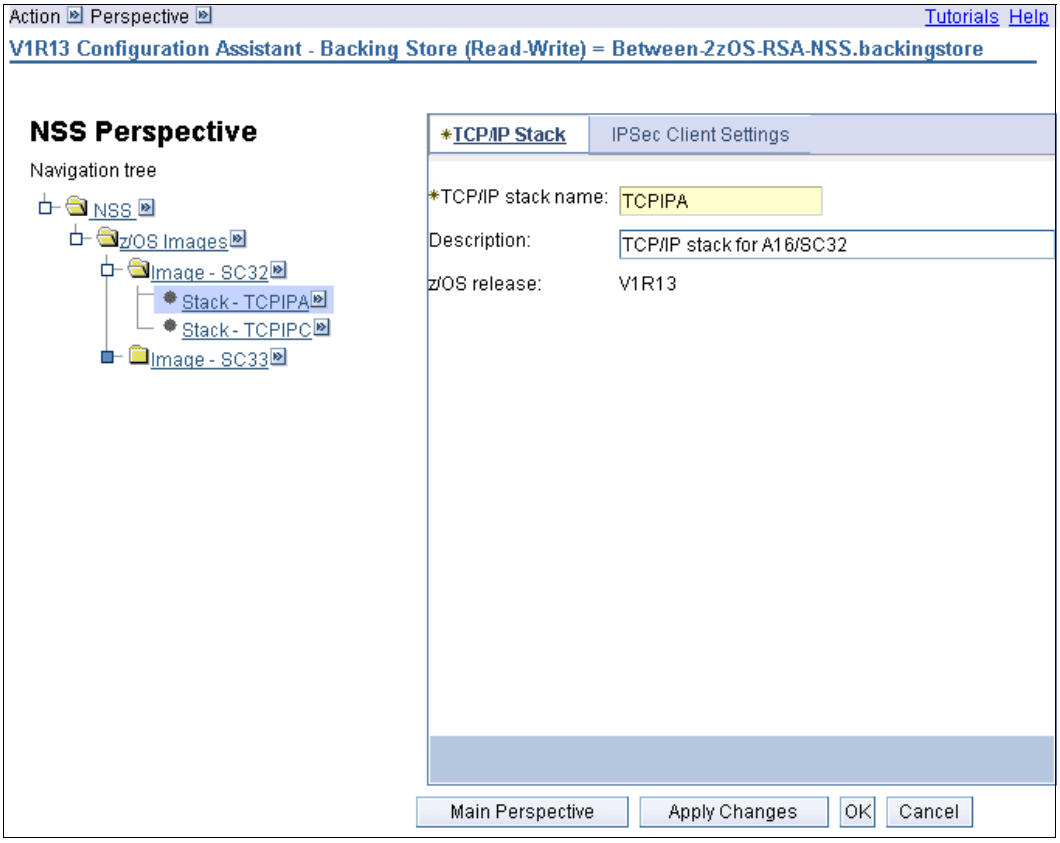


Figure 9-27 NSS client identity: Symbolic client name

14. You can then select an **NSS Summary** as displayed in Figure 9-28. Close the Help window.

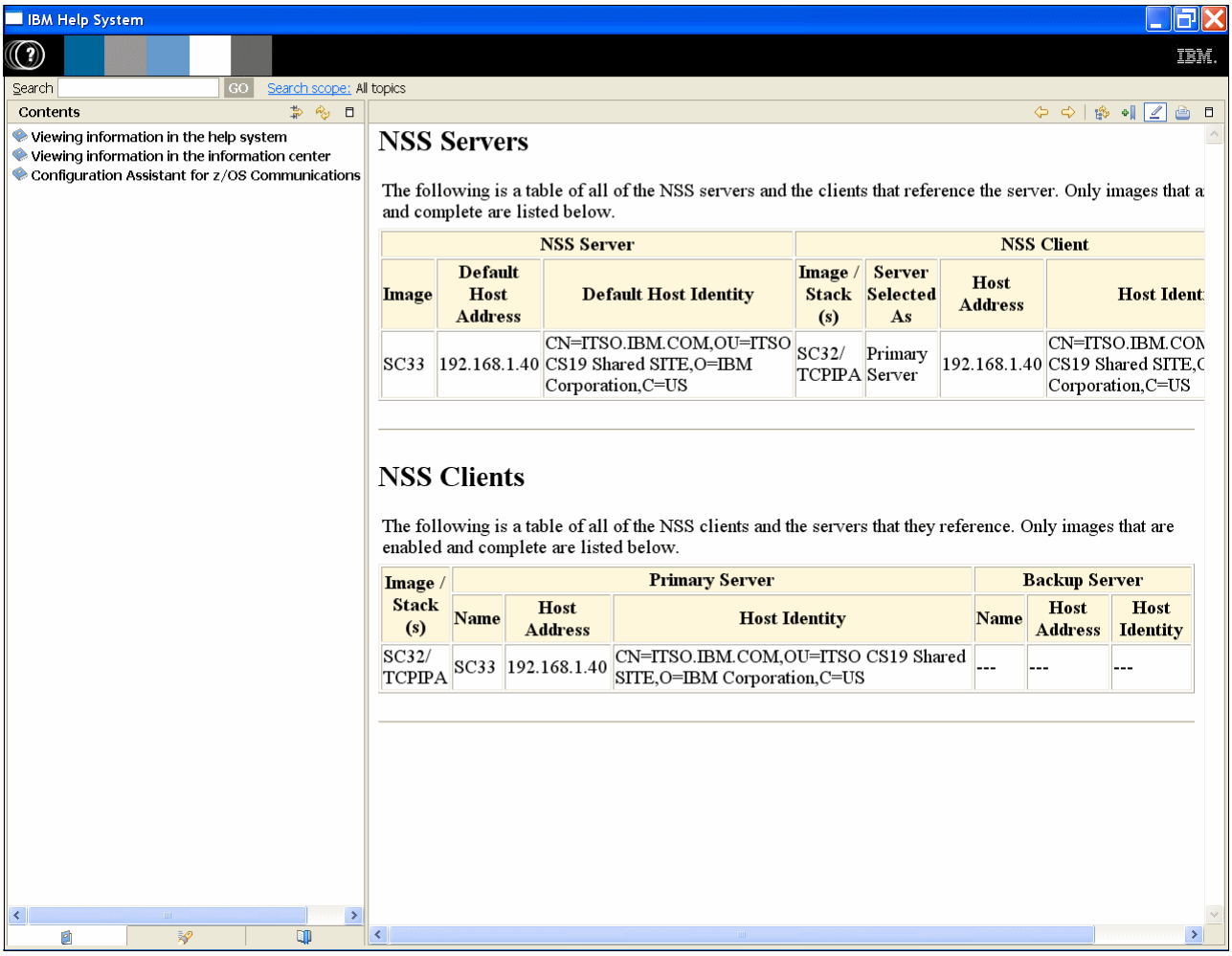


Figure 9-28 Summary of NSS definitions in this backing store file

15. (Optional) Highlight **TCPIPC** in Image - SC32 as shown in Figure 9-28 on page 383. So far only the TCPIPA stack is enabled for NSS. We allow a second stack in image SC32 to use native IKED services and leave it *Stack Disabled* for NSS. Return to the **Main Perspective**.

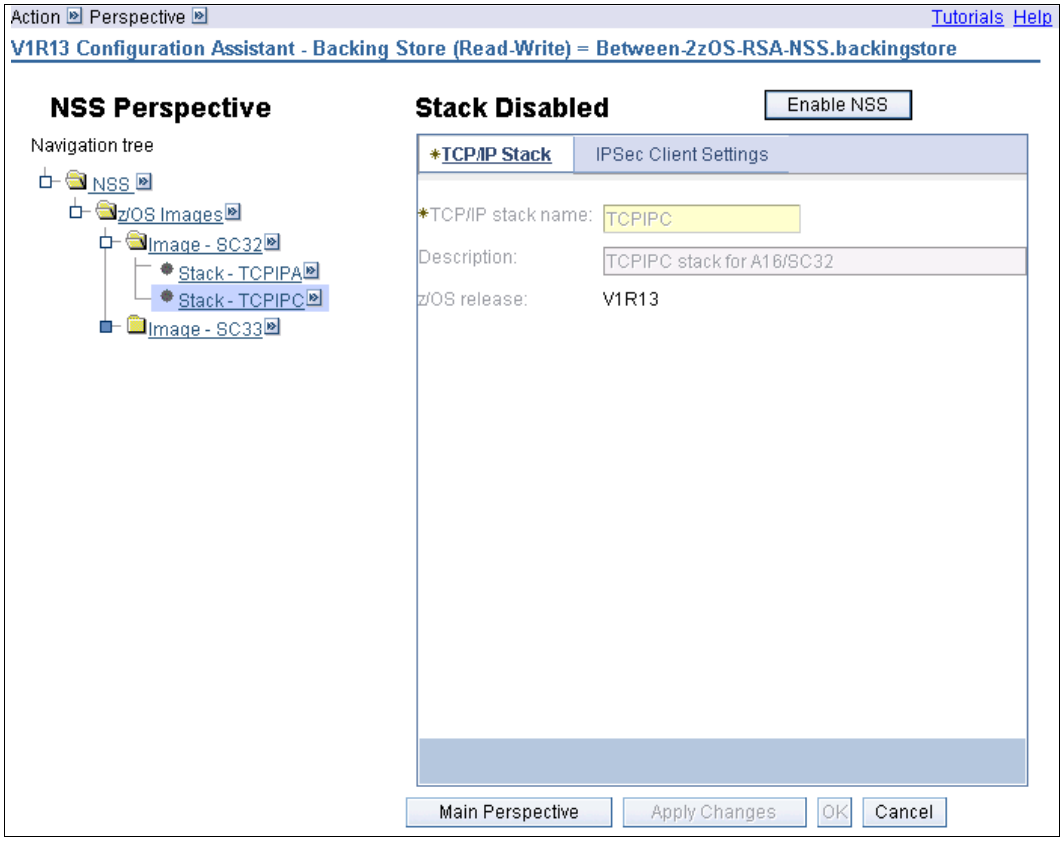


Figure 9-29 NSS Perspective: TCPIPC stack is not enabled for NSS

16. Highlight SC33 (TCPIPE) as shown in Figure 9-30 and select the Main Perspective for this stack. Note 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 the next section, we explain how to verify the IPSec connectivity rules that exist for the NSS server and client.

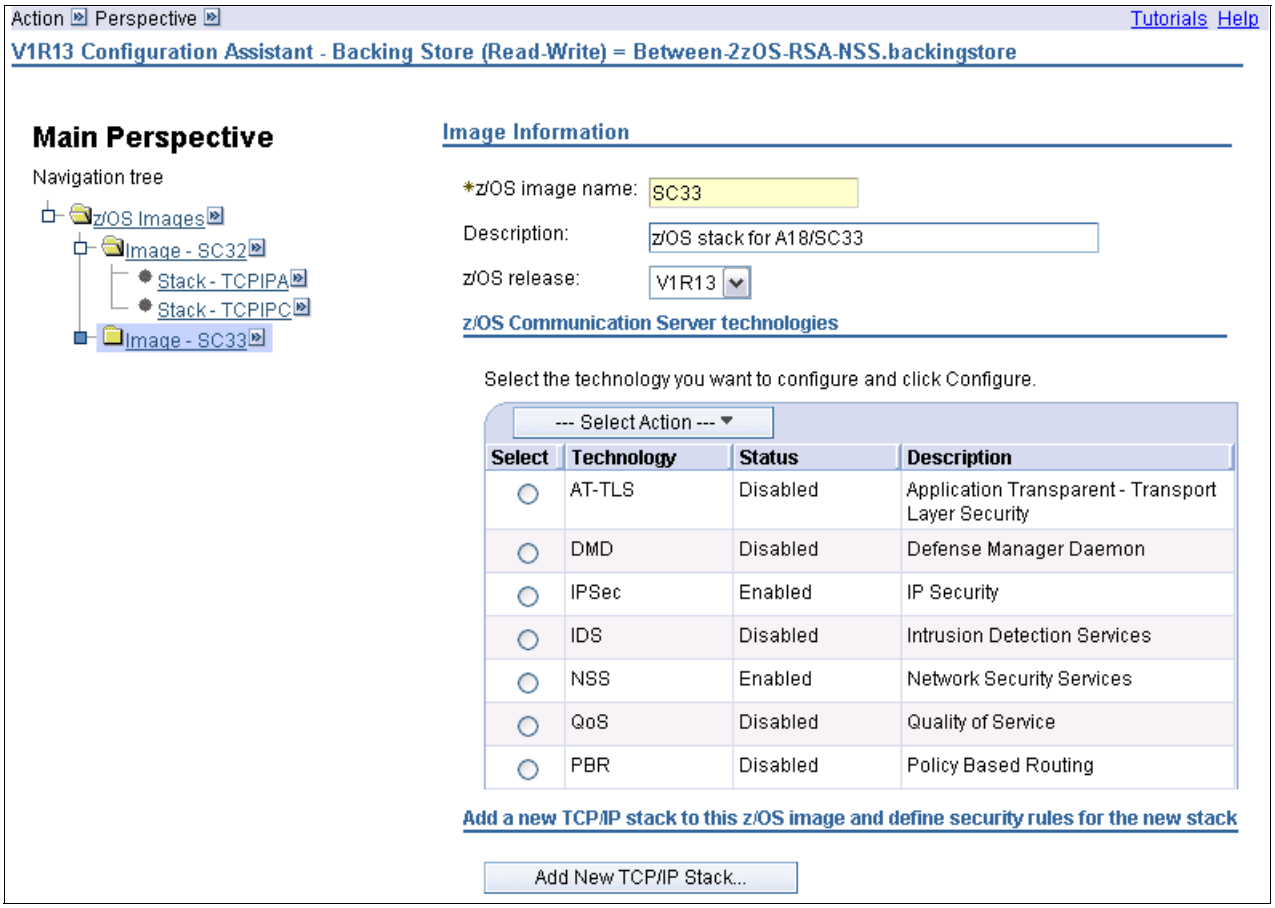


Figure 9-30 AT-TLS is not yet enabled at TCPIPE, the NSS server stack

Verifying and augmenting existing IPSec policy for NSS

To verify and augment the existing IPSec policy for NSS, follow these steps:

1. From the Main Perspective, go to the IPSec Perspective and highlight `TCPIPE` (the NSS server) stack. Look for a connectivity rule entry that can apply to the relationship between the NSS server and the NSS client, as shown in Figure 9-31.

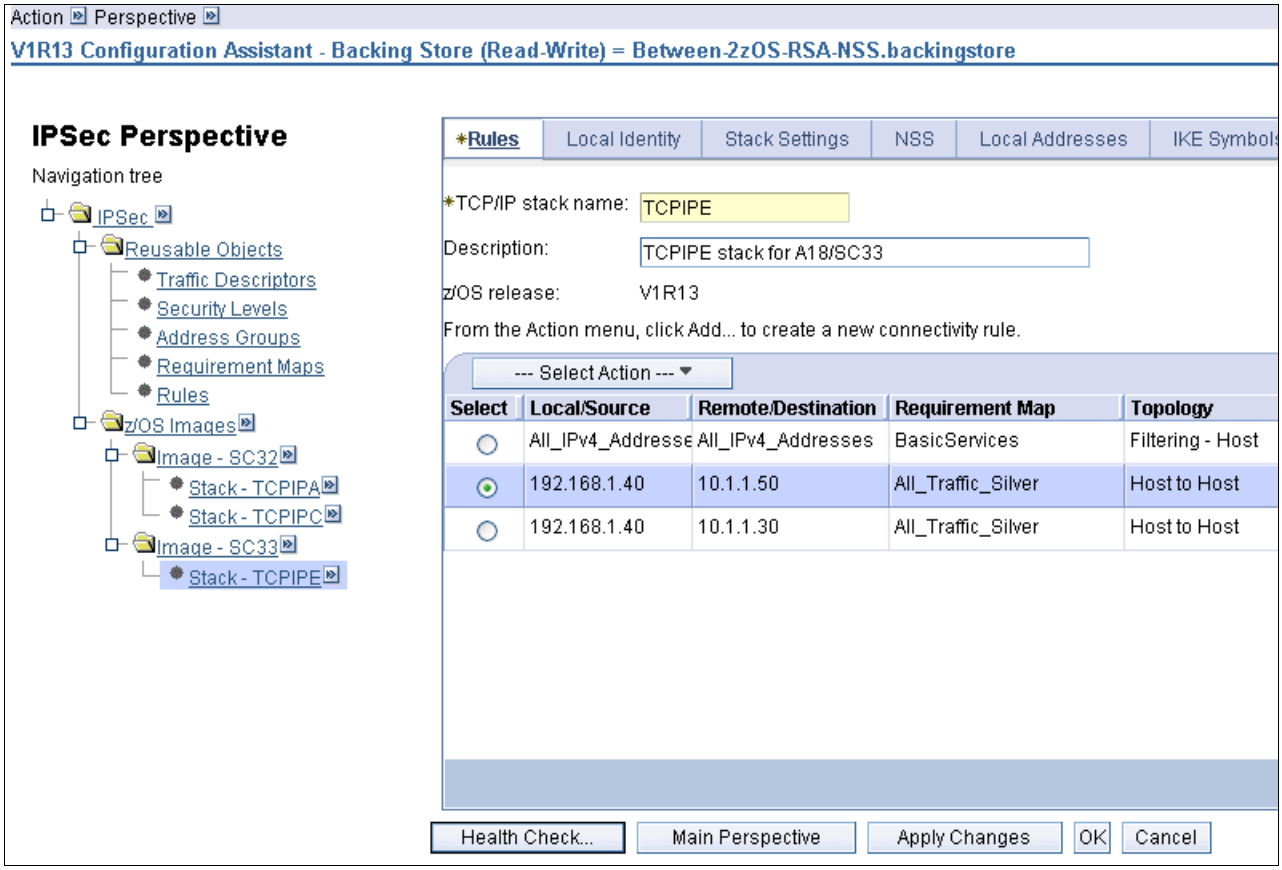


Figure 9-31 IPSec Perspective: Connectivity rules for NSS Server and Client

2. Click **View Details** for the highlighted connectivity rule to determine whether the rule is appropriate for NSS traffic (see Figure 9-32).

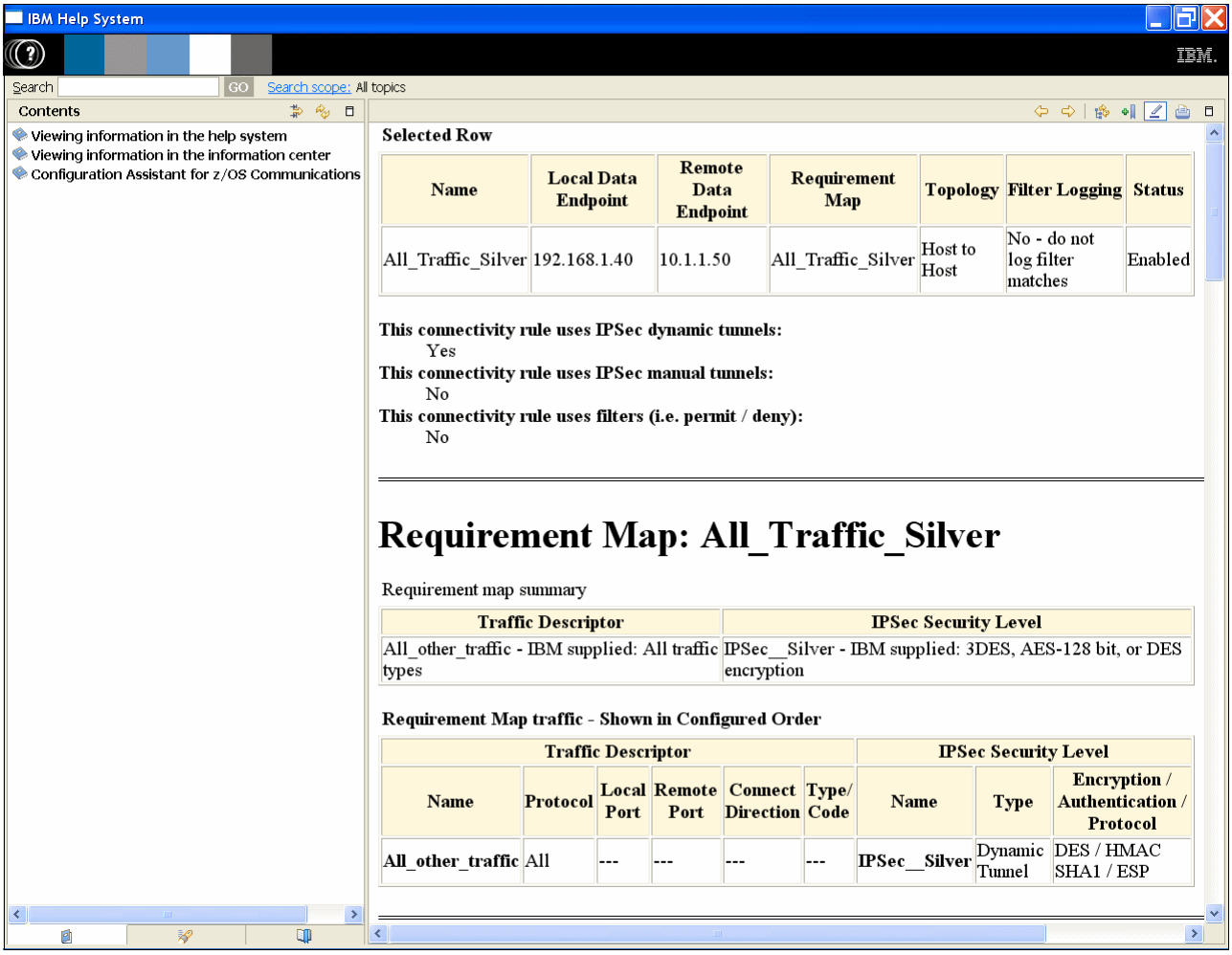


Figure 9-32 IPSec Connectivity: All traffic is encrypted

Note on the Requirement Map that all the traffic between the two addresses is encrypted. However, NSS uses TLS for encryption. You will not want to apply IPSec security to a TLS encrypted session because it is redundant. Such double encryption adds overhead and can cause a performance impact.

3. To take care of the problem of double encryption along with the ensuing overhead that leads to performance problems, you need to add a new Connectivity Rule to the policies. This IPSec rule specifies no encryption of data to and from Port 4159 (the NSS server port). Begin by closing the Help window to return to the Connectivity Rules window of IPSec as shown in Figure 9-33. Click **Select Action** → **Add**. When the Welcome window opens, keep the defaults (do not change them), and click **Next** to continue.

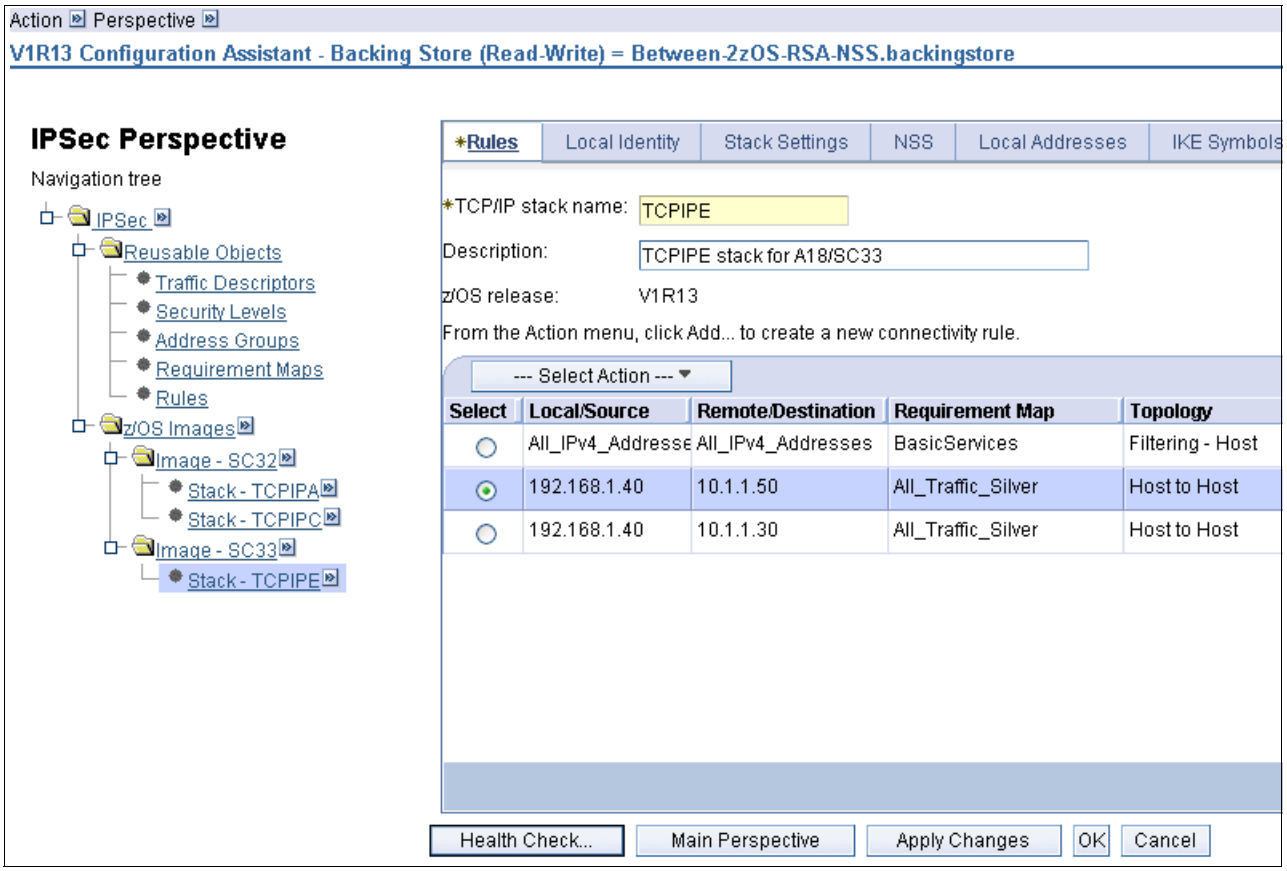


Figure 9-33 Adding a new IPSec connectivity rule to prevent double encryption for NSS

4. In the New Connectivity Rule: Network Topology window, shown in Figure 9-34, ensure that the **Filtering only** option is selected. This NSS rule is for local traffic only. The goal is to create a filtering rule that excludes NSS traffic from IPSec. Click **Next** to continue.

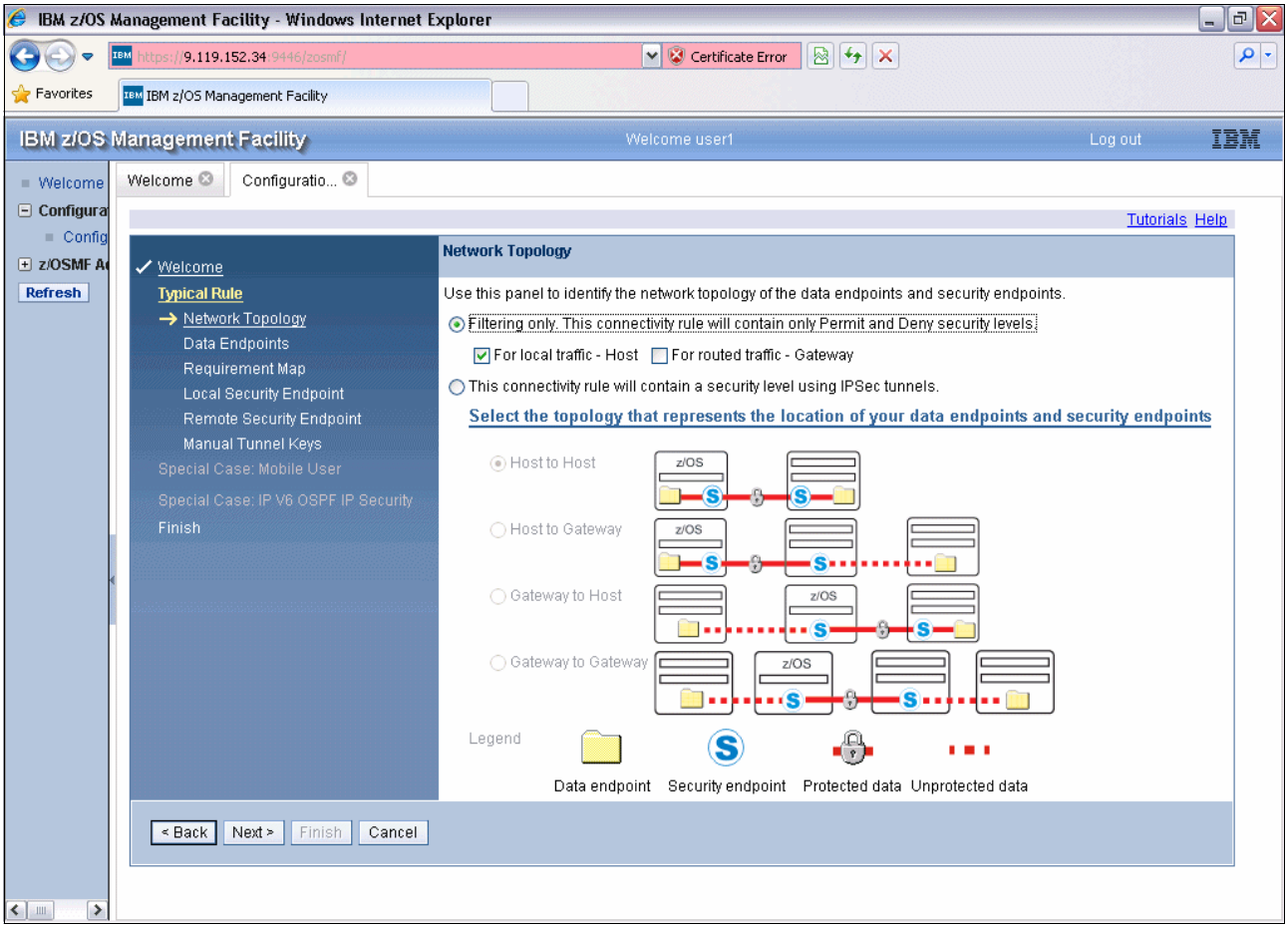


Figure 9-34 NSS Connectivity Rule: Select radio button for no encryption

5. In the New Connectivity Rule: Data Endpoints window, specify the data endpoints addresses and the new rule name (in our case, `NSS_Connectivity_Rule`), as in Figure 9-35, and Click **Next**.

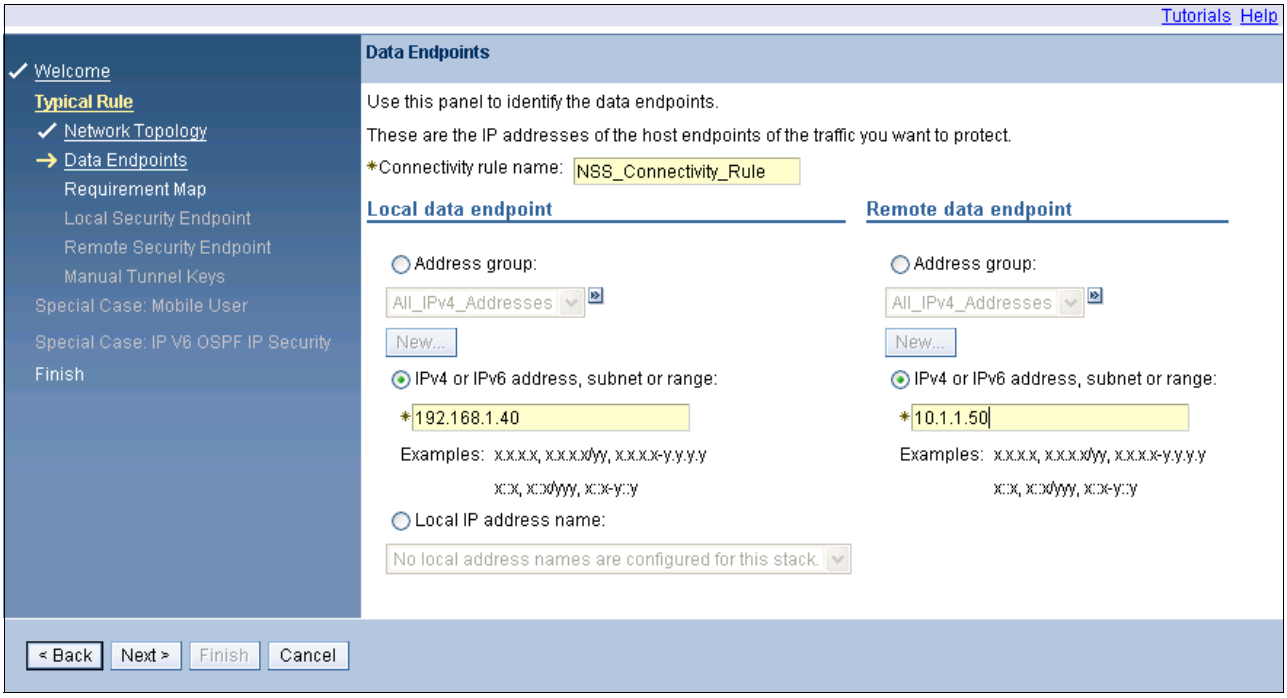


Figure 9-35 NSS Connectivity Rule

6. Create a Requirement Map for NSS traffic. There are already traffic descriptors in the Configuration Assistant for NSS traffic. In the New Connectivity Rule: Requirement Map window, shown in Figure 9-36, use the Traffic Descriptor pull-down menu to select the descriptor for NSS client and server traffic. Use the Security Level pull-down menu to permit this traffic. We removed the default deny rule in this window. Enter a name (we used `NSS_RequMap_NoEncrypt`), and click **Next** to continue.

   In our case, we built a single requirement map for both the NSS client and server. If your security requires more granularity, you can choose to build two separate requirement maps for the client and for the server.

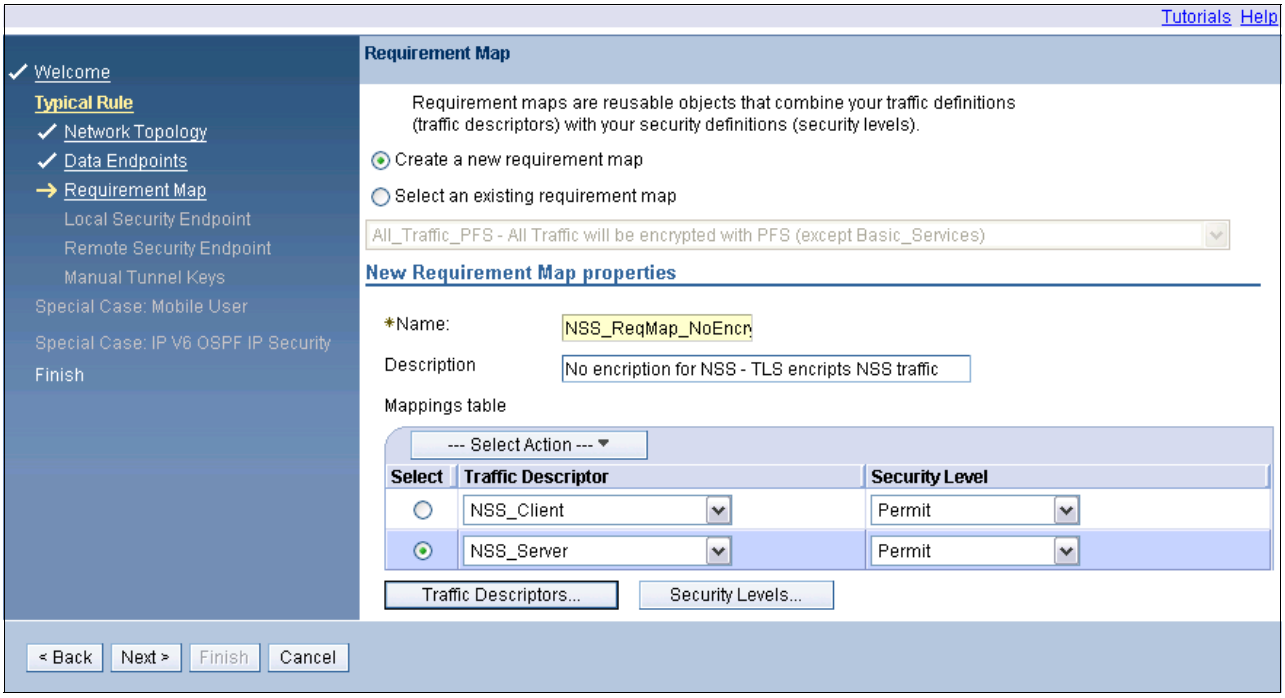


Figure 9-36 Create a requirement map for NSS

7. In the New Connectivity Rule - Finish window, choose any logging options, or simply click **Finish** to continue.
8. After clicking **Finish**, you are returned to the IPSec Perspective Connectivity Rules for stack TCPIPE. Highlight the newly created rule for NSS, and click **View Details**.

   The Help window shows that NSS traffic to and from port 4159 is permitted without enforcing IPSec encryption. See Figure 9-37.

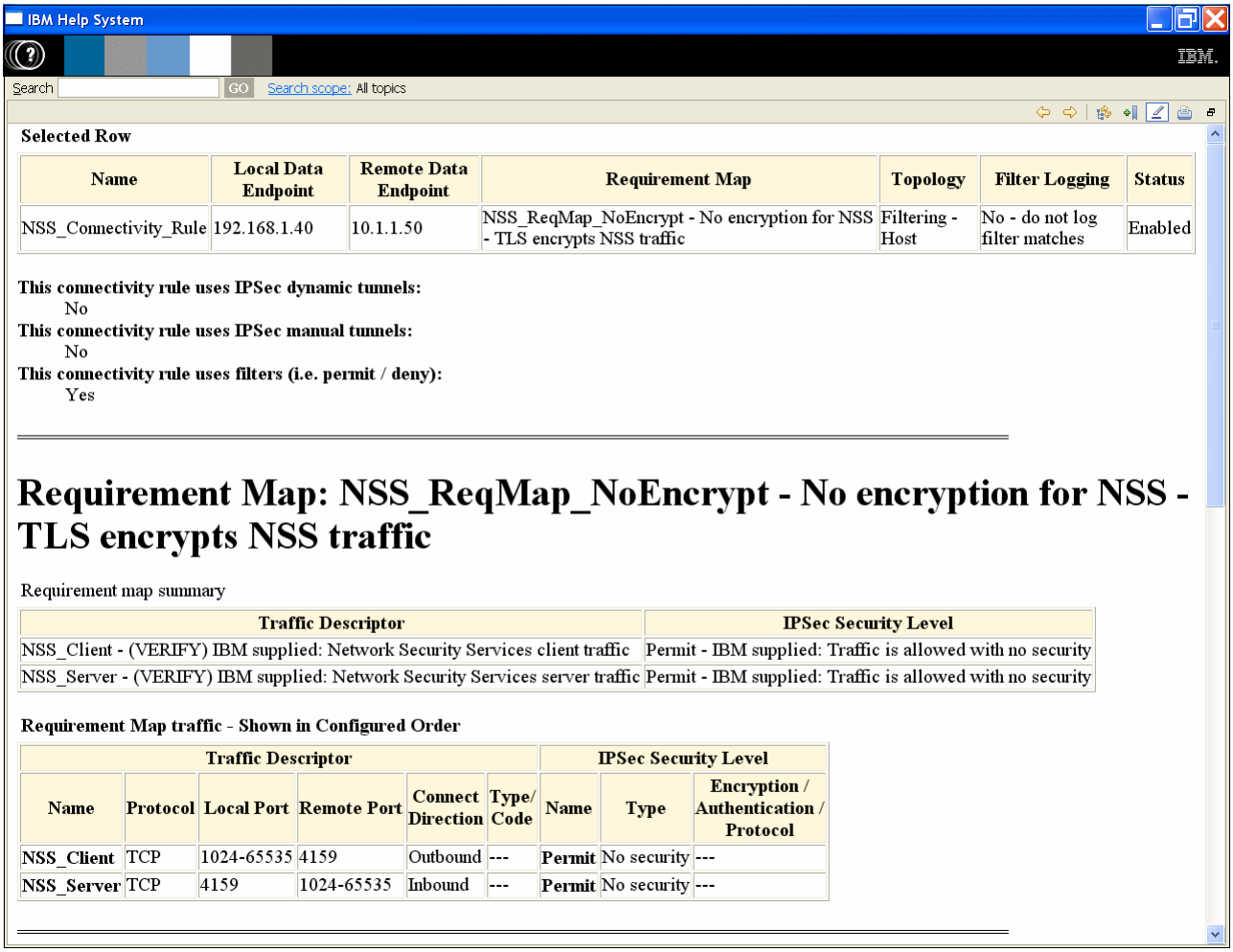


Figure 9-37 Details of NSS requirement map

9. Close the Help window to return to the IPSec Perspective Connectivity Rules window shown in Figure 9-38.

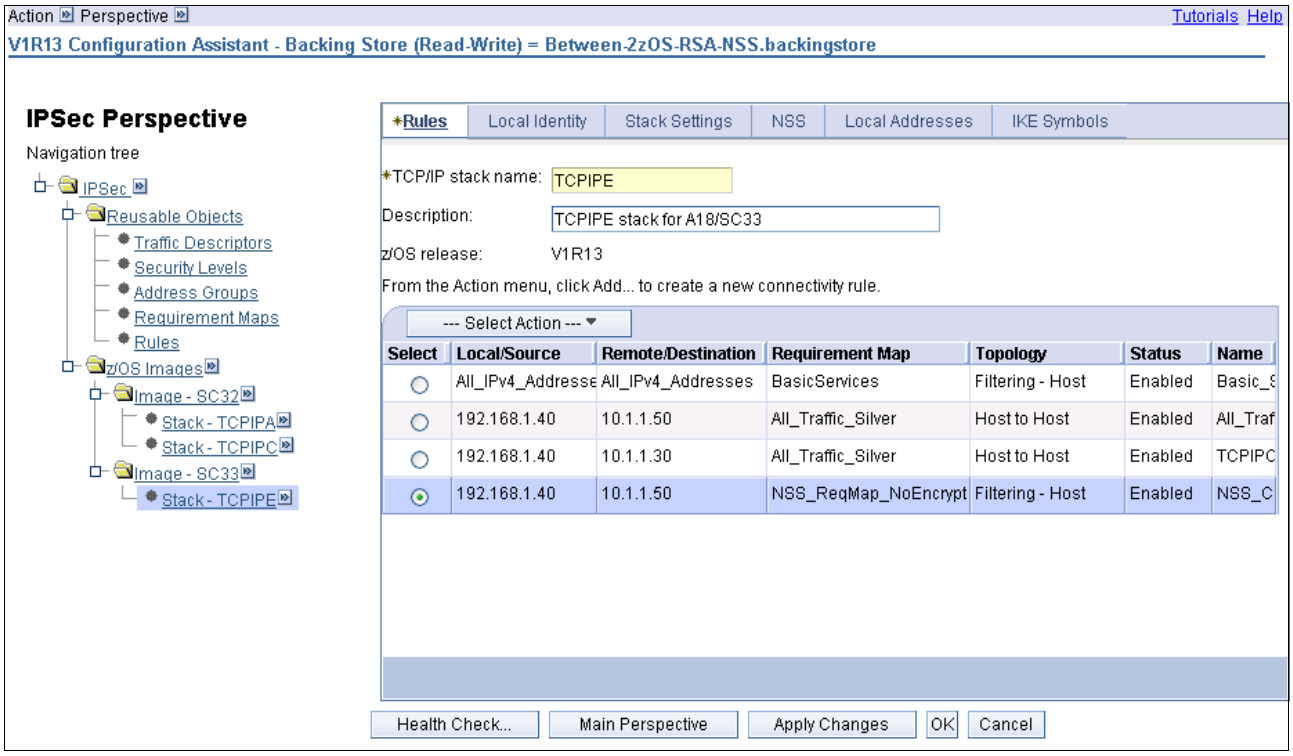


Figure 9-38 IPSec perspective with new Connectivity Rule for NSS

Notice that the rule is placed at the lowest priority of the list. In this position, the rule will never trigger because the IPSec before it will be selected first. You need to move the NSS rule above the encryption rule so that it is found first. This position is acceptable because the NSS rule matches only NSS traffic. Other traffic will not match and will drop down in priority to be captured by the IPSec rule.

To move the rule, highlight the rule, and click **Move Up** until the Permit rule is above the IPSec encryption rule with the Requirement Map of All_Traffic_Silver. Do not move it above the Basic Services rule, because there should be more traffic matches under Basic Services than there would be for NSS traffic. Remember, if possible, place the more popular rules higher in the list.

Click **Apply Changes**.

10. Select the **SC32 TCPIPA** stack, as shown in Figure 9-39, which also needs these new IP connectivity rules and requirement maps, then click **Select Action** → **Add**. As with the TCPIPE stack earlier, select **Typical** and **Filtering only** on the subsequent two windows.

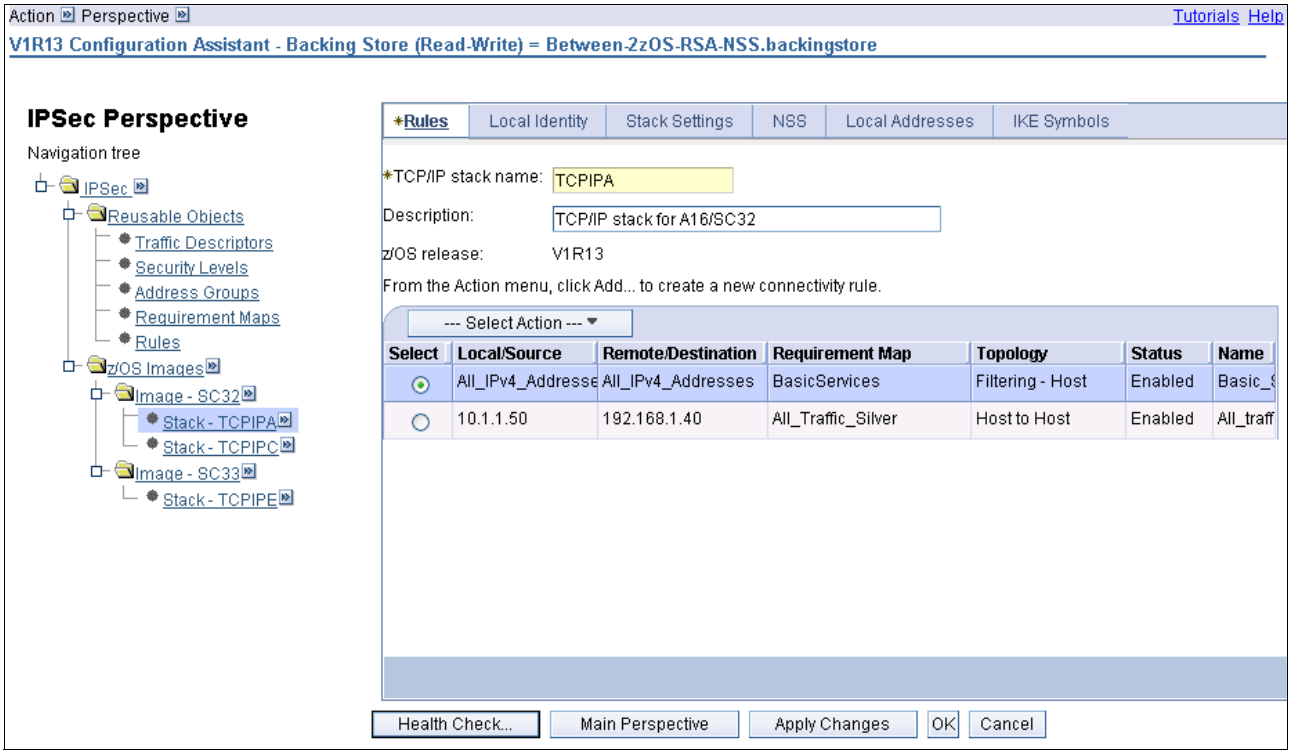


Figure 9-39 New connectivity rule needed for NSS at TCPIPA stack

11. in the New Connectivity Rule: Data Endpoints window, enter the endpoints addresses and the rule name information as shown in Figure 9-40, then click **Next**.

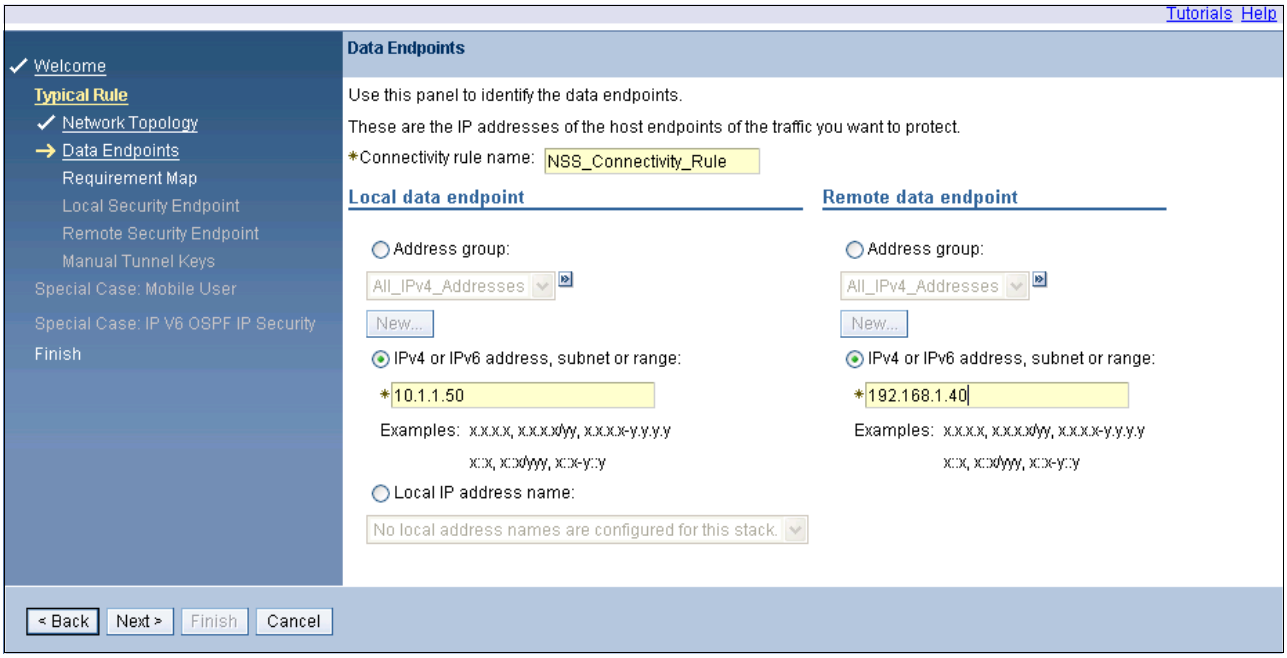


Figure 9-40 Connectivity Rule for Client to Server

12. Because you have already created a requirement map for NSS, you can now simply select an existing requirement map from this window (see Figure 9-41), and then click **Next**. Select your logging options on the next window and click **Finish**.

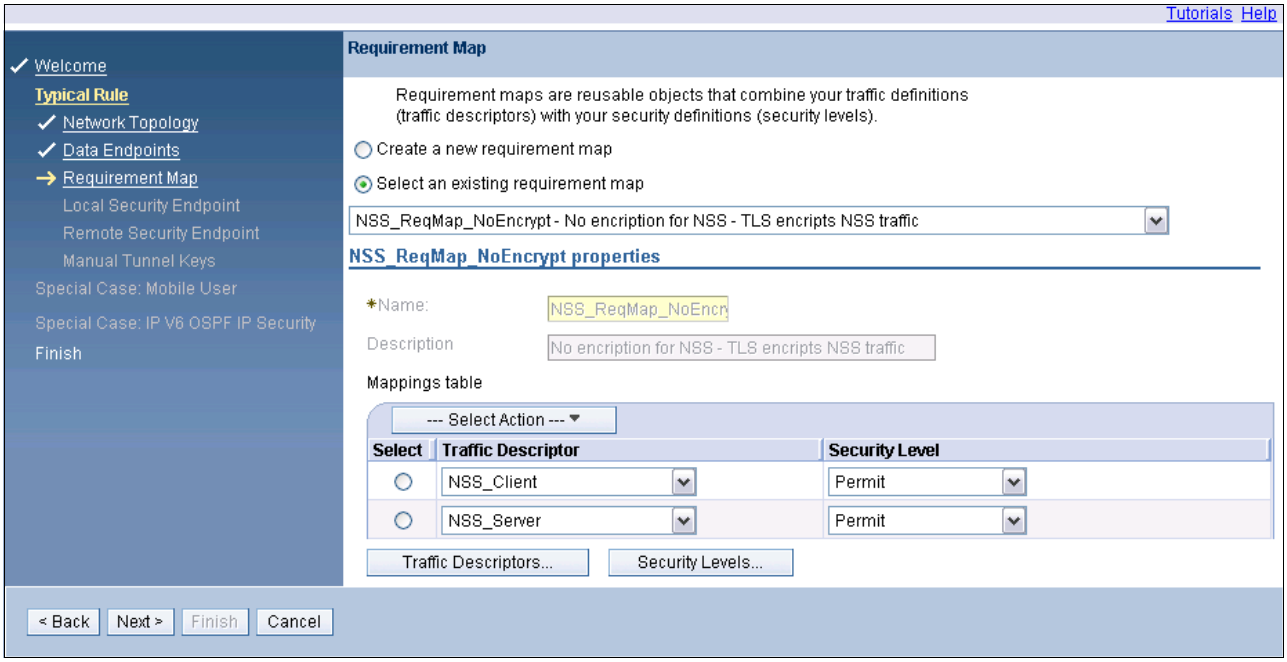


Figure 9-41 Available requirement maps for the NSS connectivity rule

13. Click **Finish**, and the TCPIPA now has the appropriate rules for NSS. As before, use the **Move Up** button to place this rule above any IPSec rules that might match on the same traffic. Click **Apply Changes**.
14. Select **Main Perspective**. TCPIPA is highlighted in the Main Perspective window, and AT-TLS is showing a status of `Disabled` for TCPIPA (see Figure 9-42).

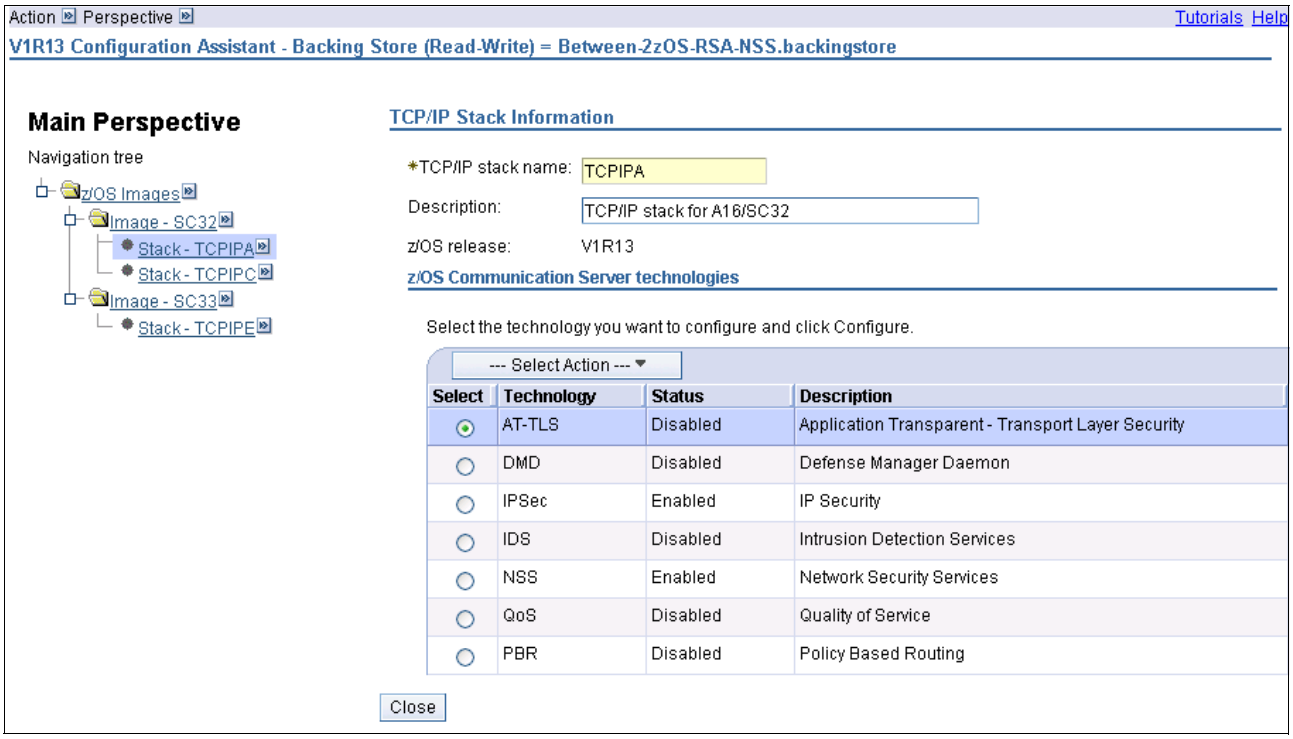


Figure 9-42 No AT-TLS enablement for TCPIPA

When TCPIPE is selected, notice that AT-TLS is not enabled there either. However, for our scenario, you must have AT-TLS for this NSS configuration to work. Therefore, although you defined the NSS portions of the solution, you are still missing the necessary AT-TLS definitions on both TCPIPA and TCPIPE. We explain how to build these in the next section.

Enabling AT-TLS for NSS

To enable AT-TLS for NSS, complete the following steps:

1. In the Main Perspective window, select **TCPIPE**, and then select the AT-TLS technology. Select Action **Enable**, and Select Action **Configure**. Note in Figure 9-43 that the stack shows a status of *Incomplete*. This window displays a list of pre-existing rules that are provided with the IBM Configuration Assistant. Highlight the rule for `Default_NSS_Server`, then click **Select Action** → **Modify**.

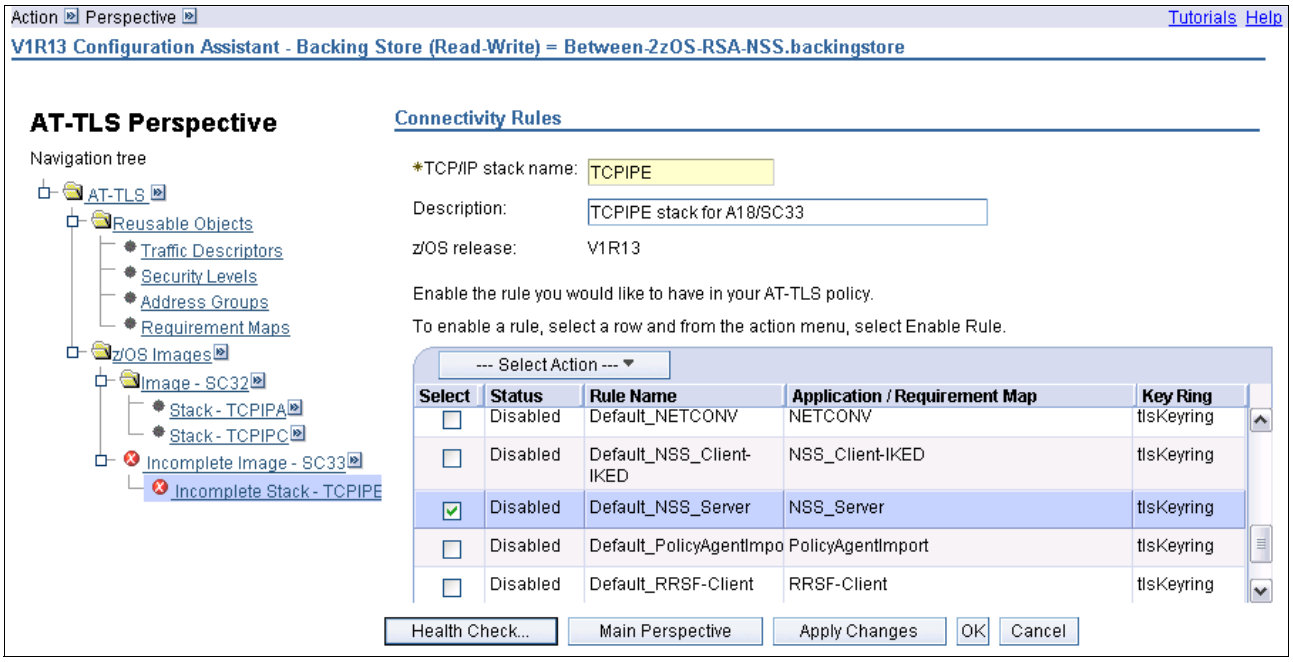


Figure 9-43 AT-TLS Perspective rule list

2. In the Modify Rule window, click **Enable rule**. You do not need to change the Data Endpoints because they default to all IPv4 addresses, which is acceptable unless you need to specify specific IP addresses.

   You need to change the security level from the default for this rule. Click the Security Level tab in the Modify Rule window, and change the security level to AT-TLS_`Silver` as shown in Figure 9-44. Click **OK** to return to the AT-TLS perspective list of rules.

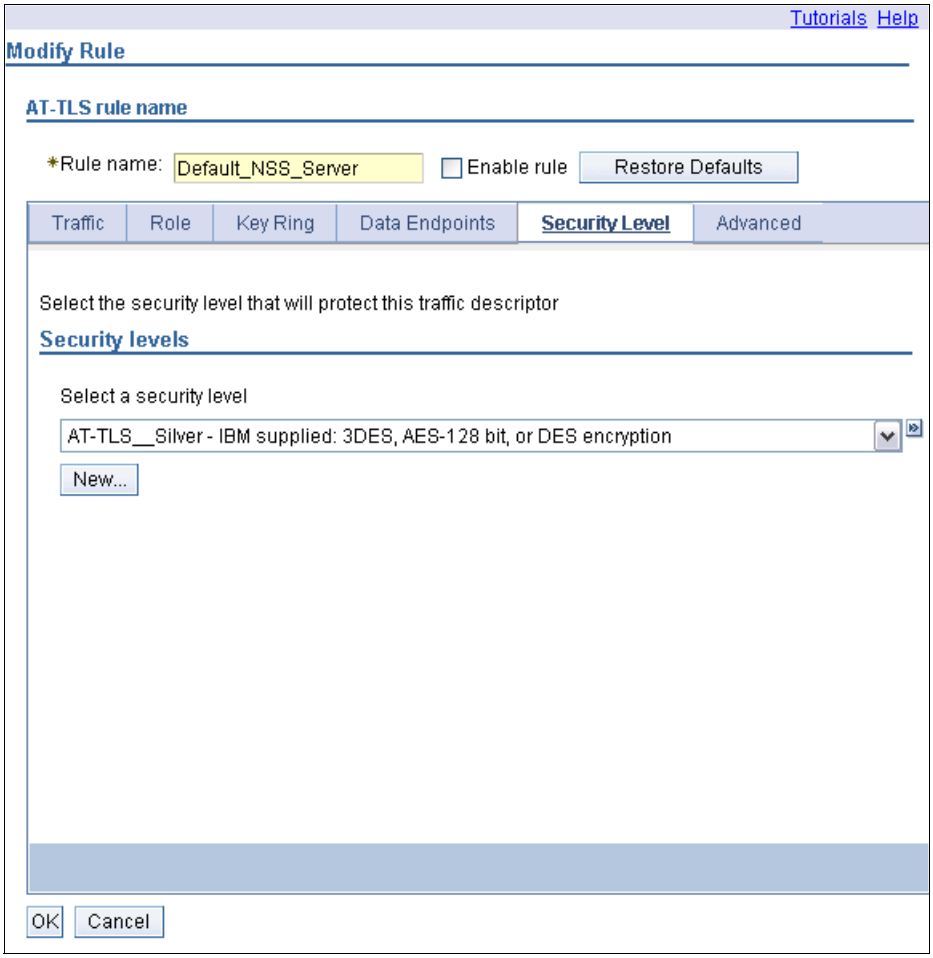


Figure 9-44 Setting the security level

3. Click **Apply Changes**.
4. The image status still shows as `Incomplete` because you have not yet specified the key ring at the image level. To correct this issue, select **Image SC33** in the Navigation tree. When prompted to enter the image key ring, enter the name `TCPIP/SharedRing1` and click **OK**. The AT-TLS configuration for the NSS server on TCPIPE is now complete.

Now, when you select **TCPIPA** from the AT-TLS Perspective window, this stack shows as `Stack Disabled` (Figure 9-45). In the upper right corner of this window, click **Enable AT-TLS**.

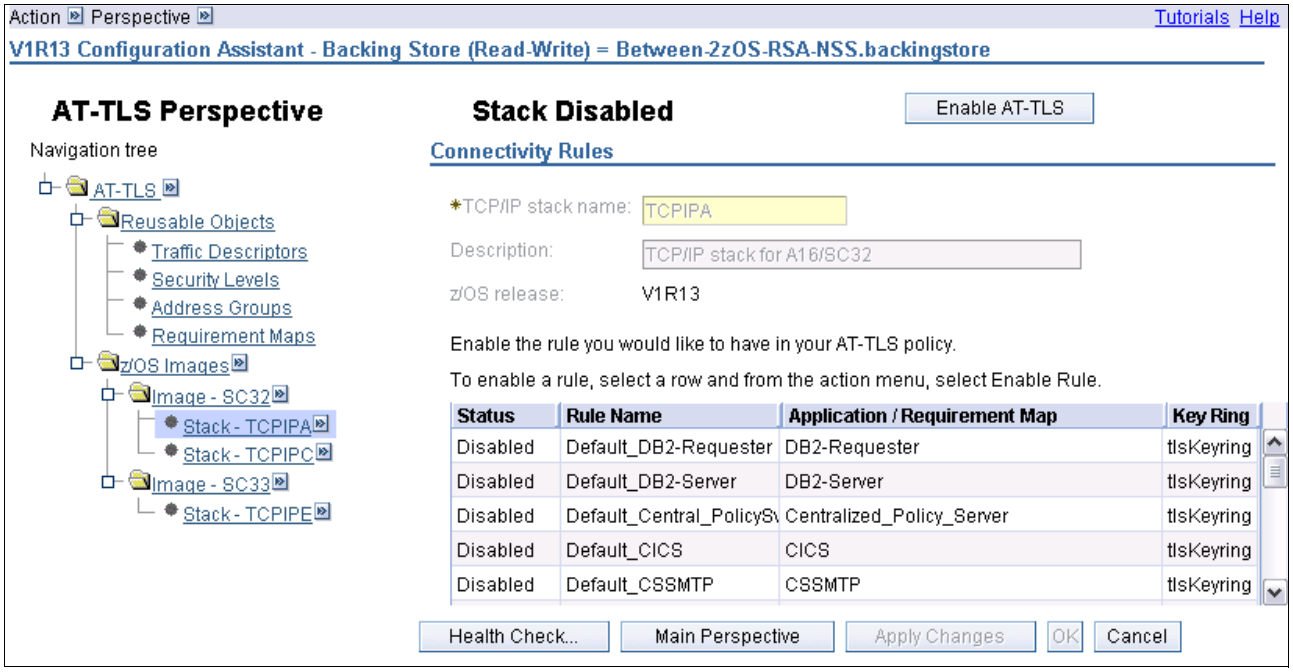


Figure 9-45 SC32 is disabled for AT-TLS

5. Highlight the rule of `Default_NSS_Client-IKED`, and click **Modify**.

6. In the Modify Rule window, click **Enable rule** as shown in Figure 9-46.

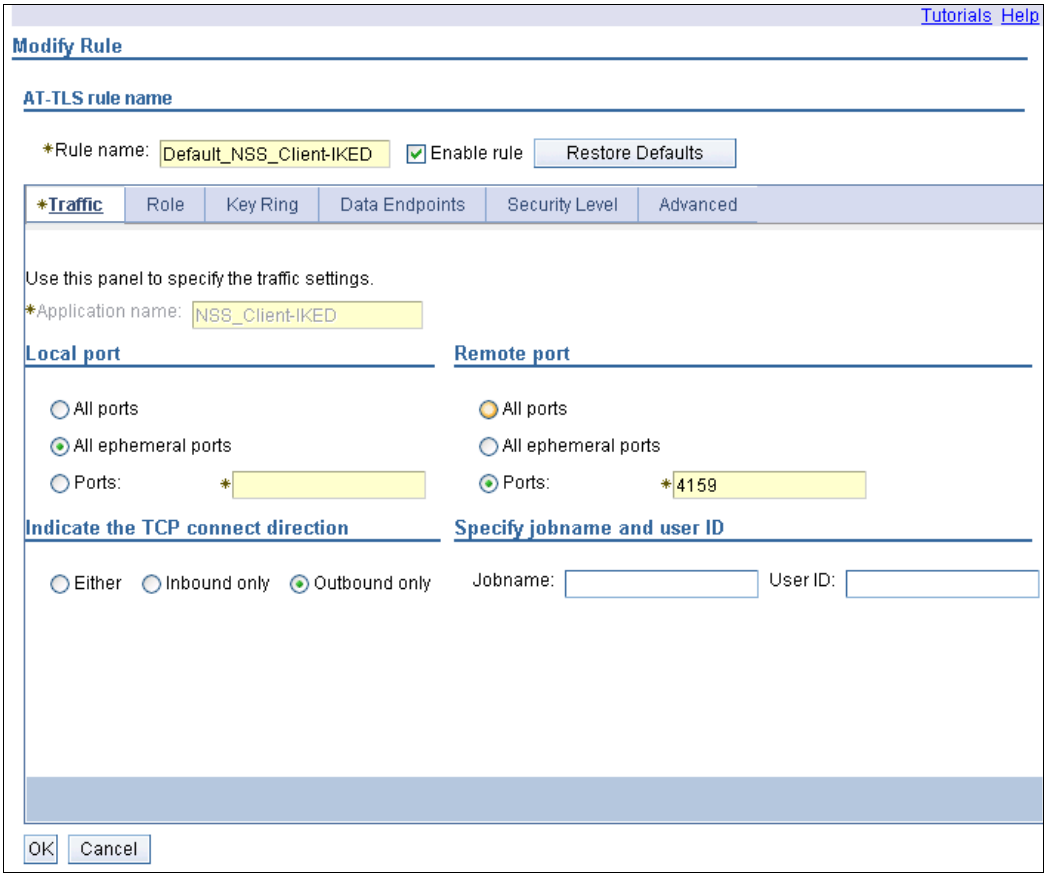


Figure 9-46 Setting the traffic for NSS_Client-IKED

7. Click the Security Level tab and change the security level to `Silver` so that it matches the security level for the server. Click **OK** to return to the AT-TLS Perspective.
8. Click **Apply Changes**.

   Notice that the Image SC32 still shows as `Incomplete`. Again, select this image in the Navigation tree, and when prompted, enter the key ring information. As for the NSS Server, we used the name `TCPIP/SharedRing1`. Click **OK**. The client is now configured.

9. To view the details of the policies that you create, select the desired image name in the Navigation tree. You might need to select the desired perspective from the Perspective pull-down menu, depending on where you are. After you select the image name, a window similar to Figure 9-47 opens.

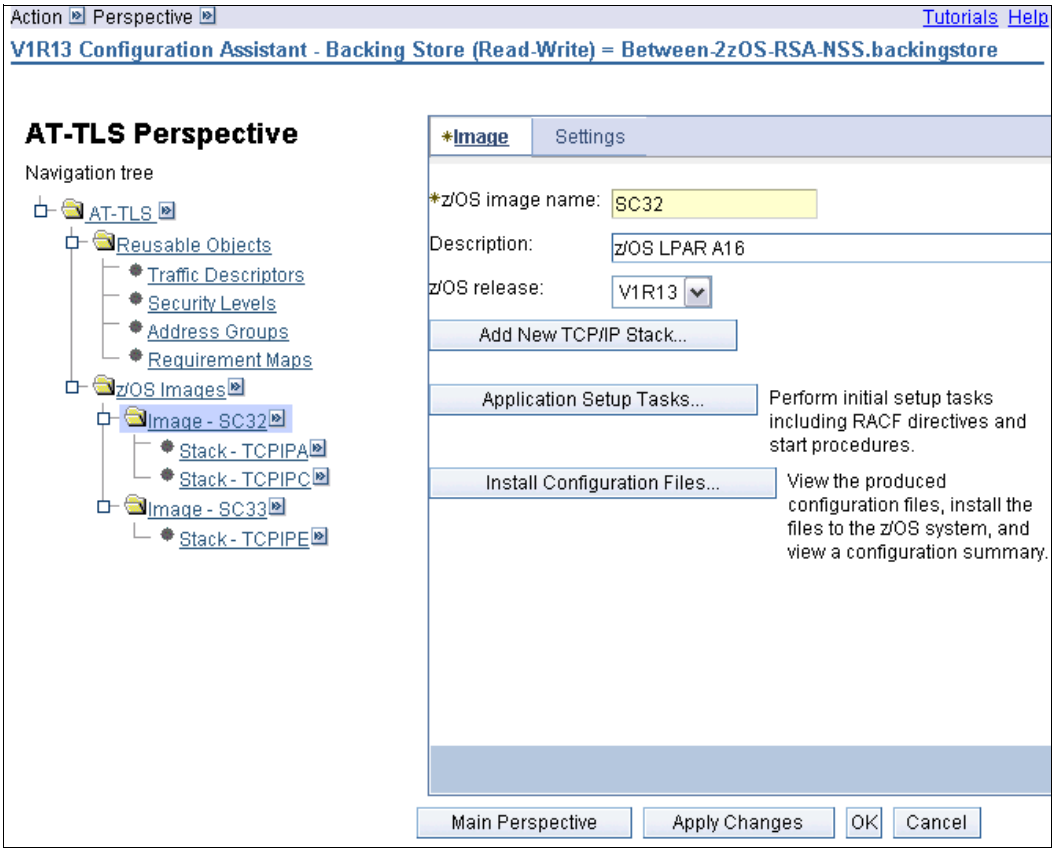


Figure 9-47 AT-TLS Perspective, image view

10. Click **Install Configuration Files**. The window shown in Figure 9-48 is displayed. Highlight the desired technology. Our sample includes only AT-TLS. Click **Select Action** → **Show Configuration File** to view the policies.

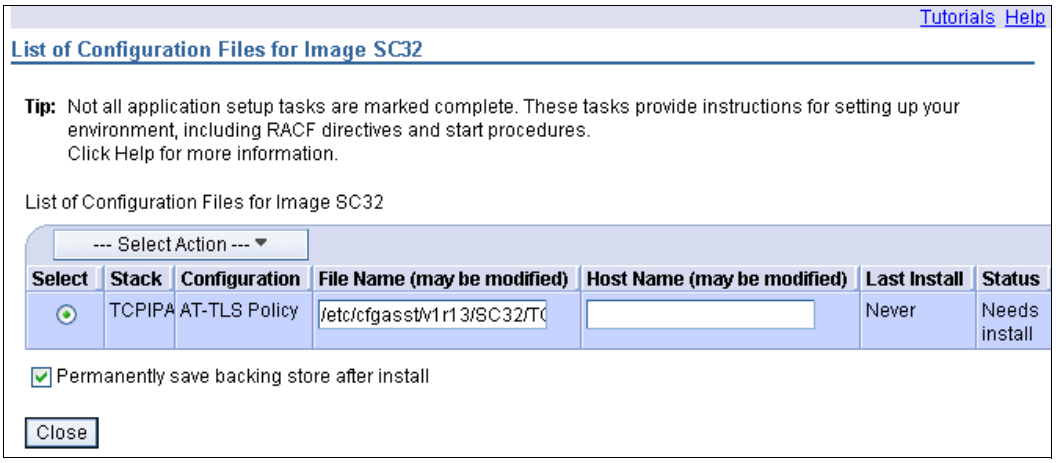


Figure 9-48 Showing your configured policies

The next two figures show the policies that we created. “AT-TLS policy for SC33_TCPIPE (NSS Server)” on page 402 shows an excerpt from the resulting AT-TLS policy file for the NSS server at TCPIPE.

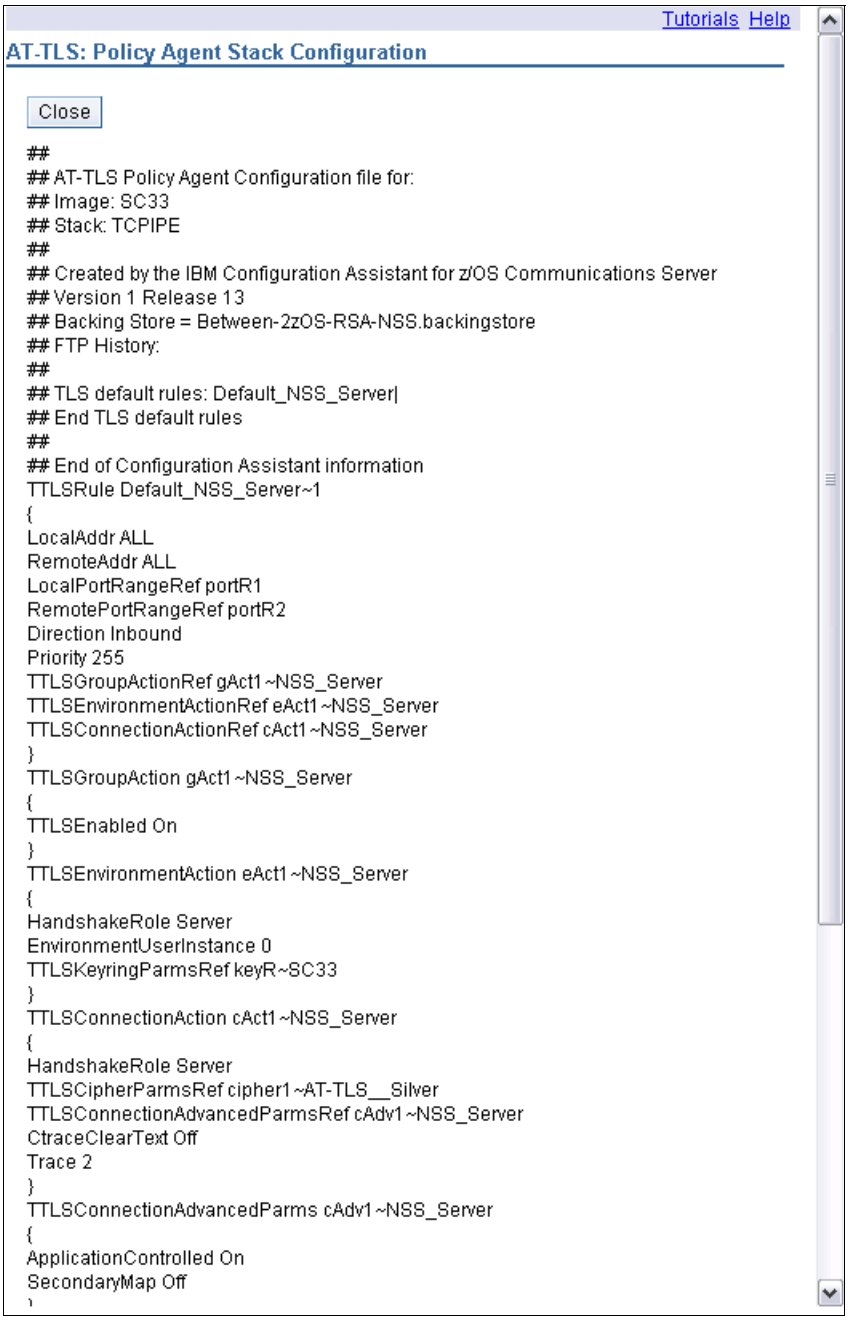


Figure 9-49 AT-TLS policy for SC33_TCPIPE (NSS Server)

Figure 9-50 shows the resulting AT-TLS policy for the NSS client.

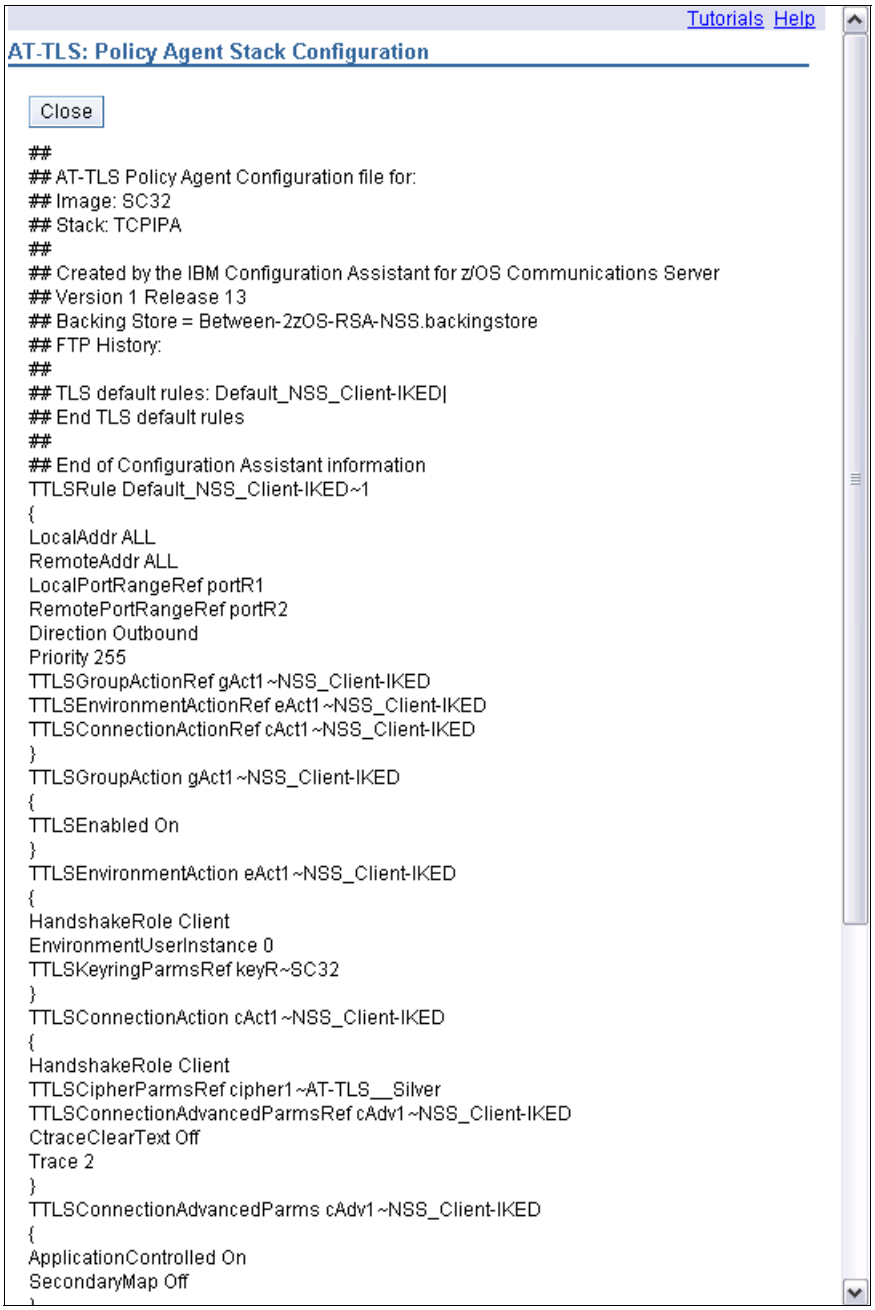


Figure 9-50 AT-TLS policy for NSS client SC32_TCPIPA

At this point, you can 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 discussed in 4.6.2, “Migrating backing store files to z/OSMF Configuration Assistant” on page 153. Then you can use FTP to copy the files with which you want to test to the appropriate nodes, SC32 and SC33.

9.3 Verifying the NSS environment for the IKED Client

This section shows the different areas that you need to verify to ensure that NSS is running properly.

9.3.1 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. We describe these definitions in 9.2.5, “Configuring prerequisites for NSS for an IKED Client” on page 345 and 9.2.6, “Configuring authorizations for NSS” on page 351.

The following files must be available at the NSS *server* node:

- NSSD procedure

  Described in 9.2.7, “Configuring the NSS server for an IKED Client” on page 361.

- NSSD configuration file and environment file

  Described in 9.2.7, “Configuring the NSS server for an IKED Client” on page 361.

- AT-TLS policy for NSS client and server communication

  Described in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

  You reference this policy in the SC33 TCPIPE image file. Example 9-31 shows the main PAGENT configuration file for SC33 in our testing environment.

  Example 9-31 Main PAGENT configuration file at SC33: pagent.sc33.conf

```
Loglevel 15
TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600
TcpImage TCPIPE /etc/pagent.sc33.tcpipe.conf  FLUSH PURGE 600
```

  Example 9-32 shows our TCPIPE image file.

  Example 9-32 PAGENT Image file for TCPIPE at SC33: pagent.sc33.tcpipe.conf

```
IPSecConfig  /etc/pagent.sc33.tcpipe.ipsec.nss.policy
TTLSConfig   /etc/pagent.sc33.tcpipe.ttls.nss.policy
```

- IPSec policy file if the NSS server is also enabled for IPSec

  Described in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

  You reference this policy in the SC33 TCPIPE image file. Example 9-32 shows the file.

The following files must be in place on the NSS *client* node:

- IKED procedure

  Described in 9.2.8, “Enabling an IKED NSS client to use NSS” on page 366.

- IKED configuration file that indicates the TCP/IP stacks that are to take advantage of Network Security Services

  Described in 9.2.8, “Enabling an IKED NSS client to use NSS” on page 366. In our environment, we named our NSS client configuration file `iked.sc32.nss.conf` and placed it in the `/etc` subdirectory of SC32.

  We enabled this configuration file in our environment file for NSSD instead of `iked.sc32.conf`. Example 9-33 on page 405 shows the environment file.

Example 9-33 /SC32/etc/iked.sc32.env for IKED procedure

```
#IKED_FILE=/etc/iked.sc32.conf
IKED_FILE=/etc/iked.sc32.nss.conf
IKED_LOG_FILE=/SC32/tmp/iked.sc32.log
IKED_LOG_FILE_CONTROL=999,3
IKED_CTRACE_MEMBER=CTIIKE00
```

- AT-TLS policy for NSS client and server communication

  Described in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

  You reference this policy in the SC32 TCPIPA image file. Example 9-34 shows the main PAGENT configuration file for SC32 in our environment.

Example 9-34 Main PAGENT configuration file at SC32: pagent.sc32.conf

```
LogLevel 15
TcpImage TCPIPC /etc/pagent.sc32.tcpipc.conf  FLUSH PURGE 600
TcpImage TCPIPA /etc/pagent.sc32.tcpipa.conf  FLUSH PURGE 600
```

Example 9-35 shows our TCPIPA image file.

Example 9-35 PAGENT Image file for TCPIPA at SC32: pagent.sc32.tcpipa.conf

```
IPSecConfig /etc/pagent.sc32.tcpipa.ipsec.nss.policy
TTLSConfig  /etc/pagent.sc32.tcpipa.ttls.nss.policy
```

- IPSec policy file for the communication between the NSS client and the NSS server

  Described in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

  You reference this policy in the SC32 TCPIPA image file. Example 9-35 shows the file.

- (Optional) IPSec policy file for the communication between the non-NSS client and NSS server.

  Described in 9.2.9, “Creating NSS files for an IKED Client with z/OSMF Configuration Assistant” on page 371.

  You reference this policy in the SC32 TCPIPC image file. Example 9-36 shows the file.

Example 9-36 PAGENT Image file for TCPIPC at SC32: pagent.sc32.tcpipc.conf

```
IPSecConfig /etc/pagent.sc32.tcpipc.ipsec.nss.policy
```

9.3.2 Initialize NSSD and the NSS client

In our testing environment, we were able to take down the TCP/IP stacks on MVS system IDs SC33 and SC32. As a result, we initialized all procedures from scratch. If this process is not possible for you, you need to execute the MODIFY REFRESH commands for PAGENT and IKED as we describe in this section.

Note that following steps include descriptions based on our optional scenario with the TCPIPC stack configuration, which is a non-NSS client.

NSS server

First, you need to initialize the NSS server, as described in the following steps:

1. Restart the TCP/IP procedure on SC33 using the following command:

   ```
   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 IKED
   ```

4. Start the NSSD procedure from the MVS console with the following command:

   ```
   S NSSD
   ```

Example 9-37 shows the startup sequence SC33.

Example 9-37 Startup of NSSD at SC33

```
S NSSD
$HASP100 NSSD     ON STCINRDR
IEF695I START NSSD     WITH JOBNAME NSSD     IS ASSIGNED TO USER NSSD
   , GROUP TCPGRP
$HASP373 NSSD     STARTED
EZD1359I NETWORK SECURITY SERVICES (NSS) SERVER RELEASE CS V1R13
SERVICE LEVEL CS100325 CREATED ON Mar 25 2010
EZD1357I NETWORK SECURITY SERVICES (NSS) SERVER INITIALIZATION
SEQUENCE HAS BEGUN
IEE252I MEMBER CTINSS00 FOUND IN SYS1.PARMLIB 1
EZD1373I NSS IPSEC DISCIPLINE ENABLED
EZD1373I NSS XMLAPPLIANCE DISCIPLINE ENABLED
EZD1353I NSS SERVER CONFIG PROCESSING COMPLETE USING FILE /etc/nssd.sc
33.conf
EZD1374E ICSF SERVICES ARE CURRENTLY UNAVAILABLE
EZD1358I NSS SERVER INITIALIZATION SEQUENCE HAS COMPLETED 2
```

In this example, the numbers correspond to the following information:

1. Shows that we remember to move the CTINSS00 member into parmlib: that is, the CTINSS00 member was found at initialization.

2. Shows that the NSS server initialized successfully.

NSS client

Next, you need to initialize the NSS client. Follow these steps:

1. Restart the TCP/IP procedure on SC32 using the following command:

   ```
   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 IKED
```

Example 9-38 shows the startup sequence at SC32.

Example 9-38 Initialization of NSS client: Startup of IKED 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 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 1
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPA
EZD1058I IKE STATUS FOR STACK TCPIPC   IS UP 2
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPC
EZD1133I IKE STATUS FOR STACK TCPIP    IS ACTIVE WITHOUT IPSECURITY
SUPPORT 3
EZD1046I IKE INITIALIZATION COMPLETE
EZD1136I THE IKE DAEMON IS CONNECTED TO THE NSS SERVER AT 192.168.1.40
 PORT 4159 FOR STACK TCPIPA 4
```

In this example, the numbers correspond to the following information:

1. Shows that TCPIPA has been enabled for IKE.

2. Shows that TCPIPC has been enabled for IKE (optional).

3. Shows that TCPIP is not enabled for IP security or for IKE.

4. Shows that the IKE daemon is using the NSS services on behalf of TCPIPA.

9.3.3 NSS and IKE displays on SC33 and SC32

The first step in verifying the NSS environment for the IKED client is to display the configurations on SC33 and SC32.

Display the configurations for NSS

Display the NSSD server on SC33 as shown in Example 9-39.

Example 9-39 Display of NSSD configuration at SC33

```
F NSSD,DISPLAY
EZD1386I DISPLAY NSS CONFIGURATION 687
DISPLAY Network Security Server Configuration Parameters:
    Port        = 4159    1
    SyslogLevel = 255     (0x00ff) 2
```

```
    KeyRing     = "NSSD/NSSD_keyring" 3
    ---------------------------------
    Discipline IPSec        = Enabled 4
    Discipline XMLAppliance = Enabled 4
    ---------------------------------
  IPSec Discipline Configuration Parameters:
    FIPS140     = No
    URLCacheInterval = 10080
    There are 0 CertificateURL and CertificateBundleURL entries
```

In this example, the numbers correspond to the following information:

1. Shows that the NSS server is listening on port 4159.
2. Shows that we have raised the syslog level for NSS to a 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 altered the syslog levels in the files manually as we continued testing.

   You need to reduce this syslog level after you are satisfied with testing so you do not impose a performance burden on the system.
3. Shows that the NSSD server is providing IKE client certificates from the named key ring.
4. Show that the NSS server is currently supporting the IPSec and XMLAppliance disciplines.

Example 9-40 shows the display results of IKED on SC33, where the NSSD server resides.

Example 9-40 Display of IKE process at SC33

```
F IKED,DISPLAY
EZD1158I DISPLAY IKE CONFIGURATION 732
DISPLAY IKE configuration parameters:
  Values loaded from /etc/iked.sc33.conf
  IkeSyslogLevel = 255
  PagentSyslogLevel = 63
  SMF119 = NONE
  Keyring = IKED/IKED33_keyring
  IkeRetries = 10
  IkeInitWait = 30
  FIPS140 = no
  Echo = yes
  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. 1
```

In this example, the number corresponds to the following information:

1. Shows that the IKED procedure on SC33 is itself not an NSS client.

   Although 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 execute the following command.

```
F IKED,DISPLAY
```

Example 9-41 shows the display for this command.

Example 9-41 Displaying configuration of NSS client: IKED on behalf of TCPIPA

```
F IKED,DISPLAY
EZD1158I DISPLAY IKE CONFIGURATION 358
DISPLAY IKE configuration parameters:
  Values loaded from /etc/iked.sc32.nss.conf
  IkeSyslogLevel = 255 1
  PagentSyslogLevel = 255 1
  SMF119 = NONE
  Keyring = IKED/IKED32_keyring 2
  IkeRetries = 10
  IkeInitWait = 30
  FIPS140 = no
  Echo = no
  PagentWait = 0
  NssWaitLimit = 60
  NssWaitRetries = 3
  IKE configuration contains no SupportedCertAuth labels.
  NetworkSecurityServer   = 192.168.1.40 3
                Port      = 4159 4
                Identity = CN=ITSO.IBM.COM,OU=ITSO CS19 Shared SITE,
O=IBM Corporation,C=US 5
  IKE configuration contains no NetworkSecurityServerBackup info.
  NssStackConfig 1: Stack = TCPIPA 6
                    Client name = SC32_TCPIPA
                    Userid name = NSS32A
                    AuthBy = Password
                    Cert Service: Enabled
                    RemoteMgmtService: Enabled
```

In this example, the numbers correspond to the following information:

1. Shows the high logging levels, which we edited into the files after we transferred them from the IBM Configuration Assistant GUI.
2. Shows the key ring that the NSS client is using.
3. Shows the IP address of the NSS server.
4. Shows the port number that the NSS client must connect to for communication with the NSS server.
5. 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.
6. Shows the NSS configuration parameters that we coded in the IKE configuration for TCPIPA. Notice 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 you complete testing that you reduce the logging levels on PAGENT, IKE, and the NSS server.

Next, determine if the appropriate connections are established at the client and server nodes.

AT-TLS connection

Use the **netstat** command to determine if you have AT-TLS connections between the client and server as shown in Example 9-42.

Example 9-42 TLS at SC33, the NSS server node

```
D TCPIP,TCPIPE,N,TTLS
TTLSGRPACTION                          GROUP ID          CONNS
GACT1~NSS_SERVER                       00000003              1 1
NSS~TLS~ON                             00000004              0
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the number corresponds to the following information:

1. Shows that we have one connection using the AT-TLS action for the NSS server.

Display the connections at the server with the **netstat** command as shown in Example 9-43.

Example 9-43 Netstat connection at the server site for client NSSD

```
D TCPIP,TCPIPE,N,CONN,CLIENT=NSSD
USER ID  CONN     STATE
NSSD     00004095 ESTBLSH      1
  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
```

In this example, the number **1** shows that a connection (# 4095) is established with `10.1.1.50` from the listening port of `4159`. To determine if this is a TLS connection, you can run the commands shown in Example 9-44.

Example 9-44 TLS connection detail for server

```
D TCPIP,TCPIPE,N,TTLS,CONN=4095,DETAIL
CONNID: 00004095
  JOBNAME:      NSSD
  LOCALSOCKET:  ::FFFF:192.168.1.40..4159
  REMOTESOCKET: ::FFFF:10.1.1.50..10029
  SECLEVEL:     TLS VERSION 1 1
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA 2
  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 3
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~NSS_SERVER 4
    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
```

In this example, the numbers correspond to the following information:

1. Shows that the connection was established with TLS V1 security.

2. Shows the negotiated cipher suite of DES.

3. Shows the key ring used for client authentication.

4. 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 9-45.

Example 9-45 pasearch -t at the NSS server

```
TCP/IP pasearch CS 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   1
    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 2
   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 key rings 1 and certificate labels 2 that you might be having problems with because of faulty RACF definitions and configuration files.

At the client node, display the **netstat** output as shown in Example 9-46.

Example 9-46 TTLS activity at TCPIPA

```
D TCPIP,TCPIPA,N,TTLS
TTLSGRPACTION                              GROUP ID           CONNS
 GACT1~NSS_CLIENT                           00000003               1 1
 GACT1~NSS_CLIENT                  (STALE)  00000001               3 2
 NSS~TLS~ON                                 00000004               0
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

In this example, the numbers correspond to the following information:

1. Shows that we have one connection using the AT-TLS action for the NSS client.
2. Shows that we have old, stale connections that are remembered from previous attempts. Note that you might not always see stale entries, of course.

To examine this connection in more detail, issue a generic **netstat** command to view connections as shown in Example 9-47.

Example 9-47 Connections at TCPIPA for IKED

```
D TCPIP,TCPIPA,N,CONN,CLIENT=IKEDC
USER ID  CONN     STATE
IKEDC    0002063A ESTBLSH       1
  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
```

In this example, the number corresponds to the following information:

1. Shows that we established a connection (# 2063A) with 192.168.1.40 and its port 4159, the NSS listening port.

To determine if this is a TLS connection, run the commands shown in Example 9-48.

Example 9-48 The NSS TLS connection

```
D TCPIP,TCPIPA,N,TTLS,CONN=2063A,DETAIL
CONNID: 0002063A
  JOBNAME:      IKEDC
  LOCALSOCKET:  10.1.1.50..10029
  REMOTESOCKET: 192.168.1.40..4159
  SECLEVEL:     TLS VERSION 1    1
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA 2
  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 3
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
```

```
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~NSS_CLIENT
    HANDSHAKEROLE:              CLIENT
    V3CIPHERSUITES: 4           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
```

In this example, the numbers correspond to the following information:

1. Shows that the connection was established with TLS V1 security.
2. Shows the negotiated cipher suite of DES.
3. Shows the key ring used for client authentication.
4. 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

To display the TLS connection between the client and server, use the **ipsec** command as shown in Example 9-49.

Example 9-49 Display known NSS clients from perspective of NSS server

```
CSO2 @ SC33:/u/cs02>ipsec -x display      1

CS ipsec  NSS Client Name: n/a  Wed Dec 17 15:58:02 2008
Primary:  NSS Server      Function: Display            Format:   Detail
Source:   Server          Scope:    n/a                TotAvail: 1
SystemName: SC33

ClientName:                     SC32_TCPIPA
ClientAPIVersion:               2
StackName:                      TCPIPA
SystemName:                     SC32
ClientIPAddress:                ::ffff:10.1.2.51
ClientPort:                     10243
ServerIPAddress:                ::ffff:192.168.1.40
ServerPort:                     4159
UserID:                         NSS32A
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
ConnectState:                   connected
TimeConnected:                  2008/12/17 15:51:50
TimeOfLastMessageFromClient:    2008/12/17 15:51:50
**********************************************************************

1 entries selected
CSO2 @ SC33:/u/cs02>
```

In this example, the number corresponds to the following information:

1. 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, as shown in Example 9-50.

Example 9-50 NSS client short display at NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display -r short

CS ipsec  NSS Client Name: n/a  Wed Dec 17 16:00:38 2008
Primary:  NSS Server      Function: Display            Format:   Short
Source:   Server          Scope:    n/a                TotAvail: 1
SystemName: SC33

ClientName               |ClientAPIVersion|StackName|SystemName
                         |ClientIPAddress|ClientPort|ServerIPAddress|ServerPort
                         |UserID
                         |RemoteManagementSelected|RemoteManagementEnabled
                         |CertificateServicesSelected|CertificateServicesEnabled
                         |ConnectState
                         |TimeConnected|TimeOfLastMessageFromClient

SC32_TCPIPA              |2|TCPIPA|SC32
                         |::ffff:10.1.2.51|10243|::ffff:192.168.1.40|4159
                         |NSS32A
                         |Yes|Yes
                         |Yes|Yes
                         |Yes|Yes
                         |connected
                         |2008/12/17 15:51:50|2008/12/17 15:51:50


1 entries selected
```

Another way to view the same information is by using the **-r wide** option, as shown in Example 9-51.

Example 9-51 Wide option for ipsec command from NSS server

```
CSO2 @ SC33:/u/csO2>ipsec -x display -r wide

CS ipsec  NSS Client Name: n/a  Wed Dec 17 16:03:17 2008
Primary:  NSS Server      Function: Display            Format:   Wide
Source:   Server          Scope:    n/a                TotAvail: 1
SystemName: SC33

ClientName|ClientAPIVersion|StackName|SystemName|ClientIPAddress|ClientPort|Serv
erIPAddress|ServerPort|UserID|RemoteManagementSelected|RemoteManagementEnabled|C
ertificateServicesSelected|CertificateServicesEnabled|ConnectState|TimeConnected
|TimeOfLastMessageFromClient

SC32_TCPIPA|2|TCPIPA|SC32|::ffff:10.1.2.51|10243|::ffff:192.168.1.40|4159|NSS32A
|Yes|Yes|Yes|Yes|connected|2008/12/17 15:51:50|2008/12/17 15:51:50

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 shown in Example 9-52.

Example 9-52 Information about a single NSS client displayed at NSS server

```
CSO2 @ SC33:/u/cs02>ipsec -x display -z SC32_TCPIPA

CS ipsec  NSS Client Name: SC32_TCPIPA  Wed Dec 17 16:04:09 2008
Primary:  NSS Server      Function: Display            Format:   Detail
Source:   Server          Scope:    n/a                TotAvail: 1
SystemName: SC33

ClientName:                     SC32_TCPIPA
ClientAPIVersion:               2
StackName:                      TCPIPA
SystemName:                     SC32
ClientIPAddress:                ::ffff:10.1.2.51
ClientPort:                     10243
ServerIPAddress:                ::ffff:192.168.1.40
ServerPort:                     4159
UserID:                         NSS32A
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
CertificateServicesEnabled:     Yes
ConnectState:                   connected
TimeConnected:                  2008/12/17 15:51:50
TimeOfLastMessageFromClient:    2008/12/17 15:51:50
*********************************************************************

1 entries selected
```

Next, you can move to the client system (SC32) and issue **ipsec**, which displays the data shown in Example 9-53.

Example 9-53 Display at NSS client: only one of three stacks is NSS-enabled

```
CSO2 @ SC32:/u/cs02>ipsec -w display
CS ipsec  NSS Client Name: n/a  Wed Dec 17 16:05:48 2008
Primary:  Stack NSS       Function: Display            Format:   Detail
Source:   IKED            Scope:    n/a                TotAvail: 3
SystemName: SC32

StackName:                      TCPIP
ClientName:                     n/a
ClientAPIVersion:               n/a
ServerAPIVersion:               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    1
ClientName:                     SC32_TCPIPA
ClientAPIVersion:               2
ServerAPIVersion:               2
NSServicesSupported:            Yes
RemoteManagementSelected:       Yes
RemoteManagementEnabled:        Yes
CertificateServicesSelected:    Yes
CertificateServicesEnabled:     Yes
NSClientIPAddress:              10.1.2.51
NSClientPort:                   10243
NSServerIPAddress:              192.168.1.40
NSServerPort:                   4159
NSServerSystemName:             SC33
UserID:                         NSS32A
ConnectionState:                connected
TimeConnectedToNSServer:        2008/12/17 15:51:50
TimeOfLastMessageToNSServer:    2008/12/17 15:51:50
*********************************************************************
StackName:                      TCPIPC
ClientName:                     n/a
ClientAPIVersion:               n/a
ServerAPIVersion:               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
```

In this example, the number corresponds to the following information:

1. Shows that TCPIPA is the only NSS-enabled stack on SC32.

At this point, you know that there is a valid connection between the TLS and the NSS server and client. You have not seen evidence, however, that any connections using IPSec were established with TLS services as described in the next section.

IPSec tunnels with NSS services

The last part of the verification of NSS services requires that you establish IPSec tunnels and confirm that they used the NSS server. First, display the client side as shown in Example 9-54.

Example 9-54 Are there any IPSEC tunnels with endpoints on SC32?

```
CSO2 @ SC32:/u/csO2>ipsec -p TCPIPA -y display

CS 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 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 9-54 shows that neither the TCPIPA nor the TCPIPC stack has a connection established with IP security.

Figure 9-51 shows the output that includes an IKE tunnel between TCPIPC and TCPIPE (using IKE in pre-shared key mode) and an IKE tunnel established between TCPIPA and TCPIPE (using NSS certificate services for TCPIPA). The IKE tunnel between TCPIPA and TCPIPE used digital signature authentication.

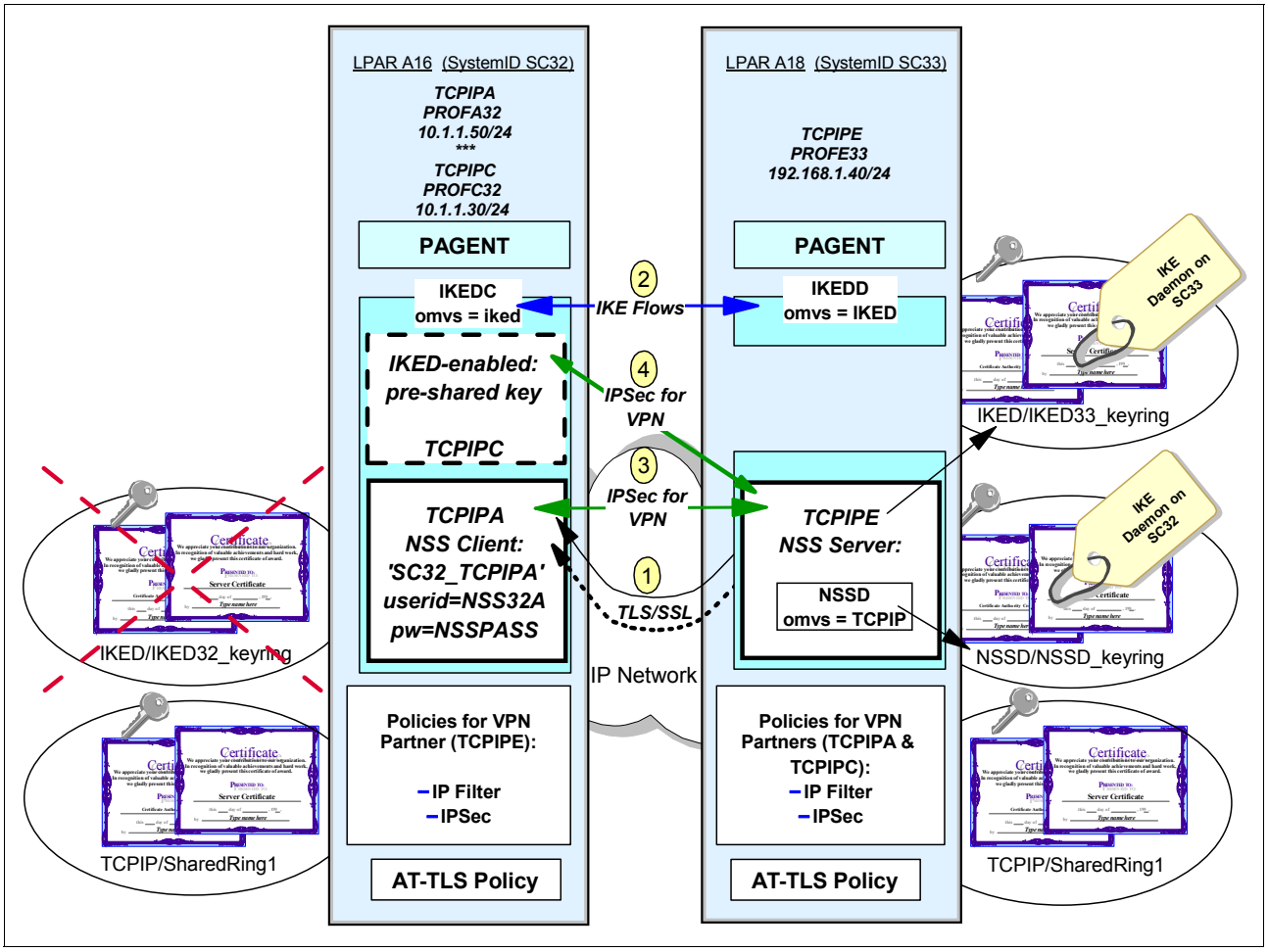


Figure 9-51 TCPIPA is NSS-enabled and TCPIPC is not NSS-enabled

In Figure 9-51, we use IKED 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 9-51, 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.

Tip: Because of 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 work equally well if we had created a unique IKE key ring per MVS image. Recall the earlier discussion of key ring strategies in "Key rings for the NSS implementation" on page 332.

Figure 9-51 on page 419 shows the following important activities:

1. Represents the secure TLS connection over which IKED (SC32) communicates with the NSS server on behalf of TCPIPA. This connection is also the only one in which NSSD is directly involved.
2. Represents the secure IKE tunnel over which IKED (SC32) and IKED (SC33) negotiate Security Associations.
3. Represents the secure IPSec connection over which stacks TCPIPC and TCPIPE communicate.
4. 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 turn on an IPSec tunnel without NSS services as shown in Example 9-55.

Example 9-55 Establish IPSEC tunnel; display the tunnel between TCPIPC and TCPIPE

```
CSO2 @ SC32:/u/csO2>ping -p TCPIPC -s 10.1.1.30 192.168.1.40      1
CS: Pinging host 192.168.1.40
sendto(): EDC5111I Permission denied. (errno2=0x74420291) 2

CSO2 @ SC32:/u/csO2>ping -p TCPIPC -s 10.1.1.30 192.168.1.40 3
CS: Pinging host 192.168.1.40
Ping #1 response took 0.001 seconds.


CSO2 @ SC32:/u/csO2>ipsec -p TCPIPC -y display 4

CS ipsec  Stack Name: TCPIPC Wed Dec 17 16:18:04 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y2 5
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:              3596169681
 AuthOutboundSpi:             1676713295
HowToEncrypt:                 DES
 EncryptInboundSpi:           3596169681
 EncryptOutboundSpi:          1676713295
Protocol:                     ALL(0)
LocalPort:                    0
```

```
RemotePort:                   0
OutboundPackets:              1
OutboundBytes:                264
InboundPackets:               1
InboundBytes:                 264
Lifesize:                     0K
LifesizeRefresh:              0K
CurrentByteCount:             0b
LifetimeRefresh:              2008/12/17 19:14:26
LifetimeExpires:              2008/12/17 20:17:45
CurrentTime:                  2008/12/17 16:18:04
VPNLifeExpires:               2008/12/18 16:17:45
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
PassthroughDF:                n/a
PassthroughDSCP:              n/a

**********************************************************************

1 entries selected
```

The first time that we ping TCPIPE (1 and 2), the connection does not succeed. See the following note for an explanation. The subsequent ping 3 succeeds and brings up a tunnel. 4 shows the command we use to display the dynamic tunnel, and 5 shows the tunnel ID.

Why the connection fails at the first ping command: Our IPSec policy has been specified with OnDemand. ICMP commands, such as **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 turn on 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 might 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 digital
signature mode of authentication for stack TCPIPE using the local keyring.
```

Next, we open a connection between TCPIPA, which is an NSS client, and TCPIPE as shown in Example 9-56.

Example 9-56 Display the tunnel between TCPIPA and TCPIPE

```
CSO2 @ SC32:/u/csO2>ping -p TCPIPA -s 10.1.1.50 192.168.1.40  1
CS: Pinging host 192.168.1.40
Ping #1 response took 0.001 seconds.

CSO2 @ SC32:/u/csO2>ipsec -p TCPIPA -y display 2

CS ipsec  Stack Name: TCPIPA  Wed Dec 17 18:23:58 2008
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y4 3
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:              3307626159
 AuthOutboundSpi:             2788841584
HowToEncrypt:                 DES
 EncryptInboundSpi:           3307626159
 EncryptOutboundSpi:          2788841584
Protocol:                     ALL(0)
LocalPort:                    0
RemotePort:                   0
OutboundPackets:              1
OutboundBytes:                264
InboundPackets:               1
InboundBytes:                 264
Lifesize:                     OK
LifesizeRefresh:              OK
CurrentByteCount:             Ob
LifetimeRefresh:              2008/12/17 21:19:13
LifetimeExpires:              2008/12/17 22:23:13
CurrentTime:                  2008/12/17 18:23:58
VPNLifeExpires:               2008/12/18 18:23:13
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
PassthroughDF:                n/a
PassthroughDSCP:              n/a
************************************************************************

1 entries selected
************************************************************************
```

In this example, the numbers correspond to the following information:

1. We ping TCPIPE from TCPIPA and are successful.
2. Shows the command we use to display the dynamic tunnel.
3. Shows the tunnel ID for the tunnel between TCPIPA and TCPIPE.

However, the information displayed in Example 9-56 on page 422 does not show that the tunnel was established because of Network Security Services. We can determine this information by reviewing NSSD and IKED log entries as discussed in the next section.

9.4 Diagnosing the NSSD environment

This section contains resources, guidance, and examples for diagnosis in the NSSD environment.

9.4.1 Resources and guidance

The following resources are most useful when diagnosing issues in an NSS implementation:

- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781

A useful step is 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 issues that you might run into. The most important messages for diagnosing problems are:

- IKE daemon messages
- TRMD messages
- PAGENT messages
- NSS daemon messages

Your syslog daemon implementation can place the pertinent messages in separate logs or in one large log. However, many of the messages you might need for advanced diagnosis are only available if you raise the logging levels.

Tip: 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 can edit the PAGENT, IKED, and NSSD configuration files manually 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.

9.4.2 Examples of logging information for diagnosis

In this section, we discuss examples of logging information that you can use to diagnose problems. Example 9-57 displays messages that you might receive when you cannot obtain a TLS connection between the NSS client and the NSS server.

Example 9-57 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 9-58 shows the SERVAUTH checking that is done when the NSS client attempts to initiate a connection.

Example 9-58 SERVAUTH checking at NSS server

```
NSSD: DBG0032I NSS_VERBOSE ServauthCheck(NSS32A
,EZB.NSS.SC33.SC32_TCPIPA.IPSEC.CERT) 1 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(CS02
2    ,EZB.NETMGMT.SC33.SC33.NSS.DISPLAY) rc 0 (AUTH (cached)) racfRC 0 racfRsn 0
```

1. 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 9-59 displays the TLS processing when the client requests NSS services.

Example 9-59 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 CS02     - 1 connections
open
IKEDC    IKE: EZD1123I NMI connection from user CS02     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 9-60.

Example 9-60 NSSD messages in syslog (local4.)*

```
EZD1359I Network security services (NSS) server Release CS Service Level
CSCS070402 created on Apr  2
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 1
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 2
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 3
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 9-60 that look alarming, many of these certificates are not important to our implementation. Note the following information:

1. This certificate is the IKE certificate for our client. It is not important 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.
2. This certificate has signed our client's IKE certificate. It is important that this is a CA certificate.
3. The IKE key ring named is the one we expected.

9.5 Worksheet questions for NSSD implementation (IKED Client)

This section provides a list of worksheet questions that you can copy and complete for use with your own NSS implementation. Before you complete the worksheet questions, it is helpful to create a simple network diagram similar to the diagram that we created for our NSS implementation as shown in Figure 9-11 on page 343. Developing a logical network diagram can assist you in your implementation.

Use the following worksheet questions for your own NSS implementation:

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon?

2. What is the superuser user ID associated with the OMVS segment under which the NSS Daemon is to run?

3. With which TCP/IP stack or stacks does the NSS daemon establish affinity?

4. What is the address or the DNS name that the NSS client will use to connect to the NSS server?

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?
   - 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?

7. What “client ID” do you want to assign to each TCP/IP stack that is to request authentication of the NSS server?

8. What type of authentication should our client user ID present: a password or a passticket?
   - Authentication Type:
   - Value if applicable:
9. What is the user ID associated with the OMVS Segment of the client’s IKED procedure?

10. What is the name of the key ring that NSS should use to manage all of the clients’ private keys and certificates?

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).`**

    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?

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?

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?

15. What are the IP addresses of the NSS server and the NSS client for which you might have to establish IP filters on the NSS client node?
    - 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 might have to establish IP filters on the NSS server node?
    - Client:
    - Server:

17. What are the TSO user IDs of the IPSec administrators who are authorized to manage the VPN?

9.6 Additional information

For additional information about Network Security Services, see the following resources:

- *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

10

Network Security Services for WebSphere DataPower appliances

In this chapter, we explain how you can use Network Security Services (NSS) to centralize the security authorizations for middleware appliances such as IBM WebSphere® DataPower devices. The Network Security Services Daemon (NSSD) can be the server to a middleware appliance for the Security Authorization Facility (SAF) authorization process.

We discuss the following topics in this chapter.

Section	Topic
10.1, “Basic concepts” on page 430	Basic concepts behind a NSS implementation
10.2, “Configuring NSS” on page 437	The steps needed to configure the NSS environment on z/OS, on the middleware appliance, and on a Web Services Requester platform
10.3, “Verifying the NSS configuration with the NSS Client (XML Appliance Discipline)” on page 516	How to test your configuration
10.4, “Additional information” on page 532	Conclusions drawn from our work with NSS and a DataPower appliance, and a listing of reference materials to consult
10.5, “NSS configuration worksheet for an NSS XMLAppliance client” on page 533	A blank worksheet to help configure the NSS environment for an NSS Client of the XML Appliance Discipline

10.1 Basic concepts

To understand NSS, you must first understand the disciplines that it can serve, which are illustrated in Figure 10-1.

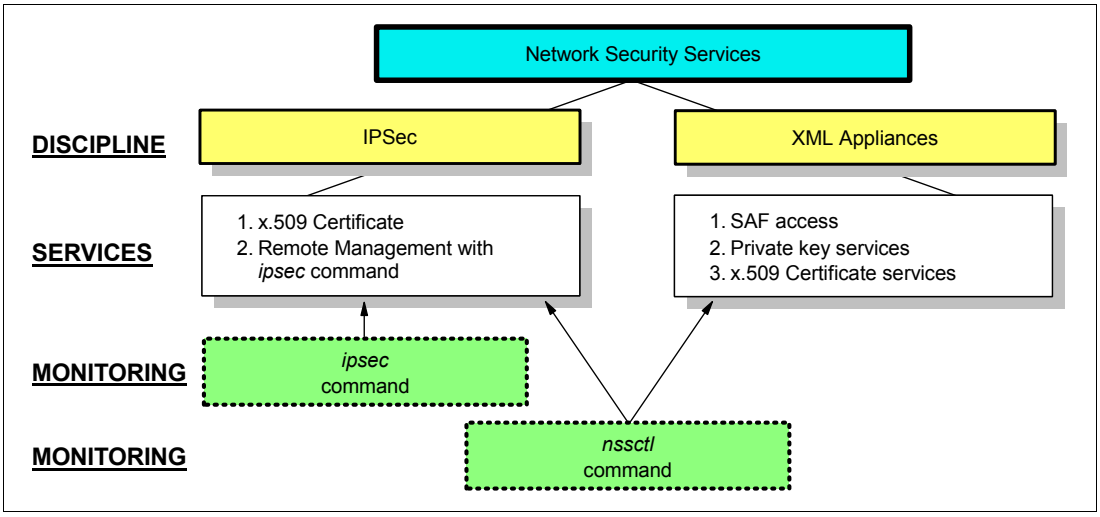


Figure 10-1 Network Security Services disciplines

In z/OS V1R9, NSS was introduced to support the IPSec discipline when IPSec is implemented with the Internet Key Exchange (IKE) protocol using RSA signature mode authentication. The role of the NSS in IPSec is to centralize the certificate and private key operations for NSS IPSec clients running the IKE daemon (IKED), and to centralize the control and display of security information for remote NSS IPSec clients through the execution of a locally run **ipsec** command.

In z/OS V1R10 the role of NSS was expanded to serve the XMLAppliance discipline. In this role, NSS provides SAF authorization for DataPower appliances as NSS XMLAppliance clients. In z/OS Communications Server today, the XML Appliance discipline provides two additional services:

- Access to the private security key that the appliance requires for certain security exchanges and protocols.
- Retrieval of the x.509 digital certificates that are used to implement a secure connection between the WebSphere DataPower appliance and its users.

Each WebSphere DataPower appliance can be configured to act as one or more NSS XMLAppliance clients to use the NSS XMLAppliance services. If configured to do so, traffic traveling through each DataPower appliance can cause the client to issue requests to the NSS server to exploit the services within the XMLAppliance discipline.

The **ipsec** command monitors the status of NSS IPSec clients. The **nssctl** command monitors the status of all NSS clients.

This chapter discusses DataPower only as an NSS client. It assumes that you know how to implement a DataPower appliance. All product documentation is available at the DataPower product support page:

http://www.ibm.com/software/integration/datapower/support/

For other information about DataPower implementation, consult the following resources:

- *IBM WebSphere DataPower SOA Appliances Part I: Overview and Getting Started*, REDP-4327
- *IBM WebSphere DataPower SOA Appliances Part II: Authentication and Authorization*, REDP-4364
- *IBM WebSphere DataPower SOA Appliances Part III: XML Security Guide*, REDP-4365
- *IBM WebSphere DataPower SOA Appliances Part IV: Management and Governance*, REDP-4366
- *DataPower Architectural Design Patterns: Integrating and Securing Services Across Domains*, SG24-7620

Capacity: The NSS server can support a maximum of 500 NSS client connections belonging to both disciplines (IPSEC and XMLAppliance) concurrently. The Network Management Interface (NMI) of the NSS can support up to 10 concurrent connections.

10.1.1 NSS benefits

When interfacing with WebSphere DataPower appliances, NSS centralizes the authentication and authorization of middleware appliances and of web services users at a z/OS node that resides in a single secure zone. It also centralizes the storage of private security keys and digital certificates. Thus, the control of the user and security mechanisms for a DataPower appliance becomes centrally manageable and auditable from z/OS. Although the NSS XMLAppliance services are open interfaces and can be exploited by any XMLAppliance, IBM WebSphere DataPower SOA appliances are the only known exploiter of this function.

10.1.2 Review of DataPower

Middleware appliances, such as IBM WebSphere DataPower Integration XI50 and IBM WebSphere DataPower XML Security Gateway XS40, secure and accelerate the parsing and transformation of many types of messages, including XML messages.

One use case for a DataPower appliance might be to modernize access to existing applications, such as COBOL programs that are interoperating with CICS. For example, you might take XML messages and transform them into COBOL copy books that can be sent to IBM MQSeries® on System z. From MQ, the COBOL can then be passed on to the existing applications. Users in the network then access the information and processing available to those COBOL applications through a CICS task. Figure 10-2 shows an example of this process.

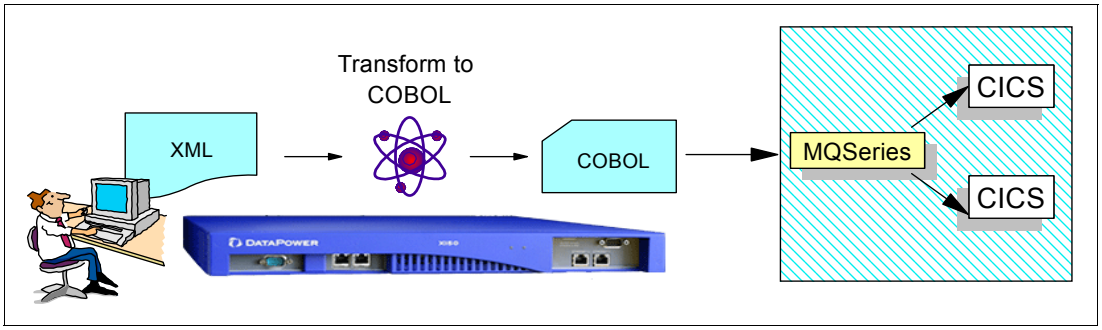


Figure 10-2 DataPower Message Transformation: XML to COBOL

The costs that are associated with parsing and transforming XML messages are high. As organizations adopt service-oriented architectures (SOAs) and as traffic volumes grow, these messages are frequently employed. In addition, it might be beneficial to offload secure operations, such as encryption and decryption, to a middleware appliance. These developments have increased demand for middleware appliances such as DataPower to perform specialized processing on behalf of larger enterprise systems such as IBM System z.

Middleware appliances similar to DataPower can also perform security tasks, such as using LDAP or IBM Tivoli® Access Manager to authenticate users and to authorize users to access services and resources. However, performing this type of security on an additional network platform in an enterprise means adding to a proliferation of security focal points and an increase in administration costs. With an interest in centralizing security services to reduce complexity and administrative overhead, organizations want to exploit existing security services on behalf of middleware appliance users. As a result, middleware appliances need to integrate with z/OS and its security offerings and particularly with a Security Access Facility such as RACF.

The z/OS NSS server, a logical extension to the z/OS security environment, is designed as a centralized server that can provide SAF security services to a number of clients. It is, therefore, positioned to provide security services to the middleware appliances themselves. It does so through an NSS discipline called the *XMLAppliance discipline*.

10.1.3 The NSS solution for XMLAppliance Clients: SAF service

This discipline includes the XMLAppliance SAF access service to the NSS server. The SAF access service provides the capability to invoke a selected set of SAF functions on behalf of its clients. Essentially, NSS becomes the conduit for making remote SAF calls.

A DataPower appliance connected to the NSS server issues inline calls to the NSS server to authenticate users and to check whether users are authorized to perform certain actions on the appliance.

Figure 10-3 shows the sequence of events when the z/OS NSS Server provides centralized services for the security authorizations of the DataPower appliances.

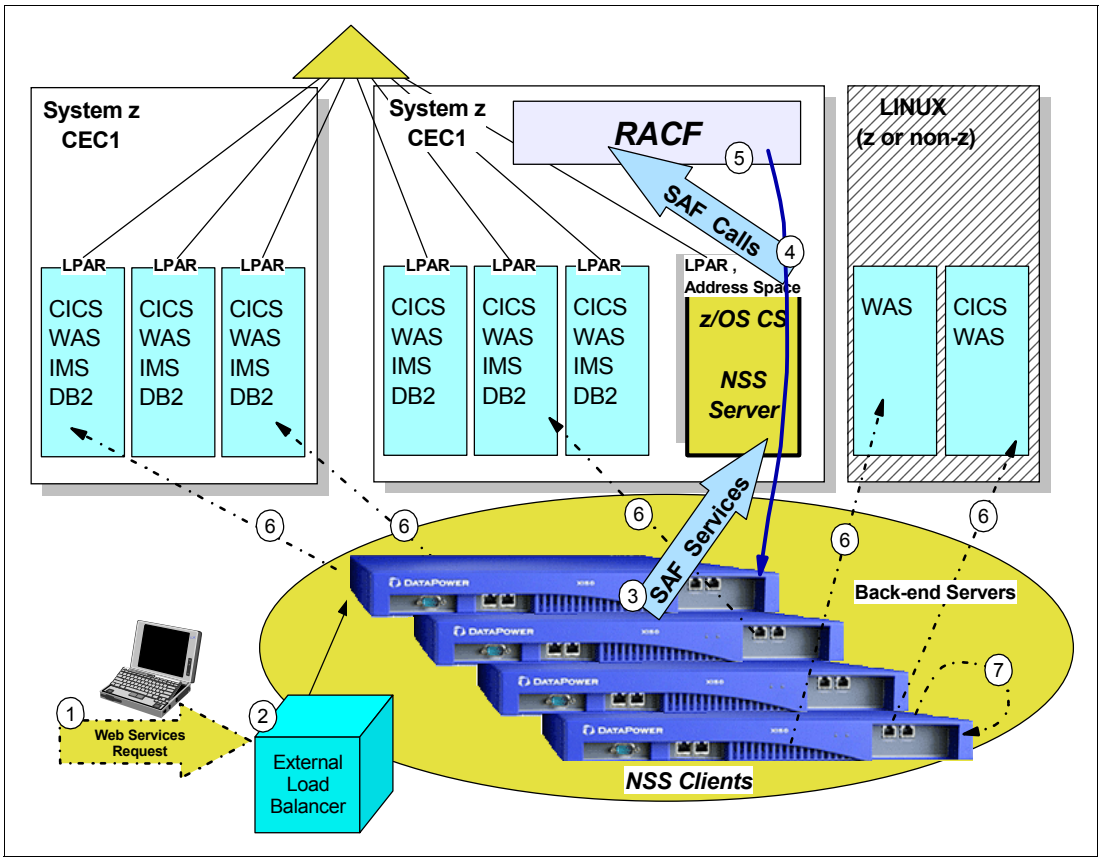


Figure 10-3 DataPower XML appliances accessing SAF services through z/OS NSS

Figure 10-3 includes the following sequence of events:

1. A user or program sends a web services request to the network with a destination IP address of a DataPower appliance.
2. The request arrives at a platform performing load balancing, where it is directed to one of the DataPower appliances. If the network configuration is not using load balancing, then the request arrives at one of the DataPower devices without the services of a load balancing technology.
3. The DataPower SOA appliance receives the request and parses it to determine what types of services should be invoked at the DataPower device. One service can be an SAF access service, causing the DataPower appliance to send out an SAF request to the NSS server over a connection secured with SSL/TLS and AT-TLS. The NSS server is enhanced to accept such queries from authorized middleware appliances.

   Although not depicted, the secured connection requires SSL key rings at the mainframe and a key repository at the WebSphere DataPower appliance.
4. The NSS server intercepts the SAF request and interfaces with the SAF product, which is RACF in this example.
5. The SAF product, RACF, authenticates the sender of the original web services request and determines whether the sender is authorized to access the resource requested in the web services request. RACF returns a reply to the NSS server and subsequently to the DataPower service.

6. After authentication and authorization, the DataPower appliance invokes any additional service that is associated with the original web request and, in this example, makes a connection to a final target application at a back-end server.
7. The connection can also be made to a server or process internal to the DataPower appliance.

A network administrator can also monitor the various types of NSS clients and the associated client connections using the z/OS UNIX **nssctl** command. Each type of NSS client displays in the client listing, along with pertinent information such as the NSS discipline, the services that they are accessing, name, (connection) state, the time connected, the time of the last message received, the selected and enabled services, and so on.

10.1.4 NSS solution for XMLAppliance clients: Private key and certificate services

The *XMLAppliance private key service* provides key services in the following ways:

- WebSphere DataPower can retrieve over the secured NSS connection a non-ICSF-protected RSA private key that is associated with an authorized certificate and perform RSA private key operations. For example, the platform can use the private key to generate RSA signatures *locally* or to decrypt RSA data *locally*.
- As a more secure alternative, a z/OS security administrator can choose to have a certificate's private keys stored in an ICSF-protected VSAM data set and allow only authorized clients to request certain ICSF operations, for example RSA decryption or RSA signature generation, using those RSA private keys. The XMLAppliance private key service allows authorized WebSphere DataPower clients the use of ICSF callable services for RSA private keys that are protected in such ICSF-protected VSAM data sets. In this case, the ICSF-protected private keys are not retrieved by this service. Instead, the operations with the RSA keys are performed entirely on System z, which is considered more secure because the RSA private keys never leave the system.

The *XMLAppliance certificate service* allows authorized clients to list and retrieve only authorized certificates from the NSS server's configured key ring. The client can request a list of x.509 certificates from the NSS server's key ring and then retrieve a DER-encoded x.509 certificate and use it for various RSA public-key operations. This service allows security administrators to distribute common certificates across an enterprise easily and allows each client to decide how to use this information.

Combined, these services allow z/OS system and security administrators tighter control over sensitive RSA keying material while extending the use of z/OS managed x.509 Certificates to non-z/OS clients.

Figure 10-4 shows DataPower XML appliances as clients accessing private key and certificate services through z/OS NSS. The text discusses the sequence of events when the z/OS NSS server provides centralized services for the security authorizations of the DataPower appliances. The figure shows how the NSS XMLAppliance private key and certificate services might fit into a typical DataPower deployment.

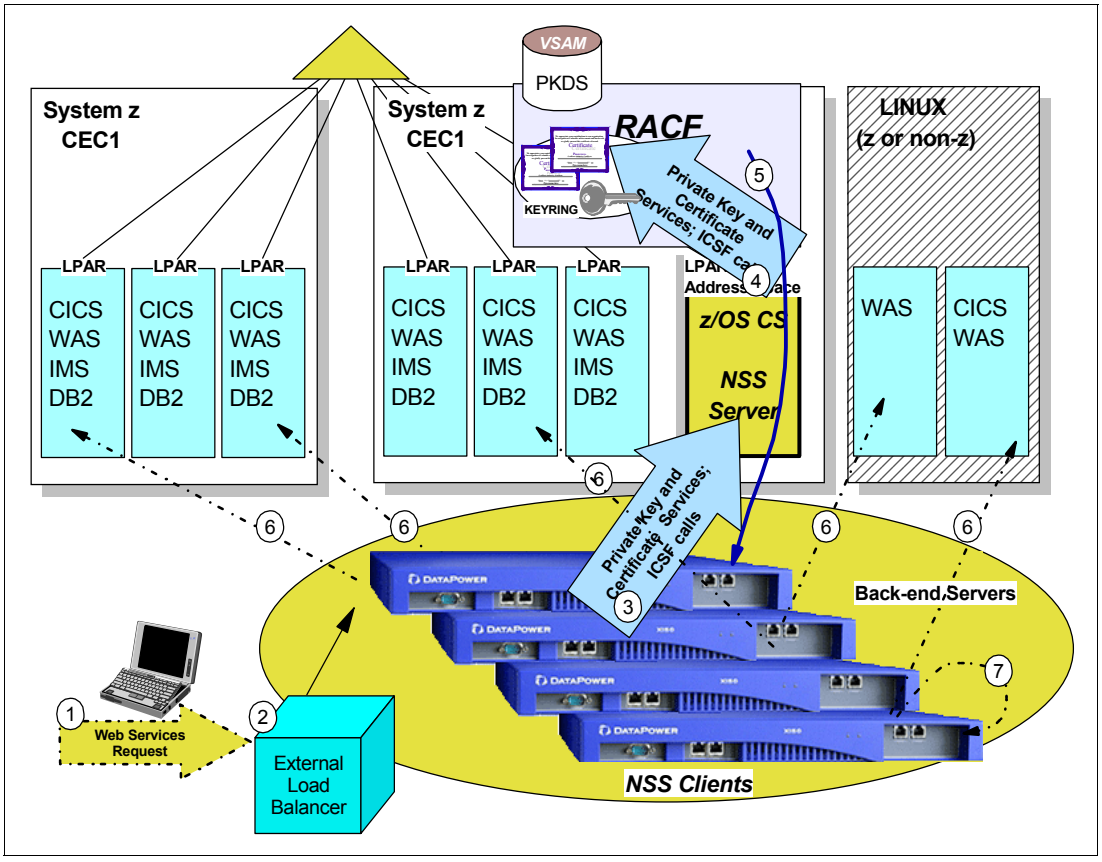


Figure 10-4 DataPower XML appliances accessing private key and certificate services

Figure 10-4 includes the following sequence of events:

1. A user or program sends a web services request to the network with a destination IP address of a DataPower appliance.
2. The request arrives at a platform performing load balancing, where it is directed to one of the DataPower appliances. If the network configuration does not use load balancing, then the request arrives at one of the DataPower devices without the services of a load balancing technology.
3. The DataPower SOA appliance receives the request and parses it to determine the types of services to invoke at the WebSphere DataPower device.

   With the NSS SAF access service, WebSphere DataPower can request client authentication or authorization to a particular SAF resource. Figure 10-3 on page 433 illustrates how this service works.

   Another service might be the certificate service. The XMLAppliance client can request a list of x.509 certificates from the NSS server's key ring. The client can then retrieve a DER-encoded x.509 certificate and use it for various RSA public key operations.

   Yet another service might be the private key service.

The request for the service causes the DataPower appliance to send a request over the secured connection to the NSS server.

Although not depicted, the secured connection between NSS and the WebSphere DataPower appliance requires SSL key rings at the mainframe and a key repository at the WebSphere DataPower appliance.

Figure 10-4 on page 435 illustrates the key ring that holds the personal certificates, which are associated non-ICSF private keys, and the certificate authority certificates that might be necessary for the private key and certificate services of the XMLAppliance discipline. NSS uses only one key ring for these security objects, which can be requested by multiple WebSphere DataPower appliances. The NSS server must have access to this key ring and must have access to the private keys and certificates on this key ring.

4. To service requests, the NSS server makes calls to the SAF, the Integrated Cryptographic Services Facility (ICSF), System SSL, and various z/OS UNIX system calls.
   - If this is a request for a list of x.509 certificates from the NSS server's key ring, RACF verifies that the XML client is authorized to the certificates. The list might include host (personal) certificates and certificate authority (CA) certificates.
   - If this is a request for a private key on the key ring, RACF verifies that the XML client is authorized to access the key. Alternatively, if the private key is managed by ICSF, RACF verifies that the XML client is authorized to access ICSF-protected resources.
5. Replies are returned to the NSS Server and, subsequently, to the WebSphere DataPower service.
6. The remainder of the DataPower appliance flows might proceed as usual. Such flows often terminate in large back-end server applications such as CICS, DB2, Information Management System (IBM IMS™), or WebSphere Application Server.

The configuration depicted in Figure 10-4 on page 435 allows a z/OS security administrator to centrally manage user authentication and authorization privileges for WebSphere DataPower clients, and to centrally manage and control access to RACF certificates and private keys.

A network administrator can also monitor the various types of NSS clients and the associated client connections using the z/OS UNIX **nssctl** command. Each type of NSS client displays in the client listing, along with pertinent information such as the NSS discipline, the services that they are accessing, name, (connection) state, the time connected, the time of the last message received, the selected and enabled services, and so on.

10.2 Configuring NSS

The network topology that we used to configure our NSS scenario includes a single DataPower XI50 middleware appliance that connects directly to a z/OS LPAR in which a NSS is executing. Figure 10-5 shows the network diagram for this scenario.

Tip: Our scenario illustrates the following components:

- The secured connection between the WebSphere DataPower appliance and the NSS

  This step is required for any implementation of the XMLAppliance discipline: SAF, private key, or certificate services.

- The SAF access service

  This step illustrates only the SAF access service. It does not illustrate the private key or certificate services. Consult the WebSphere DataPower documentation detailed in 10.4, “Additional information” on page 532 for information about implementing these type of services.

Figure 10-5 shows our test of a DataPower XI50 as an NSS XMLAppliance Client to NSS for SAF services on z/OS LPAR SC33.

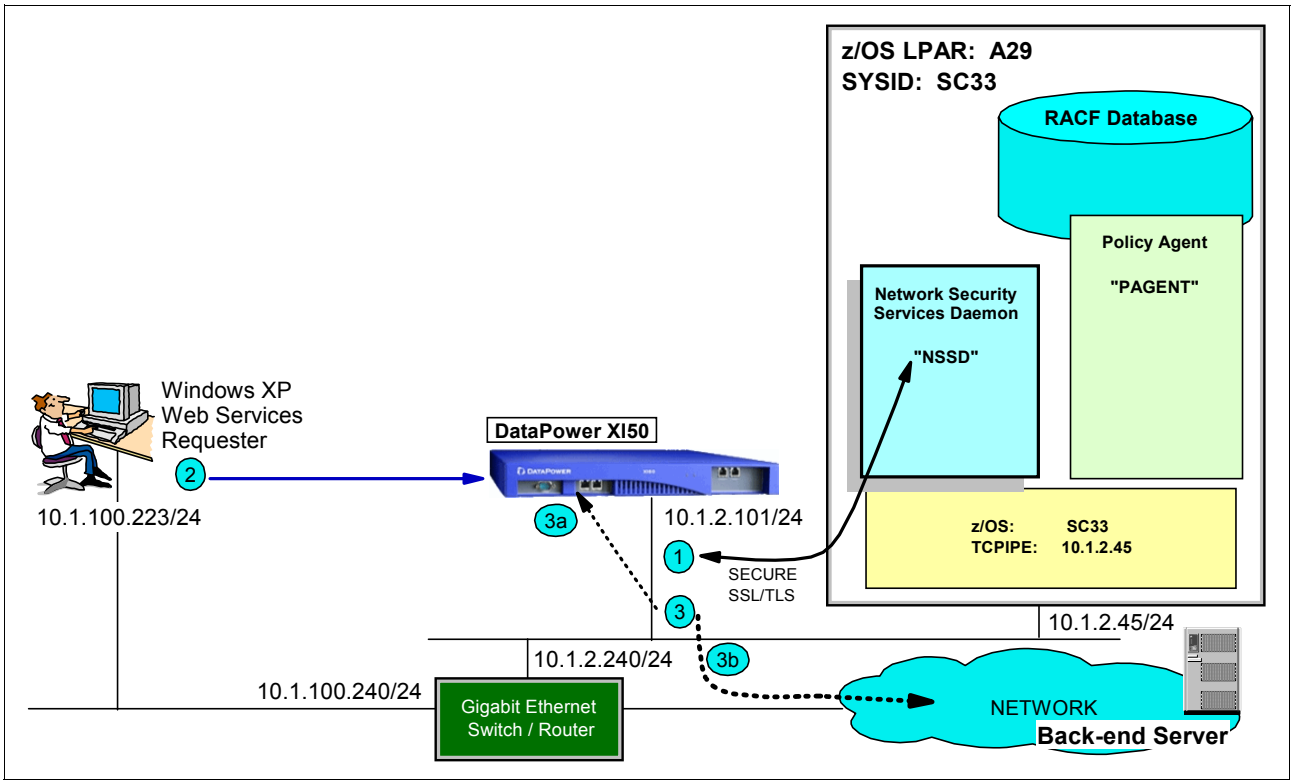


Figure 10-5 DataPower XI50 as an NSS XMLAppliance Client to NSS for SAF services

At 1 in Figure 10-5 on page 437, a secure connection is established between the WebSphere DataPower appliance and the NSS Server. The connection requires SSL/TLS with server authentication. The z/OS Communications Server end of the connection is configured with an AT-TLS policy that is loaded by policy agent into the running TCP/IP stack. This connection between NSSD and the WebSphere DataPower appliance establishes a trusted relationship between the two endpoints so that authentication can be performed against the RACF database for Web Requesters that contact the DataPower appliance.

At 2 in Figure 10-5 on page 437, a Web Requester or user submits a request from a Windows XP platform to the DataPower appliance. The DataPower appliance is updated with information that intercepts the request and invokes one of the XMLAppliance services. If it is an SAF service, a request is sent to RACF over the trusted connection established in 1 to authenticate the Web Services Requester and to authorize it to access resources on the System z. Alternatively, the request from the Web Requestor might cause the DataPower client to request XMLAppliance private key or certificate services from the NSS, although these services are not tested in our scenario.

At 3 in Figure 10-5 on page 437, a positive response to the request or requests for an NSS service causes the DataPower appliance itself to invoke one or more subsequent services. The service might be an XML transform service, a request to connect to a service within the appliance (3a), a request to connect to another node outside the appliance using SSL/TLS (3b), or any other relevant action or combination of actions.

Our simple test scenario focuses on the SAF service of the NSS Server XML Discipline and connects only to services within the DataPower appliance as in 3a. We do not require the services of a back-end server.

10.2.1 Overview of NSS configuration for an NSS XMLAppliance Client

The flows in Figure 10-5 on page 437 require customization steps at the z/OS Communications Server, at the DataPower appliance, and at the platform that the Web Requester uses. Figure 10-6 depicts the relationships among the components on each of the platforms:

- The z/OS environment with SAF, an external security manager (RACF in our case), and z/OS Communications Server
- The WebSphere DataPower appliance environment
- The Web Requester environment

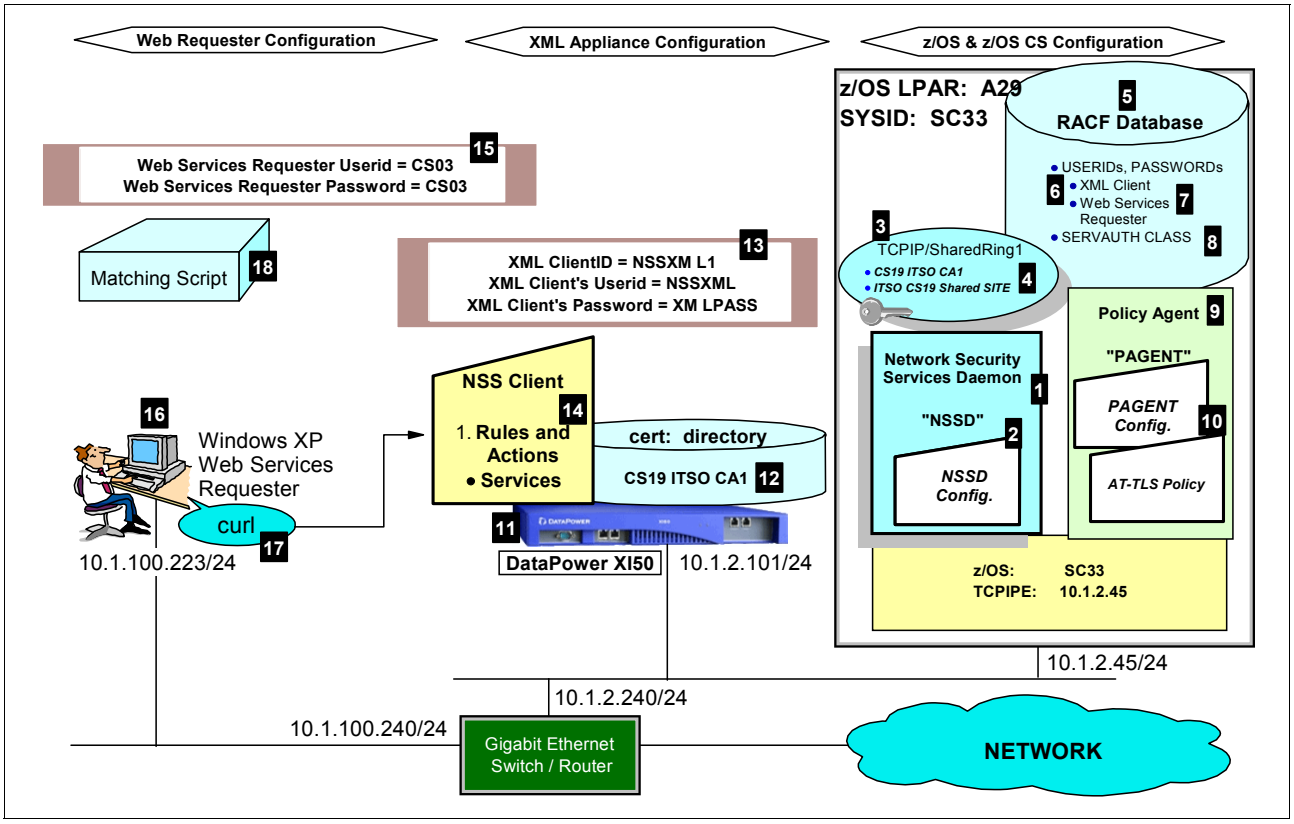


Figure 10-6 Implementation Detail for the XMLAppliance Discipline

The z/OS environment and configuration

At 1 in Figure 10-6, you see the NSS Daemon (NSSD) with its configuration file (2). You also see a key ring owned by TCP/IP, SharedRing1 (3), which is populated with a certificate authority (CA) certificate named `CS19 ITSO CA1` and a server certificate labeled `ITSO CS19 Shared SITE` (4). The key ring and the certificates with any private key are stored in the RACF database (5). This key ring is used by the NSS Server for its AT-TLS secured connection with NSS clients.

The RACF database (5) is updated also with the user ID and password or application key of the NSS XMLAppliance client (6), the user IDs and passwords of the Web Services Requesters (7), and with a number of SERVAUTH class definitions (8) that further secure access to SAF services on the System z.

The policy agent (9) is updated to include an AT-TLS policy (10) for the secure communication between the DataPower XML Appliance client and the NSS server.

The DataPower SOA Appliance environment and configuration

The DataPower SOA Appliance (11) in the middle of the visual in Figure 10-6 on page 439 represents the NSS XMLAppliance Client. The NSS XMLAppliance client must be configured for SSL/TLS to build the trusted communication path to the NSS server. Therefore, at point 12, you see that we have stored in the appliance's key database the CA certificate that has signed the NSS server certificate on z/OS (4). This CA certificate must be exported from its repository on z/OS and then sent to the appliance to store on the key database. As we described in Chapter 3, "Certificate management in z/OS" on page 39, the CA's signature authenticates the server certificate that is sent to the client during SSL/TLS negotiation.

In defining the NSS XMLAppliance client configuration in the DataPower SOA Appliance, we assigned a client ID of *NSSXML1* for the NSS client, a client user ID of *NSSXML,* and a password of *XMLPASS* for authentication with RACF (13).

Finally, we defined a set of rules and actions (14) at the DataPower appliance that will invoke the various services required:

- The SAF access service that is to be directed to the NSS server
- Any other services that can be invoked when the Web Services Requester submits a request to the DataPower appliance

The Web Services Requester environment and configuration

The Web Services Requester on the left side of the visual in Figure 10-6 on page 439 is identified in RACF with a user ID of *CS03* and a password of *CS03* (15). The user submits a request for web services from a workstation (16) through a **curl** command (17) that points to a script stored on the workstation. The script (18) is called a *matching script*, because its contents are matched against a *matching rule* (14) that was defined previously on the DataPower appliance. In our example, the matching script contents match the condition defined in the matching rule, and thus the actions (that is, services) specified in the rule are executed.

Tips: In our test, we requested the following services:

- SAF authentication of the user ID and password of the Web Services Requester
- Authorization to access a resource on the System z

Our test does not request any further processing, such as a COBOL transformation or a new HTTP or a WebSphere Application Server connection to a back-end server.

Although our example used the script contents as the matching criteria, in fact other criteria, such as an incoming URL, can be used as a match to trigger an action.

10.2.2 Preparing for configuration

The easiest way to build an NSS environment is to have the prerequisite technologies such as policy agent and AT-TLS already functioning at z/OS. Thus, the policy agent is already running on the z/OS system that is to house the NSS server. Also, working x.509 certificates are available. With these technologies firmly in place, it is then easier to set up the NSSD environment for an NSS client that is connecting to the XMLAppliance Discipline.

The prerequisite technologies also mean that a middleware appliance such as DataPower is available and already attached to the network.

Finally, having the prerequisites means that a logical network diagram, such as the one presented in Figure 10-6 on page 439, has been developed to represent the relationships, the IP addresses, and the desired naming conventions of the NSS and XML environment.

Worksheet with NSS and XML Naming and Numbering Conventions

To further simplify the implementation of NSS, we provide a worksheet that you can complete with all the information that is required for a successful NSS implementation. A blank worksheet for you to complete is provided in 10.5, “NSS configuration worksheet for an NSS XMLAppliance client” on page 533.

Here is a sample worksheet that we completed for our implementation as illustrated in Figure 10-6 on page 439:

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon?
   - SC33
2. What is the superuser user ID associated with the OMVS segment under which the NSS daemon is to run?
   - NSSD
3. With which TCP/IP stack (or stacks) does the NSS daemon establish affinity?
   - TCPIPE
4. What is the address or the DNS name that the NSS XMLAppliance client will use to connect to the NSS server?
   - 10.1.2.45
5. What “symbolic client name” or ClientID do you want to assign to each NSS XMLAppliance client that is to request authentication of the NSS server?
   - NSSXML1
6. What RACF “user ID” do you want to assign to each NSS XMLAppliance Client ID that is to request authentication of the NSS server?
   - NSSXML

   Rule: Multiple NSS clients can use a single user ID. However, each NSS client must have a unique symbolic client name (i.e. ClientID).

   Guideline: Because SAF user IDs are used to authorize a client to the NSS services, avoid sharing a single user ID across NSS disciplines.

7. What type of authentication should our NSS XMLAppliance client user ID present: a password or a passticket?
   - password of XMLPASS
8. What is the name of the key ring that NSS uses to manage the necessary certificates and private keys for NSS IPSec clients, and for XMLAppliance private key services and certificate services clients? Although the scenario to support NSS XMLAppliance SAF services clients does not require this key ring, the configuration of the NSS server includes this key ring name.
   - NSSD/NSSD_keyring

9. What are the labels of the certificates that need to reside on this NSS key ring. You need to document the labels of the certificates for the IPSec discipline and for the private key and certificate services of the XMLAppliance discipline.
   - _____N/A_____ in our XMLAppliance SAF services scenario
   - _____
   - _____
10. What is the name of the key ring that NSS uses to manage the certificates for secure communications between itself and its NSS XMLAppliance clients?
   - TCPIP/SharedRing1
11. What are the labels of the certificates on the NSS node that are used for the TLS/SSL secure connection between the NSS server and the NSS client?
   - Signing Certificate: CS19 ITSO CA1
   - Server Certificate: ITSO CS19 Shared SITE
12. Do you plan to use x.500 distinguished name filtering in RACF for certificate mapping? If so, you will need to activate RACF digital certificate mapping.
   - No
13. If the NSS server has implemented IP filtering and IPSec, what are the IP addresses of the NSS server and the NSS client for which you might have to establish IP filters on the NSS server node?
   - Client: 10.1.2.101
   - Server: 10.1.2.45
14. What are the TSO user IDs of the IPSec administrators who are authorized to manage NSS and its clients?
   - CS02, CS03, CS04, CS05, CS06 (all members of the SYS1 RACF group)
15. What are the RACF user IDs of the Web Services Requesters that might need authentication and authorization at z/OS RACF?
   - CS02, CS03, CS04, CS05, CS06 (all members of the SYS1 RACF group)
16. What are the SERVAUTH resources to which the Web Services Requester user IDs require authorization and what type of access do they need (Read, Update, Control)?
   - SERVAUTH class of EZB.DB2.DB2PROCA.SALES (access Read)

Role of the IBM Configuration Assistant

The IBM Configuration Assistant program allows you to configure an NSS server for an NSS client through a graphical user interface (GUI). It then generates the NSSD configuration file and the necessary AT-TLS policy for NSS client-server traffic between the server and the NSS client. For our scenario, which employed the XMLAppliance Discipline for an NSS client, we primarily used the existing flat files for the NSS server to configure the environment rather than the GUI. In our tested scenario we used the GUI only for configuring the AT-TLS policy. However, we also include information about using the GUI to build the JCL, the RACF definitions, the SYSLOGD directives, and so on in 10.2.4, “Creating NSS Server files for an NSS XMLAppliance Client with IBM Configuration Assistant” on page 465.

10.2.3 Configuring the NSS environment at z/OS

Tip: Although our scenario depicts only the SAF service, this section applies to the implementation of any service of the XMLAppliance discipline:

- SAF service
- Private key service
- Certificate service

As we have discussed, you need to complete many tasks prior to defining the NSS server and client environment. Figure 10-6 on page 439 shows many of these prerequisite components. Figure 10-7 focuses on the z/OS portion of the entire environment as follows:

- We begin configuring the NSS environment by showing the RACF environment authorizations. (5 through 8.) You learn how the RACF authorizations relate to the definitions in the DataPower appliance. (13.)
- From there, we show how to configure the NSSD, its JCL, and its configuration file. (1 through 2.) Although not depicted in Figure 10-7, we show how to configure SYSLOG daemon to isolate NSSD messages.
- From there, we proceed to the creation of certificates for the AT-TLS policy. (3 and 4.)
- Next, we show the AT-TLS policy that secures the connection between the NSS and its XMLAppliance clients. (9 and 10.)
- After this, we show how to code the TCP/IP profile for AT-TLS and optionally IPSec. (A.)

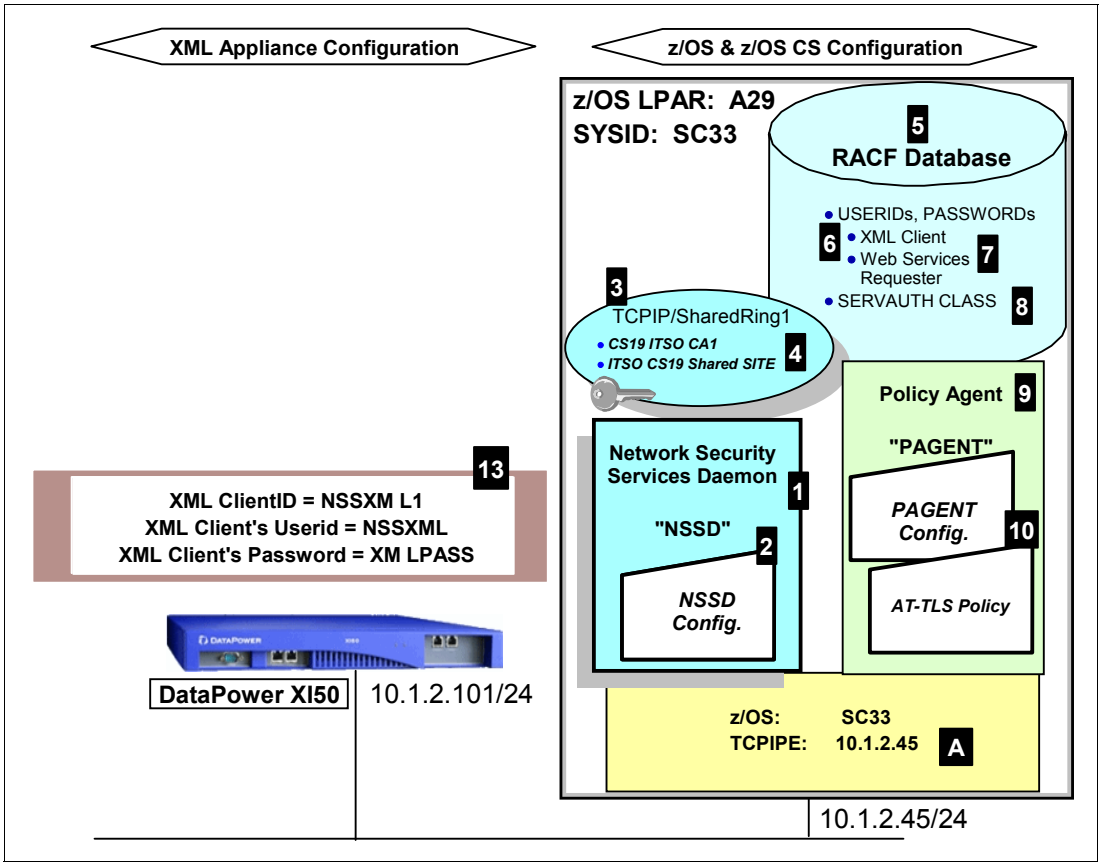


Figure 10-7 z/OS Details for an NSS and NSS XMLAppliance client scenario

Configuring authorizations for the NSS and its users

We used RACF as our external security manager for NSS. 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 10-1 shows the RACF commands that 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 10-1 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
   SETROPTS GENERIC(STARTED) REFRESH
   ```

2. Permit the NSSD user ID to SYS1.PARMLIB:

   ```
   PERMIT SYS1.PARMLIB ID(NSSD) ACCESS(READ)
   ```

3. Define 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. Permit the administrator user IDs that will administer key ring access to the following facility classes. The users CS01 through CS06 belong to the RACF group SYS1:

   ```
   PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(SYS1) ACC(UPDATE)
   PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.GENREQ CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(SYS1) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(SYS1) ACC(UPDATE)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

5. Permit the NSSD user ID READ access to key rings it owns and CONTROL and UPDATE access to key rings it does not own:

   ```
   PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(NSSD) ACC(CONTROL)
   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(NSSD) ACC(UPDATE)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

   Tip: In our scenarios, NSSD owns the NSSD key ring for the NSS clients. For this key ring, NSSD requires only READ access. However, our scenarios also require access to the AT-TLS key rings, which are owned by the user ID of TCPIP. Therefore, the NSSD user ID requires CONTROL and UPDATE access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING facility classes respectively, as shown in our examples.

6. Permit the NSSD user ID to the BPX.DAEMON facility class:

   ```
   PERMIT BPX.DAEMON CLASS(FACILITY) ID(NSSD) ACCESS(READ)
   setropts raclist(facility) refresh
   ```

7. If you have chosen to implement secured signon for the NSS client's user ID with passtickets, follow the instructions for passticket authorization as detailed in *z/OS*

Communications Server: IP Configuration Guide, SC31-8775. We elected to use a simple password authentication for our NSS client user IDs. We, therefore, skipped this step.

8. If you have chosen to implement RACF certificate name filtering for an NSS client, follow the instructions for certificate name filtering as detailed in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. We chose not to map x.500 distinguished names to a RACF ID and, therefore, skipped the step to define the DIGTNMAP class.

Tip: The next implementation steps require that you authorize SERVAUTH access to the NSS client. The maximum allowable length of a SERVAUTH profile name is 64 characters. Each NSS discipline and its service requires a separate set of SERVAUTH authorizations, as follows:

- IPSec Certificate Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.IPSEC.CERT`

- IPSec Network Management Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.IPSEC.NETMGMT`

- XML Appliance SAF Access Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.XMLAPPLIANCE.SAFACCESS`

9. Our scenarios do not illustrate the XMLAppliance Private Key Service or the XML Appliance Certificate Service. However, if you are planning to exploit these services, you must also provide the following SERVAUTH authorizations in a later step.

Tip: The following SERVAUTH authorizations control whether an NSS XMLAppliance client can register with the NSS server for the service:

- XML Appliance Certificate Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.XMLAPPLIANCE.CERT`

- XML Appliance Private Key Service

  `EZB.NSS.`*`sysname.symbolic_clientname`*`.XMLAPPLIANCE.PRIVKEY`

The NSS IPSec client

The tasks in this step apply to the *NSS IPSec client* only:

1. Create a user ID for the NSS IPSec client. We named our IPSec client *NSS32A*. The client's password is to be *NSSPASS*. Example 10-2 shows the JCL that we used to create this user ID.

Example 10-2 IPSec Client user ID and password generation

```
//ADDNSSx 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   NSSXML GROUP(SYS1) OWNER(HAIMO) AUTHORITY(JOIN) +
            UACC(ALTER)
  ALU NSSXML 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 NSSXML -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
//*
```

2. If you choose to define an NSSD profile in the APPL class with UACC(NONE), issue the following command to authorize each SAF user ID to the NSSD application:

```
PERMIT NSSD CLASS(APPL) ID(NSS32A) ACC(READ)
SETROPTS RACLIST(APPL) REFRESH
```

3. Permit this user ID to a new class that is created with the commands shown in Example 10-3. This class is used for the NSS IPSec Certificate Services.

Example 10-3 Authorizing user IDs to new SERVAUTH Cert 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
```

This class uses the following syntax:

EZB.NSS.<*NSSD MVS System ID*>.<*NSS Client Symbolic Name*>.CERT

Now you understand why you should design and diagram this scenario carefully, and that you assign meaningful naming conventions to it.

You need a SERVAUTH class similar to this class for each client. Alternatively, you can use wildcards, but you lose some granularity and control with wildcards.

4. NSS has a second capability beyond IPSec Certificate Services called *IPSec management*. Therefore, permit the NSS IPSec client user ID and the user ID of the NSS administrator to a new SAF class that is created with the commands shown in Example 10-4. This class is used for the NSS Network Management Services.

Example 10-4 Authorizing user IDs to new SERVAUTH facility class for NSS IPSec discipline

```
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
```

The syntax of this second class is similar to the syntax 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

5. Define a unique SERVAUTH class resource profile for each NSS IPSec client. One of the fields in it is composed of the *label name* of the NSS IPSec 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.

You first need to verify the label on that certificate using a RACF command to list the contents of the IKE key ring that contains the certificate assigned to the TCPIPA NSS IPSec client as shown in Example 10-5.

Example 10-5 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, you verify that the label that is assigned to the client certificate is *IKE Daemon on SC32*.

Now, you can define the new class resource with one of the following formats:

- `EZB.NSSCERT.<NSSD MVS System ID>.<NSS Client IKE Certificate Label>.HOST`
- `EZB.NSSCERT.<NSSD MVS System ID>.<mappedlabelname>.HOST`

Example 10-6 shows our definition.

Example 10-6 SERVAUTH Resource profile for NSS IPSec client's User (Personal) 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)
ACCESS(READ)
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 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 ($).

This rule affects the following characters:

- Asterisk (*)
- Percent sign (%)
- Ampersand (&)
- Blank

Tip: We executed the RDEFINE and the PERMIT statements from the TSO ISPF environment. As a result, even if we entered the label in lowercase, our definitions would be successful because TSO would convert the alphabetics to uppercase automatically.

For more information about mapped label names, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

6. Now that you are now familiar with mapped label names, you need to build another profile for the CA certificate that has signed the NSS IPSec client's certificate. As established in Example 10-5 on page 447, that label is `My Local Certificate Authority` as shown in Example 10-7.

Example 10-7 SERVAUTH Resource profile for NSS IPSec client's CA 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, which represent the blanks in the certificate label of the CA certificate. The syntax is similar to previous syntax:

- EZB.NSSCERT.*<NSSD MVS System ID>*.*<NSS Client CA Certificate Label>*.CERTAUTH
- EZB.NSSCERT.*<NSSD MVS System ID>*.*<mappedlabelname>*.CERTAUTH

Hint: To avoid having to insert dollar signs ($) into mapped label names, avoid using the following characters when creating RACF certificates:

- Asterisk (*)
- Percent sign (%)
- Ampersand (&)
- Blank

To avoid the uppercase conversion requirement that is imposed by SERVAUTH definitions, always create RACF certificates in uppercase.

7. Example 10-8 shows additional SERVAUTH resource that you must define to allow remote users to monitor (DISPLAY) or manage (CONTROL) NSS IPSec clients that are using the IPSec Discipline.

Example 10-8 Resource to monitor and manage NSS IPSec 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(CSO1,
CSO2,CSO3,CSO4,CSO5,CSO6,CSO7,CSO9,CS10) ACC(READ)

PERMIT EZB.NETMGMT.SC33.SC32_TCPIPA.IPSEC.DISPLAY CLASS(SERVAUTH) ID(CSO1,
CSO2,CSO3,CSO4,CSO5,CSO6,CSO7,CSO9,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
- EZB.NETMGMT.*<NSSD MVS System ID>*.*<NSS Client Symbolic Name>*.IPSEC.DISPLAY

In our case, we have a few remaining SERVAUTH resources to define. If your user, CS02, wants to execute the **ipsec** command with the **-x** option, the user can display information about the NSS server and its IPSec clients. Likewise, if your user wants to display information about any type of client (IPSec or XML) supported by the NSS server, that user needs authorization to the **nssctl** command. Therefore, on the server system, you need the resource profile and permissions for CS02 shown in Example 10-9.

Example 10-9 SERVAUTH profile to display NSS information with nssctl or ipsec -x

```
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 10-9 on page 449 is quite different from previous syntax:

```
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.

Important: There are two display commands for NSS management:

- The **ipsec** command with its various options
- The **nssctl** command

Authorization for both commands is provided through the resource profile:

```
EZB.NETMGMT.<NSS Server SYSID>.<NSS Server SYSID>.NSS.DISPLAY
```

Although both commands are used for the NSS IPSec (IKED) discipline, the XML discipline takes advantage only of the **nssctl** command.

If user CS02 wants to display information about the IKE daemon at the NSS server node with the **ipsec -w display** command, define the resource profile as shown in Example 10-10.

Example 10-10 Displaying IKE at server with ipsec -w display

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.IKED.DISPLAY CLASS(SERVAUTH) ID(CS01,CS02,CS03,
CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Tip: Up until this point, all the SERVAUTH resource profiles have been defined on the NSS server node, SC33. This pattern changes with the next SERVAUTH resource profile that is defined on the IKE node that represents the NSS IPSec client.

8. If user CS02 wants to display information about the IKE daemon's NSS IPSec clients, then you need to define the resource profile as shown in Example 10-11 at the NSS IPSec client node. The command for this type of display is again **ipsec -w**.

Example 10-11 Displaying IKE at NSS IPSec client with ipsec -w

```
rdefine SERVAUTH EZB.NETMGMT.SC32.SC32.IKED.DISPLAY

PERMIT EZB.NETMGMT.SC32.SC32.IKED.DISPLAY CLASS(SERVAUTH) ID(CS01,CS02,CS03,
CS04,CS05,CS06,CS07,CS09,CS10) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

Tip: Although the numerous authorizations that we describe here might seem quite tedious, we show them in detail to emphasize several critical points:

- The numerous individual SERVAUTH profiles and authorizations provide a great deal of control and security. To understand them well, it is necessary to provide this level of detail. However, after you understand the concepts, you can use wildcards in many of the fields to reduce the amount of definition that is required.
- The definitions require detailed planning. As mentioned in 10.2.1, “Overview of NSS configuration for an NSS XMLAppliance Client” on page 439, two of the planning steps are to prepare a logical network diagram and to complete a worksheet that contains the information for these definitions. This preliminary work is useful to expediting the steps that we cover in “Configuring authorizations for the NSS and its users” on page 444.
- If you have already worked with IPSec and IP Filtering, the task of setting up NSSD for IPSec clients becomes quite simple.

The NSS client (XML Appliance Discipline) RACF user ID

The authorization tasks in this section apply to the NSS client of the XML Appliance Discipline only. To authorize the NSS client:

1. Create a user ID for the NSS XMLAppliance client. In our testing, we named our NSS XMLAppliance client *NSSXML*. The client’s password is *XMLPASS*. Example 10-2 on page 445 shows the JCL that we used to create this user ID.

Example 10-12 NSS Client user ID and password generation

```
//ADDXML 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 NSSXML PASSWORD(NEW2DAY) +
    NAME('ID for Comm Server') +
    OWNER(HAIMO) UACC(READ) DFLTGRP(SYS1) +
    AUTHORITY(JOIN) GRPACC +
    OMVS(AUTOUID HOME(/u/nssxml) PROGRAM(/bin/sh))
  CONNECT   NSSXML GROUP(SYS1) OWNER(HAIMO) AUTHORITY(JOIN) +
            UACC(ALTER)
  ALU NSSXML PASSWORD(XMLPASS) NOEXPIRED
  PASSWORD USER(NSSXML) NOINTERVAL
  ADDSD     'NSSXML.**' UACC(NONE) OWNER(NSSXML)
  SETROPTS REFRESH RACLIST(TSOPROC ACCTNUM TSOAUTH)
  DEFINE ALIAS   (NAME('NSSXML') RELATE('UCAT.COMCAT'))
/*
//DEFINEZ  EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//SYSIN    DD     *
```

```
     DEFINE CLUSTER (NAME(NSSXML.HFS) -
            LINEAR CYLINDERS(1 1) SHAREOPTIONS(3)    -
            VOLUMES(COMST4))
/*
//CREATEZ  EXEC   PGM=IOEAGFMT,
//           PARM=('-aggregate NSSXML.HFS -compat
//             -owner NSSXML -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/nssxml/.profile'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  PROF MSGID WTPMSG
  OCOPY INDD(PDS1) OUTDD(HFS1) TEXT
/*
//CHOWN1     EXEC PGM=BPXBATCH,
//   PARM='SH chown nssxml /u/nssxml/.profile'
//STDOUT     DD  PATH='/tmp/stdout',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//STDERR     DD  PATH='/tmp/stderr',
//           PATHOPTS=(OWRONLY,OCREAT),
//           PATHMODE=SIRWXU
//*
```

2. If you choose to define an NSSD profile in the APPL class with UACC(NONE), issue the following command to authorize each SAF user ID to the NSSD application:

```
PERMIT NSSD CLASS(APPL) ID(NSSXML) ACC(READ)
SETROPTS RACLIST(APPL) REFRESH
```

3. Permit this user ID to a new class that is created with the commands shown in Example 10-3 on page 447. Sample RACF JCL to provide the necessary authorizations is documented in TCPIP.TCPIP.SEZAINST(EZARACF).

   Tip: Example 10-3 shows all the authorization steps for all services of the XMLAppliance discipline, although our test scenario depicts only the SAF access service. You only need to perform the steps that apply to the scenario that you are implementing.

Example 10-13 Authorize NSS XMLAppliance client user IDs to SERVAUTH classes for NSS service types

```
RDEFINE SERVAUTH EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.SAFACCESS UACC(NONE) 1

PERMIT EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.SAFACCESS CLASS(SERVAUTH) ID(NSSXML) ACCESS(READ) 1

RDEFINE SERVAUTH EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.CERT UACC(NONE) 2

PERMIT EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.CERT CLASS(SERVAUTH) ID(NSSXML) ACCESS(READ) 2

RDEFINE SERVAUTH EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.PRIVKEY UACC(NONE) 3

PERMIT EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.PRIVKEY CLASS(SERVAUTH) ID(NSSXML) ACCESS(READ) 3

SETROPTS RACLIST(SERVAUTH) REFRESH

SETROPTS GENERIC(SERVAUTH) REFRESH
```

In this example, the numbers correspond to the following information:

1. The authorizations for the XMLAppliance SAF access service
2. The authorizations for the XMLAppliance certificate service
3. The authorizations for the XMLAppliance private key service

Note the syntax of this class:

EZB.NSS.<*NSSD MVS System ID*>.<*NSS Client Symbolic Name*>.XMLAPPLIANCE.<*NSS Service Type*>

You need SERVAUTH class definitions similar to this class for each symbolic ClientID. Alternatively, you can use wildcards for the definition, but you lose some security granularity and control when you do so.

4. Create a SERVAUTH resource profile for each certificate that an NSS XMLAppliance client could retrieve and give the user ID of the NSS XMLAppliance client access to the appropriate profiles. A client might need access to a personal (user or host) certificate and to one or more CA certificates. See EZARACF in TCPIP.TCPIP.SEZAINST, or use the definitions in Example 10-14 on page 454.

   You first need to verify the labels on the certificates using a RACF command to list the contents of the NSSD's key ring that contain the following components:

   – The certificate that we added for the XMLAppliance client named *NSSXML*
   – The CA certificate that signed the personal certificate

You see the results at A and B in Example 10-14.

Example 10-14 RACDCERT LISTRING(NSSD_keyring) ID(NSSD)

```
Digital ring information for user NSSD:

Ring:
     >NSSD_keyring<
Certificate Label Name              Cert Owner     USAGE      DEFAULT
--------------------------------    ------------   --------   -------
My Local Certificate Authority B    CERTAUTH       CERTAUTH     NO

IKE Daemon on SC32                  ID(NSSD)       PERSONAL     NO

NSSXML DataPower 1 A                ID(NSSXML)     PERSONAL     NO
```

With this display, verify that the label that is assigned to the client certificate is *NSSXML DataPower 1*. This label appears as one of the fields (`mapped label name`) in the SERVAUTH resource profile named EZB.NSSCERT.*<NSSD MVS System ID>*.*<mappedlabelname>*.HOST. This resource grants access by the client to its own certificate, which is centrally stored on the NSS server's key ring.

Example 10-14 also shows (B) the label of the signing certificate, which is `My Local Certificate Authority`. This label appears as one of the fields (`mapped label name`) in the SERVAUTH resource profile named EZB.NSSCERT.*<NSSD MVS System ID>*.*<mappedlabelname>*.CERTAUTH. This resource grants access by the client to a copy of any CA certificates to which it might require access that are centrally stored on the NSS server's key ring.

5. Define the new class resource for the personal certificate as follows:

   EZB.NSSCERT.*<NSSD MVS System ID>*.*<mappedlabelname>*.HOST

   Example 10-15 shows our definition and how we have authorized the user ID of NSSXML to it.

   Example 10-15 Resource profile for NSS IPSec client's User (Personal) certificate

```
RDEFINE SERVAUTH EZB.NSSCERT.SC33.NSSXML$DATAPOWER$1.HOST UACC(NONE)
PERMIT EZB.NSSCERT.SC33.NSSXML$DATAPOWER$1.HOST CLASS(SERVAUTH) ID(NSSXML)
ACCESS(READ)
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 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 ($).

   This rule affects the following characters:

   – Asterisk (*)
   – Percent sign (%)
   – Ampersand (&)
   – Blank

6. Define the SERVAUTH class resource for any required CA certificates that the WebSphere DataPower appliance might need to retrieve. Use the following format:

   EZB.NSSCERT.<*NSSD MVS System ID*>.<*mappedlabelname*>.CERTAUTH

   Example 10-16 shows our definition and how we authorized the user ID of NSSXML to it.

   Example 10-16 Resource profile for CA certificates required by the NSS IPSec client

```
RDEFINE SERVAUTH EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.CERTAUTH
UACC(NONE)
PERMIT EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.CERTAUTH CLASS(SERVAUTH)
ID(NSSXML) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS GENERIC(SERVAUTH) REFRESH
```

   We defined only one resource profile for the CA certificate because we allowed this CA certificate to sign all necessary personal certificates. If various CA certificates signed different certificates, then we would need to create additional SERVAUTH resource profiles to identify the mapped label names of those certificates.

7. If using private keys with or without RSA operations, define the SERVAUTH class resource to allow an XMLAppliance client to retrieve the private key for its own certificate. This definition is required for either ICSF-protected or non-ICSF-protected private keys. Use this format:

   EZB.NSSCERT.<*NSSD MVS System ID*>.<*mappedlabelname*>.PRIVKEY

   Example 10-17 shows our definition and how we authorized the user ID of NSSXML to it.

   Example 10-17 Resource profile to retrieve private key for NSS IPSec client

```
RDEFINE SERVAUTH EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.PRIVKEY UACC(NONE)
PERMIT EZB.NSSCERT.SC33.MY$LOCAL$CERTIFICATE$AUTHORITY.PRIVKEY CLASS(SERVAUTH)
ID(NSSXML) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS GENERIC(SERVAUTH) REFRESH
```

   We defined only one resource profile for the CA certificate because we allowed this CA certificate to sign all necessary personal certificates. If various CA certificates signed different certificates, then we would need to create additional SERVAUTH resource profiles to identify the mapped label names of those certificates.

8. If using ICSF-protected keys with or without RSA operations, authorize the NSS server to the Integrated Cryptographic Services Facility (ICSF) by defining profiles for CSFSERV services. The XMLAppliance certificate services use the following callable ICSF service names for operations with the cryptographic coprocessor:

   – CSNDDSG
   – CSNDPKD

   Therefore, you need to establish the definitions as shown in Example 10-18 for the NSS server.

   Example 10-18 Authorizations for XMLAppliance clients to ICSF

```
RDEFINE CSNDDSG CLASS(CSFSERV) UACC(NONE)
PERMIT CSNDDSG CLASS(CSFSERV) ID(NSSD) ACCESS(READ)
RDEFINE CSNDPKD CLASS(CSFSERV) UACC(NONE)
PERMIT CSNDPKD CLASS(CSFSERV) ID(NSSD) ACCESS(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) REFRESH
```

9. The NSS XMLAppliance SAF access service can use RACF certificate name filtering to map an X.500 distinguished name to a RACF ID when performing SAF access checks. Consult the *z/OS Communications Server: IP Configuration Guide*, SC31-8775 for information about implementing the DIGTNMAP class and name filtering if you exploit this function.

10. In our case, we have another SERVAUTH resource to define for our NSS XMLAppliance discipline scenario. If a user, CS02, wants to display information about the NSS server and the clients it is supporting, that user needs authorization to the **nssctl** command. You provide authorization through the resource profile and permissions for CS02's RACF group as shown in Example 10-19.

Example 10-19 SERVAUTH profile to display NSS information with ipsec -x or nssctl -d

```
rdefine SERVAUTH EZB.NETMGMT.SC33.SC33.NSS.DISPLAY

PERMIT EZB.NETMGMT.SC33.SC33.NSS.DISPLAY CLASS(SERVAUTH) ID(SYS1) ACC(READ)

setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

This syntax in Example 10-19 is quite different from previous syntax:

```
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.

Important: There are two display commands for NSS management:

- The **ipsec** command with its various options
- The **nssctl** command

Authorization for both commands is provided through the resource profile:

```
EZB.NETMGMT.<NSS Server SYSID>.<NSS Server SYSID>.NSS.DISPLAY
```

Although both commands are used for the NSS IPSec discipline, the XMLAppliance discipline takes advantage only of the **nssctl** command.

Configuring the NSS server procedure and configuration file

In addition to the previous authorizations, 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, x.509 certificates for AT-TLS, and a SYSLOG daemon implementation. Optionally, its TCP/IP stack requires an IP Filtering policy and an IPSec implementation. If the IPSec discipline is to be supported, the NSS server also requires a key ring that contains the certificates for the IPSec clients. In this section, we explain these elements.

The NSS configuration file

The NSS server requires a configuration file that can be created using either of the following methods:

- Copy `nssd.conf` from `/usr/lpp/tcpip/samples` to `/etc/security`, and modify the copied file. The file assumes the EBCDIC IBM Code page IBM-1047, but you can change this value to suit your installation by using the environment variable named NSSD_CODEPAGE.
- Create the `nssd.conf` file with the Network Configuration Assistant.

In our testing, we initially chose to create the file manually with the sample file that is stored in the HFS. Later, we allowed the IBM Configuration Assistant to create the file so that we could compare the two methods.

Example 10-20 shows the pertinent parts of our NSS configuration file, which we stored in `/etc/security` as `nssd33.conf`.

Example 10-20 The /etc/security/nssd33.conf file created from nssd.conf

```
#
# 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   1
# SyslogLevel 0-255                    (dynamically modifiable)
...
  SyslogLevel 255                                                    2
...
# The NSS server attempts to open the configured key ring during
# startup. The key ring is used throughout the IPSec and XMLAppliance
# disciplines to locate certificates and/or private keys to be used for
# centralized cryptographic services.
# When using a key ring owned by the NSS server, specify only the
# ringname value. When using a key ring owned by another user, specify
# the ring name as a userid/ringname value.
# There is no default value. If KeyRing is not specified, then the
# NSS server cannot supply certificate services.
#
# This is the keyring holding the IKE certificates for IKED Server/Clients
# and any XMLAppliance certificates and private keys that we may decide to
# test with
KeyRing NSSD/NSSD_keyring 3

# Specifies a discipline that is enabled or disabled by the NSS server.
# Supported disciplines are:
#   IPSec -
#     Includes the IPSec certificate service and IPSec remote
#     management service. The default for the IPSec discipline is
#     'Enable'.
#   XMLAppliance -
#     Includes the XMLAppliance SAF access service, XMLAppliance private
#     key service, and XMLAppliance certificate service. The default for
#     the XMLAppliance discipline is 'Enable'.
# Default: IPSec Enable
# Default: XMLAppliance Enable
Discipline IPSEC Enabled 4
Discipline XMLAppliance Enabled 5
}
```

The numbers in this example correspond to the following information:

- We specify the port that the NSS server is to bind to and listen on when it is initialized at 1. Port 4159 is the default port.
- We specify a SYSLOG level of 255 at 2, which can help with problem determination. The default SYSLOG level is 1. We set the level quite high to ensure that we trap all possible messages for troubleshooting.
- We have also implemented IPSec clients for NSS, and therefore know the name of our IKE key ring: NSSD/NSSD_keyring at 3.

> **Tip:** The connection between the NSS server and the NSS client is secured with AT-TLS. For this connection we will specify a key ring name in the AT-TLS policy that holds the certificates to be used for SSL/TLS operations. However, the key ring named in the NSSD configuration file is used for a different purpose. It is used to hold not only certificates and private keys for IPSec clients (IKED), but also certificates and private keys for the private key and certificate services of the XMLAppliance discipline. Note that we are testing only the IPSec clients in this part of the chapter.

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 separate from the user ID associated with the NSS procedure. If you are not implementing IPSec clients, then any key ring name will suffice for this part of the configuration file.

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. This is true only if you do not want to preface the key ring name with the name of the OMVS user that is associated with the NSSD started task.

> **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. However, if the key ring is set up to be a shared key ring and it is populated with SITE certificates for the servers, then the owner of the key ring can be a USERID different from that under which NSSD or IKED is running.

- We intend to support both disciplines of the NSS server:
  - The IPSEC discipline (4)
  - The XMLAppliance discipline (5)

Setting BPX_JOBNAME and BPX_USERID

We decided to start the NSS server with JCL, although you might choose to start it from `/etc/rc`. The steps are documented in detail in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Tips: If you start the NSS server from the UNIX shell or from within `/etc/rc`, set the following environment variables:

```
# 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 set these environment variables, 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.

General Rule: Start 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 describe the various NSS server authorization tasks in “Configuring authorizations for the NSS and its users” on page 444.

SYSLOGD isolation for the NSS server and for the AT-TLS messages

This step is detailed in *z/OS Communications Server: IP Configuration Guide*, SC31-8775. To isolate NSSD messages in our testing, we created two entries in the `syslog.conf` file as shown in Example 10-21.

Example 10-21 NSSD entries message isolation for SYSLOG Daemon

```
*.NSS*.*.*                    /tmp/syslog/nssd.log      1
*.*.*.*;*.NSS*.*.none         /tmp/syslog/syslogd.log     2
```

The entry at **1** captures messages from any procedure that starts with the jobname of `NSS`. The entry at **2** captures all other messages (that is *.*.*.*, but not those already recorded for a jobname that begins with NSS (for example, *.NSS*.*.none). As a result, the AT-TLS messages display in the log named `syslogd.log`, but the messages resulting directly from the NSS server flow to the log named `nssd.log`.

NSS server environment variables

This is an optional step. You can specify the following variables in the environment variables:

- NSSD_CTRACE_MEMBER
- NSSD_FILE
- NSSD_PIDFILE
- NSSD_CODEPAGE

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 data set as shown in Example 10-22.

Example 10-22 NSS standard environment variables

```
NSSD_FILE=/etc/security/nssd33.conf
NSSD_CTRACE_MEMBER=CTINSS00
NSSD_CODEPAGE=IBM-1047
```

Update the NSS catalogued procedure

You can initialize the NSS daemon by using the following methods:

- From an MVS procedure

  Example 10-23 shows the modified sample that we copied from `hlq.SEZAINST(NSSD)` to our MVS procedure library.

  Example 10-23 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"',  1
//         '"_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   2
//* 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 the TCPIPE stack. Therefore, we established stack affinity to TCPIPE with the variable _BPXK_SETIBMOPT_TRANSPORT=TCPIPE, shown at 1 in Example 10-23. Our standard environment file at 2 is shown in Example 10-24.

  Example 10-24 Standard environment file for NSS server

```
NSSD_FILE=/etc/security/nssd33.conf
NSSD_CTRACE_MEMBER=CTINSS00
```

  We executed our NSS server with `PGM=NSSD`. Restrictions regarding Language Environment variables might require you to run the NSS server under BPXBATCH. More information about these restrictions is available in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

  Remember to copy member CTINS00 from SYS1.IBM.PARMLIB to the production PARMLIB if it is not already present.

- From the UNIX Shell or from within `/etc/rc`

  Remember to specify _BPX_JOBNAME if you use this method.
- From the COMMNDxx member in hlq.PARMLIB

Tip: 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. To allow PAGENT to manage starting, stopping, and monitoring the NSS server, see “Policy infrastructure management services” on page 120

Creating the AT-TLS certificates and key ring for the NSS server

We explain how to create this key ring and populate it with the IKE client certificates in “IKED certificate definition for NSS client” on page 348.

Tip: The key ring can be the same as the actual IKED key ring if it is populated with SITE certificates, or it can be a key ring that is 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 policy files at the NSS server and client nodes

You might need to consider the following types of policies for NSS communications:

- You must have an AT-TLS policy for the NSS connections between server and client.
- You might need an IP Security policy. If IPSec has been enabled at the IP stack under which NSS runs, you need to verify that the IPSec policy will allow communication between the NSS server and the NSS client, whether this communication is an NSS IPSec client or an NSS XMLAppliance client. If the IPSec or filtering policy does not permit such traffic, you need to build a policy that does permit communication between the NSS server and its clients.

PAGENT main and image files

Example 10-25 shows the contents of the main PAGENT file.

Example 10-25 Main PAGENT file (/etc/pagent.sc33.conf)

```
Loglevel 255
TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600
TcpImage TCPIPE /etc/pagent.sc33.tcpipe.conf  FLUSH PURGE 600
```

Example 10-26 shows the contents of the TCPIPE image file for policy agent.

Example 10-26 Image file for TCPIPE: /etc /pagent.sc33.tcpipe.conf

```
IPSecConfig /etc/pagent.sc33.tcpipe.ipsec.nssdp.policy
TTLSConfig  /etc/pagent.sc33.tcpipe.ttls.nssdp.policy 1
```

In this example, the number corresponds to the following information:

1. Shows how the TCPIPE stack is pointing to the AT-TLS policy (`/etc/pagent.sc33.tcpipe.ttls.nssdp.policy`) that is used for NSS on port 4159. Example 10-27 on page 462 shows a copy of the AT-TLS policy.

PAGENT AT-TLS policy

We created the AT-TLS policy in Example 10-27 with the IBM Configuration Assistant. It contains two rules at **1** and **2**. It shows the AT-TLS Policy, `/etc/pagent.sc33.tcpipe.ttls.nssdp.policy`, for the NSS Server to communicate with its clients. We explain how to create the second rule using the IBM Configuration Assistant in 10.2.4, “Creating NSS Server files for an NSS XMLAppliance Client with IBM Configuration Assistant” on page 465.

Example 10-27 AT-TLS Policy for the NSS Server to communicate with its clients

```
## AT-TLS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPE
##
## CA generated policy statements for TTLSRule(s) in support of NSS function.
##
TTLSRule 1                          NSSServerRuleAt-TLS~1 a
{
  LocalAddrSetRef                   addr1
  RemoteAddrSetRef                  addr1
  LocalPortRangeRef                 portR1
  RemotePortRangeRef                portR2
  Direction                         Inbound
  Priority                          255 b
  TTLSGroupActionRef                gAct1~NSS_Server c
  TTLSEnvironmentActionRef          eAct1~NSS_Server c
  TTLSConnectionActionRef           cAct1~NSS_Server c
}
TTLSGroupAction                     gAct1~NSS_Server
{
  TTLSEnabled                       On
  Trace                             255
}
TTLSEnvironmentAction               eAct1~NSS_Server
{
  HandshakeRole                     Server
  EnvironmentUserInstance           1
  TTLSKeyringParmsRef               keyR~SC33
}
TTLSConnectionAction                cAct1~NSS_Server
{
  HandshakeRole                     Server
  TTLSCipherParmsRef                cipher1~AT-TLS__Silver
  TTLSConnectionAdvancedParmsRef    cAdv1~NSS_Server
  Trace                             255
}
TTLSConnectionAdvancedParms         cAdv1~NSS_Server
{
  ApplicationControlled             On
  CertificateLabel                  CS19 ITSO SharedSite1 d
}
TTLSKeyringParms                    keyR~SC33
{
  Keyring                           TCPIP/SharedRing1 e
}
TTLSCipherParms                     cipher1~AT-TLS__Silver
```

```
{
  V3CipherSuites                  TLS_RSA_WITH_DES_CBC_SHA
  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                            4159
}
PortRange                         portR2
{
  Port                            1024-65535
}

##
## CA generated policy statements for TTLSRule(s) in support of NSS function.
## NSS Client keyring for TLS is at bottom.
##

TTLSGroupAction 2                 NSS~TLS~ON
{
   TTLSEnabled                    On
   Trace                          15
}
TTLSCipherParms                   NSS~CIPHER~PARMS
{
   V3CipherSuites                 TLS_RSA_WITH_3DES_EDE_CBC_SHA
}
TTLSEnvironmentAction             NSS~TLS~SERVER~ENV
{
   HandShakeRole                  Server
   TTLSCipherParmsRef             NSS~CIPHER~PARMS
   TTLSKeyRingParms
   {
      Keyring                     TCPIP/SharedRing1
   }
}
TTLSRule                          NSS~TLS~SERVER
{
   LocalPortRange                 4159
   Direction                      Inbound
   TTLSGroupActionRef             NSS~TLS~ON
   TTLSGroupActionRef             NSS~TLS~ON
   TTLSEnvironmentActionRef       NSS~TLS~SERVER~ENV
}
```

In this example, the numbers and letters correspond to the following information:

- 1 This is the rule that will take precedence over the second AT-TLS policy (2), because this rule has a higher priority of 255 (indicated at point b).
- a The name of the rule is *NSSServerRuleAt-TLS~1*.

- b This rule has a high priority: 255. As such, if there are competing rules, the rule with the higher priority will be used to apply the policy.
- c These are the names of the actions associated with the rule.
- d The server certificate label that the NSS server should use was specified in the rule with its actions.
- e The name of the key ring owner and the key ring are specified here.

PAGENT IP filtering and IPSec policies

If IP filter policies or IPSec policies are required for the TCP/IP stack on which NSS is executing, consult Chapter 7, “IP filtering” on page 217 and Chapter 8, “IP Security” on page 245 on how to produce these policies.

Update the TCP/IP profile

To update the TCP/IP profile, complete the following tasks:

a. Reserve the NSS port in the TCP/IP profile. We used the default of port 4159 as shown in Example 10-28 at 1.

Example 10-28 Reserving the NSS port

```
PORT
....
       4159 TCP NSSD 1;
      16310 TCP PAGENT             ;
....
```

2. Enable the TCP/IP stack for NSS for AT-TLS with the TCPCONFIG statement and TTLS parameter. Follow the guidelines for AT-TLS enablement as described for TN3270 in Chapter 16, “Telnet security” on page 677 or for FTP in Chapter 17, “Secure File Transfer Protocol” on page 719. Example 10-29 shows the appropriate TCPCONFIG parameter for AT-TLS.

Example 10-29 AT-TLS stack enablement

```
TCPCONFIG TTLS
```

Tip: You might 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, you might also want to update the IPSec default rules in the TCP/IP profile as shown in Example 10-30. Consult 8.5.3, “Updating the TCP/IP stack to activate IPSec” on page 266 for information about how to accomplish this task.

Example 10-30 Updating the TCP/IP profile for IP Security on behalf of NSS

```
IPCONFIG ... IPSECURITY ... 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  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
 IPSECRULE  * * LOG  PROTOCOL TCP SRCPORT 4159 DESTPORT * 2 ; Network Security
;IPSEC6RULE * * LOG  PROTOCOL TCP SRCPORT 4159 DESTPORT * 3 ; Network Security
;
ENDIPSEC
```

At 1 in Example 10-30 on page 464, we enabled IPSECURITY on the IPCONFIG statement. At 2, we included default IPSEC rules on behalf of the NSS server for IPv4. At 3, we commented out the IPSEC6RULE because we did not enable IPv6 security in this stack.

Tip: You might 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.

10.2.4 Creating NSS Server files for an NSS XMLAppliance Client with IBM Configuration Assistant

We explained how to configure the NSSD configuration file and AT-TLS policy in 10.2.3, “Configuring the NSS environment at z/OS” on page 443. In this section, we show how to perform the same configuration tasks using the IBM Configuration Assistant.

In defining NSS with the windows, you need to enter the names of two types of key rings:

- A key ring for the AT-TLS-secured connection between NSSD and the NSS client
- A key ring to hold the certificates that are necessary for an IKED and for the XMLAppliance client that is using the private key and certificate services of NSSD

Key ring design philosophy

Prior to our discussion about the IBM Configuration Assistant GUI, we include this brief discussion of the philosophies for managing keyrings.

In Chapter 9, “Network Security Services for IPSec clients” on page 325, we used two separate key rings:

- One key ring for the NSS IPSec client and server certificates
- A separate key ring for the SSL/TLS connection between NSS Server and the IKE clients (for example, NSS IPSec clients)

We could have used a single key ring for the AT-TLS certificates and the IKED certificates, but we chose to use separate key rings to maintain some granularity and to distinguish between the two types of key ring usage.

When NSS exploits the private key or certificate service on behalf of XMLAppliance clients, the server needs access the certificates for the SSL/TLS flows between client and server, and any certificates and private keys that are centrally managed for the XMLAppliance clients. As with the operations with the NSS IPSec client, NSS can be configured with one key ring for all certificate implementations or with a key ring for secured operations and a key ring for the operations that are required by the XMLAppliance private key services and certificate services clients.

You can choose the key ring design based on your philosophy for designing key ring contents and on the capabilities of servers and clients to select certificates by label name and not solely by their designation as *default*. Consult Chapter 3, “Certificate management in z/OS” on page 39 for more information about this subject. Also, see Chapter 9, “Network Security Services for IPSec clients” on page 325 for a discussion about several strategies that you can use to manage key rings.

Working with the IBM Configuration Assistant to define NSS

After you initialize the IBM Configuration Assistant on the workstation, follow these steps:

1. Open the Help windows for NSS by clicking **Help** → **NSS Overview** from the Main Perspectives windows of the GUI, which opens the window shown in Figure 10-8. From this window, you can read through the overview, or you can proceed to other selections in the left navigation window to learn how to code for NSS server and client images.

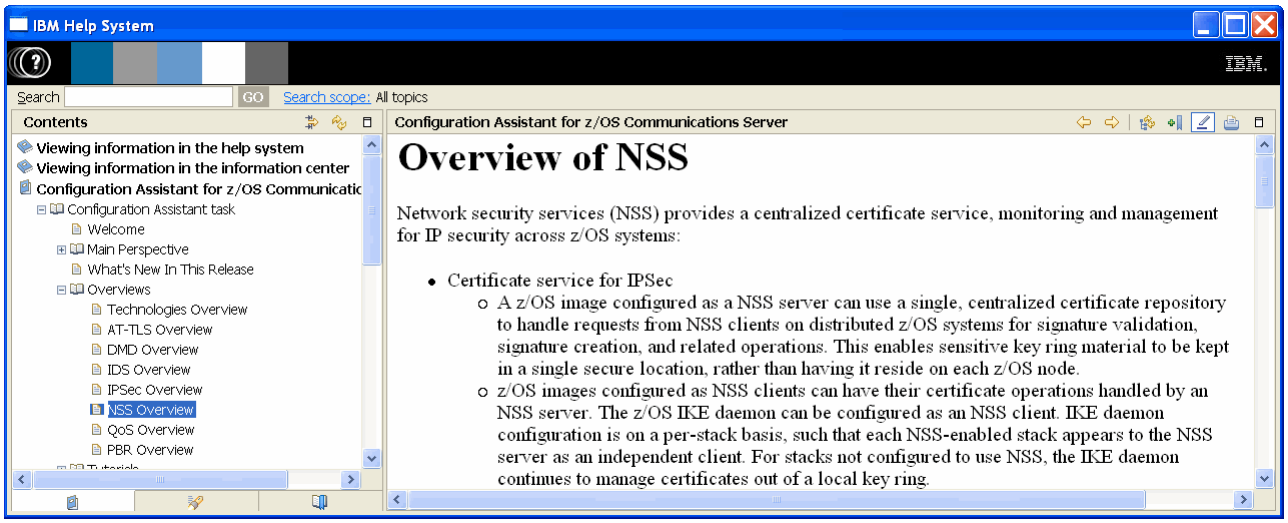


Figure 10-8 Overview of NSS in IBM Configuration Assistant Help

2. If you want to create the NSS configuration file using the IBM Configuration Assistant, select or create the image on which the NSS resides. Enable NSS on this image, as we did for MVS image SC33 in Figure 10-9.

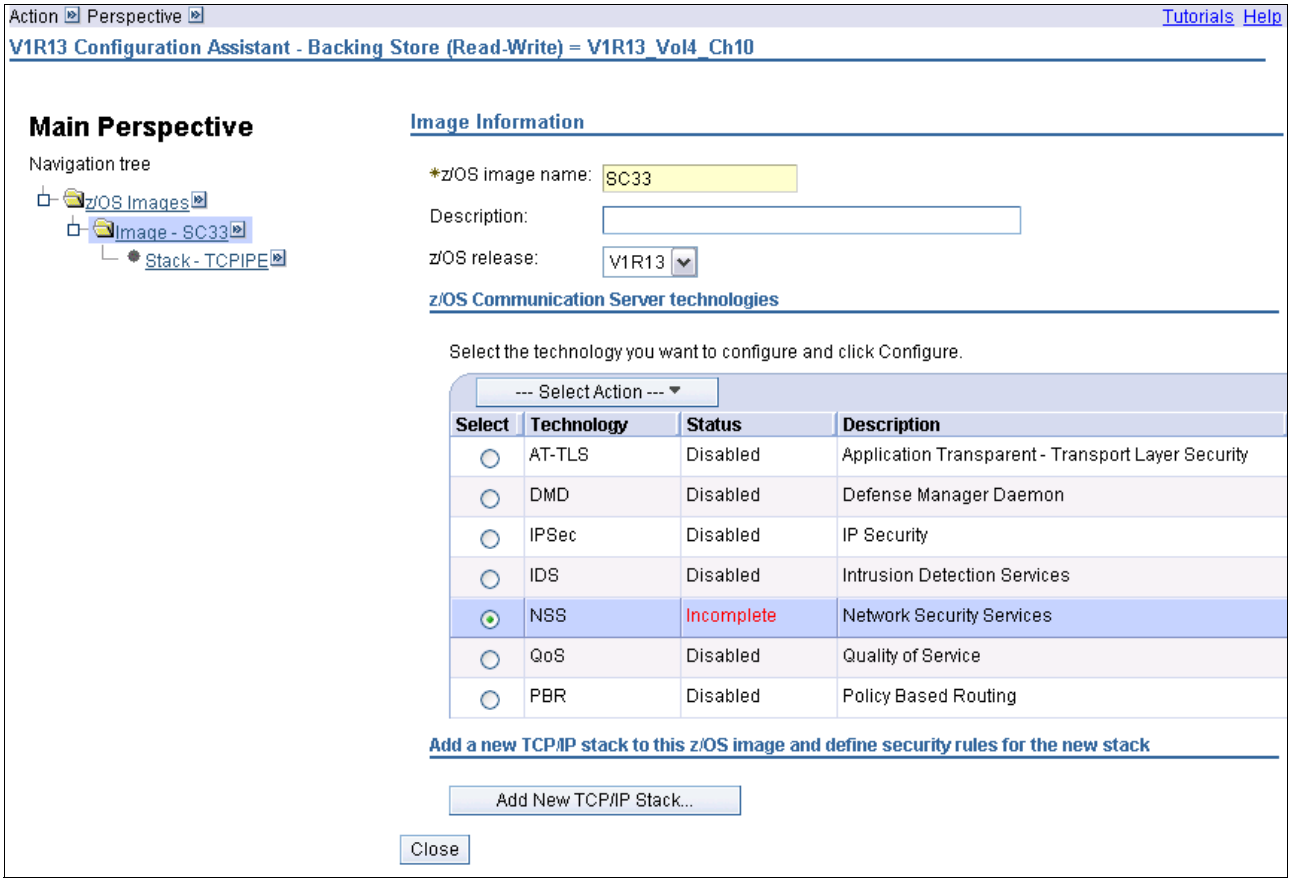


Figure 10-9 Enabling NSS at SC33

Notice that the SC33 image in our scenario has a status of `Incomplete` because we need to configure NSS on this image.

3. Click **Configure** to open the window shown in Figure 10-10. Select **Server** as the role for this MVS image, and click **Next**.

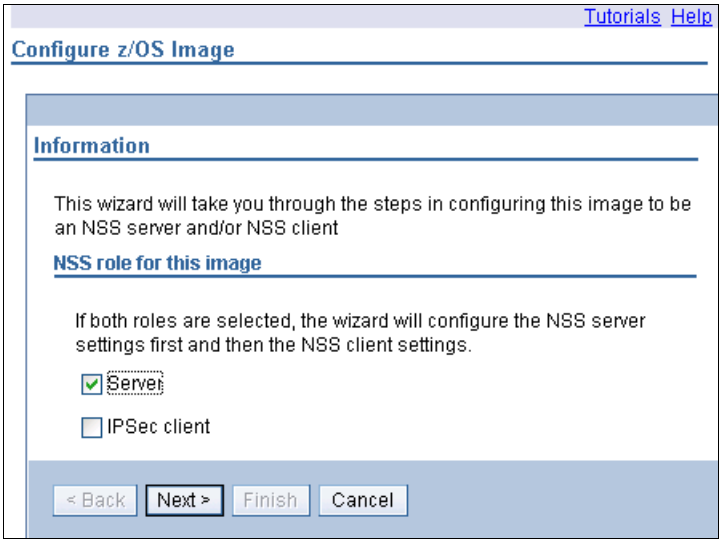


Figure 10-10 Selecting the NSS role of server for this MVS image

4. In the next window, shown in Figure 10-11 on page 469, enter the following information:
   - Port and key ring settings
     - The listening port for the NSS server is 4159 and is required for any type of NSS client relationship.
     - The NSS server key ring is named NSSD/NSSD_keyring. This information is used for certificates that are necessary for the IPSec discipline and for certificates that an XMLAppliance client might need for the private key service and the certificate service.
   - Default server settings at the MVS image level for the server address

     This information can be overridden for individual clients of the IPSec and the XMLAppliance disciplines.
   - Default server settings at the MVS image level for the server identity

     This information can be used by the NSS client to verify that the certificate it receives from the server matches an identity coded at the client. It is typically used for IKED clients rather than for NSSXML clients. We used the same information that we configured in Chapter 9, “Network Security Services for IPSec clients” on page 325 for the RSA signature mode authentication process.
   - Automatically permit and protect NSS traffic

     The AT-TLS key ring for the NSS server is TCPIP/SharedRing1. This information is required for the secured communications using AT-TLS between the NSS server and both types of clients:
     - Clients for the IPSec Discipline
     - Clients for the XMLAppliance Discipline

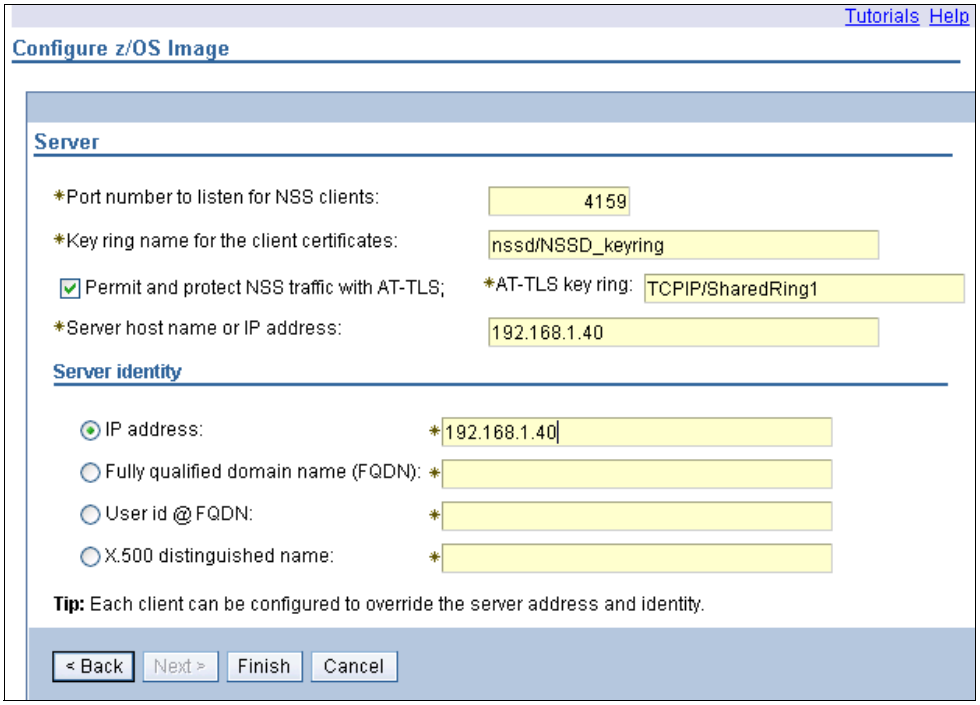


Figure 10-11 NSS configuration values

Tip: If you use the IBM Configuration Assistant to create the definitions to communicate with an NSS XMLAppliance client, the GUI syntax checker requires that you supply information for all four blocks of information, whether or not a particular discipline and its services require the information.

5. When you complete this information, click **Finish**.
6. In the NSS Perspective window, select the MVS image that you are working on (we used SC33), then click **Application Setup Tasks**.

7. Complete the tasks as directed in the Application Setup Tasks window, shown in Figure 10-12, to finish the NSS implementation.

> **Tip:** The policy agent AT-TLS policy task for the IPSec discipline and the XMLAppliance discipline is missing from the list shown in this figure. This is a known requirement.

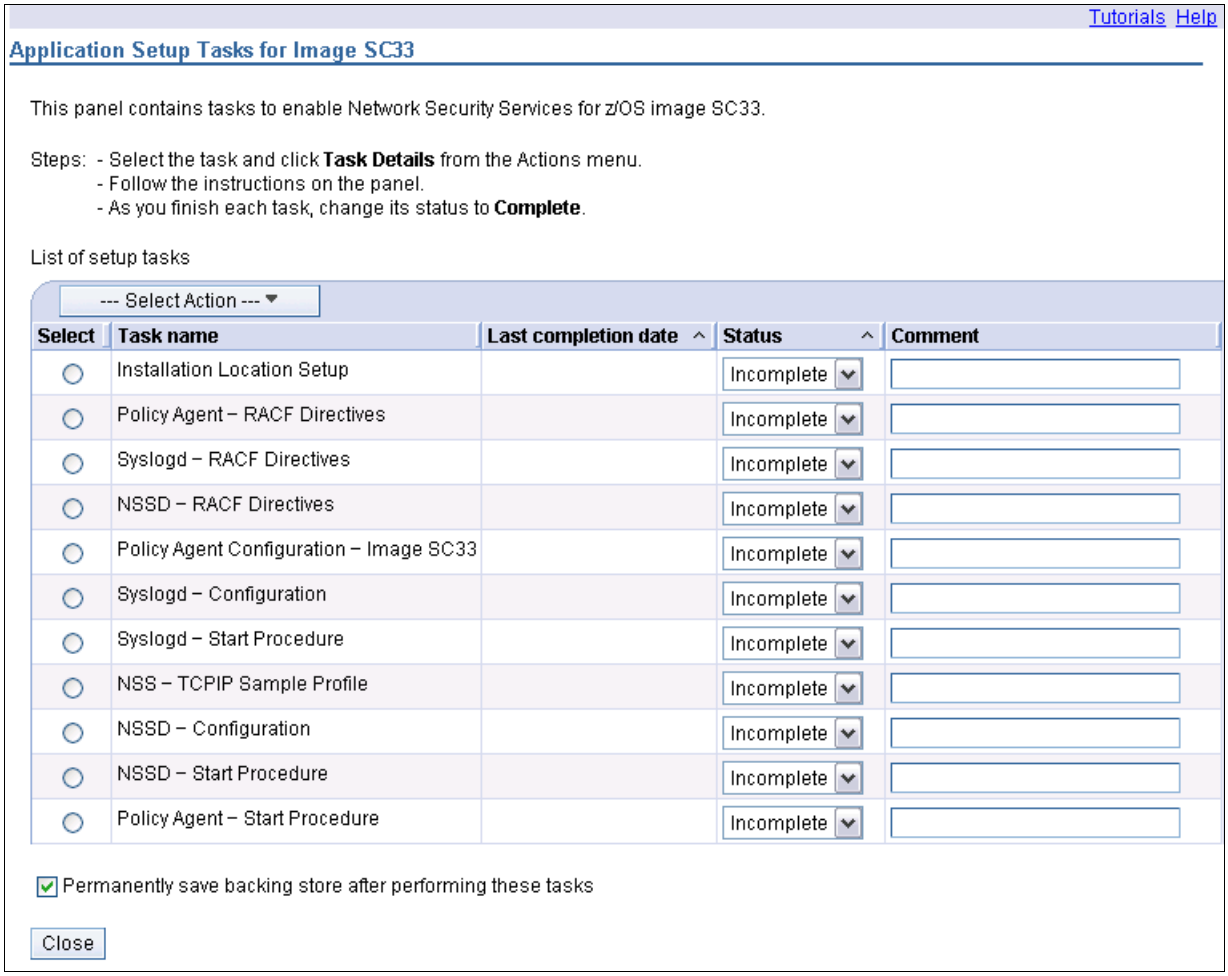


Figure 10-12 Application Setup Tasks for Image SC33

Most of these tasks are outlined in the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

> **Tip:** As you select each task you are given the opportunity to review the setup tasks and then to mark them as complete, making this listing a good task plan for your work on implementing NSSD.

It is worthwhile to look at all of these tasks and their information. However, in this book, we review only the details for the NSSD - RACF Directives tasks shown in Figure 10-12.

8. Highlight these directives, and select **Task Details**. The RACF Directives task window opens (Figure 10-13). You can click **Instructions** to view the steps for completing this task. You can also view the contents of the directives file by clicking **View Contents**.

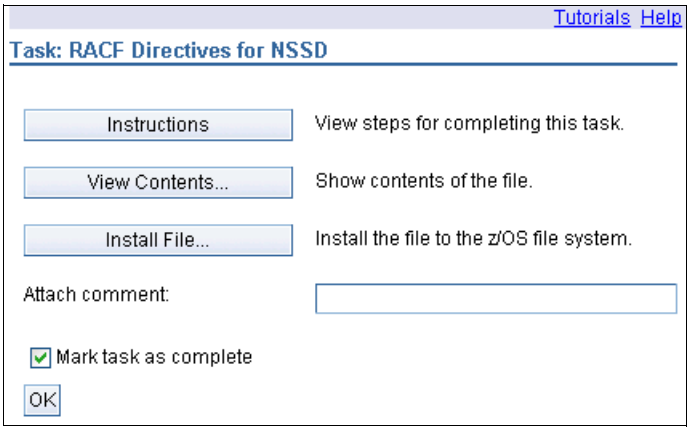


Figure 10-13 Review of the task to complete the RACF directives for NSSD

When you click **View Contents**, the RACF directives Sample Job is displayed as shown in Figure 10-14.

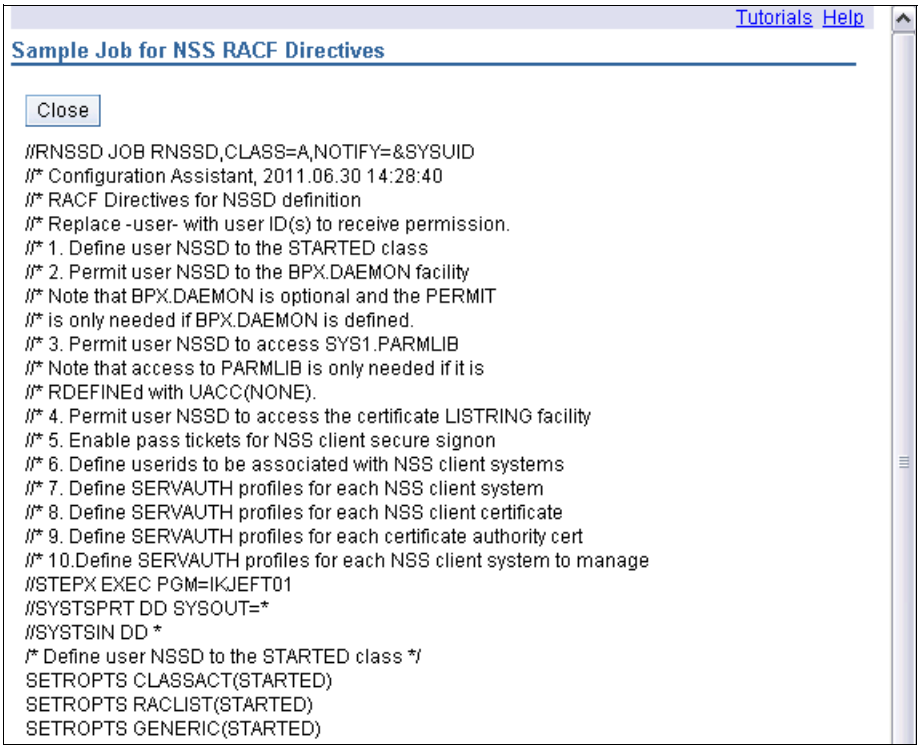


Figure 10-14 JCL to define the RACF directives for an NSS implementation

You can print the directives to give to your RACF administrator, save them to the workstation for further editing, or just close the window so that you can use FTP to copy the JCL to the System z host.

9. Click **Close** to exit this window. Mark the task as *Complete*, and click **OK**.

10. You now see a date and time for the completion of the task. You can use this task list to keep track of the progress that you make in defining the prerequisites for the NSS function as shown in Figure 10-15.

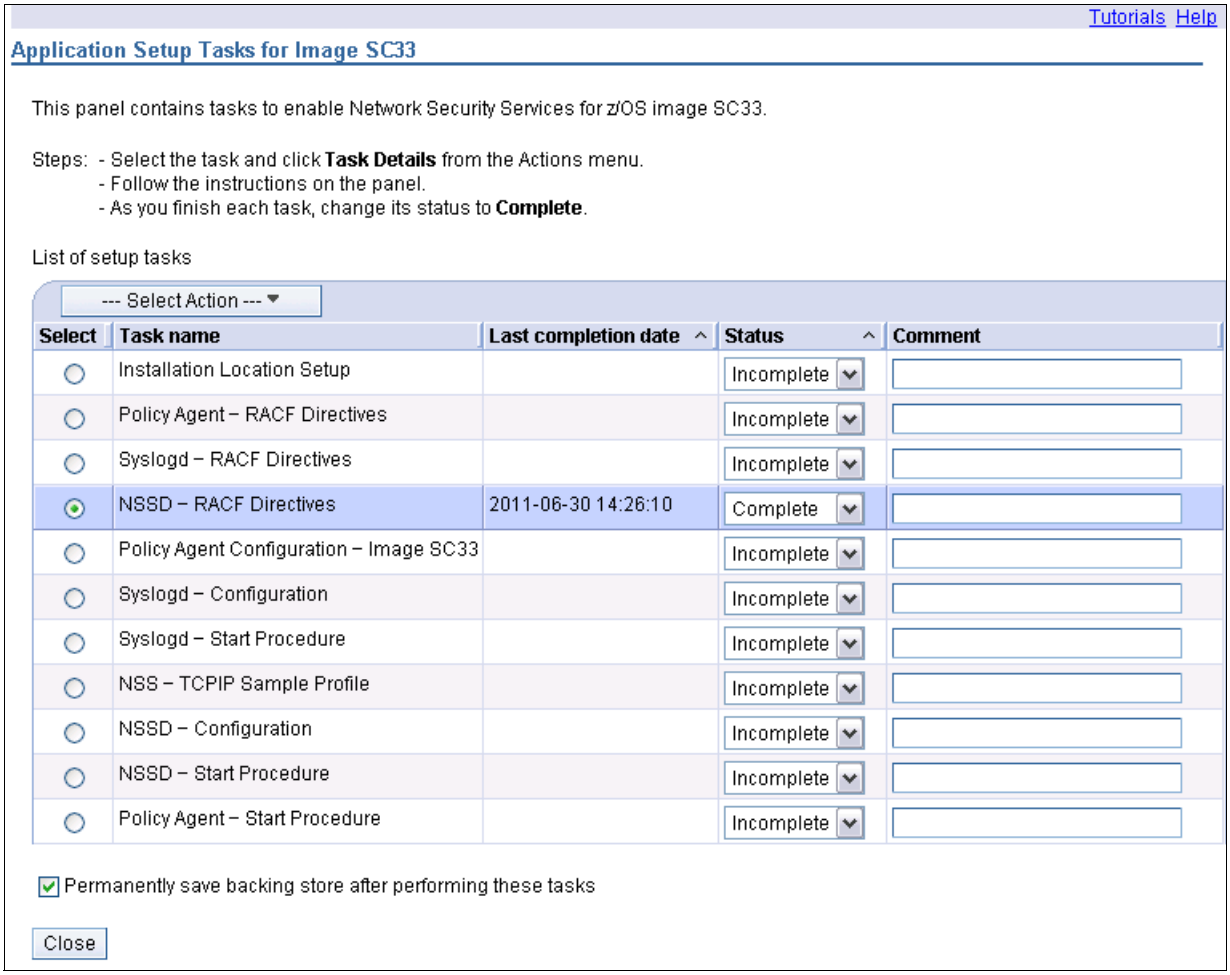


Figure 10-15 Application setup tasks: View of completed task

11. Click **Close** to exit the window.
12. Include any comments in the history log, and click **OK**.
13. Highlight the TCP/IP stack in the left navigation bar, and, if the tool informs you that the stack is still not enabled for NSS, click **Enable NSS** at the upper right of this window.
14. In the NSS Perspective window, select NSS from the left pane and click **NSS Summary**. Depending on whether this backing store file contains policies and descriptions for an NSS client, you might or might not see a list of possible NSS clients. You will, however, see the description of the NSS server that you have created.

15. Go to the Main Perspective, which shows that NSS is now enabled at the SC33 image. However, note that the AT-TLS, which is necessary for the secured connection between SC33's TCPIPE and the NSSXML client, is not yet enabled. See Figure 10-16.

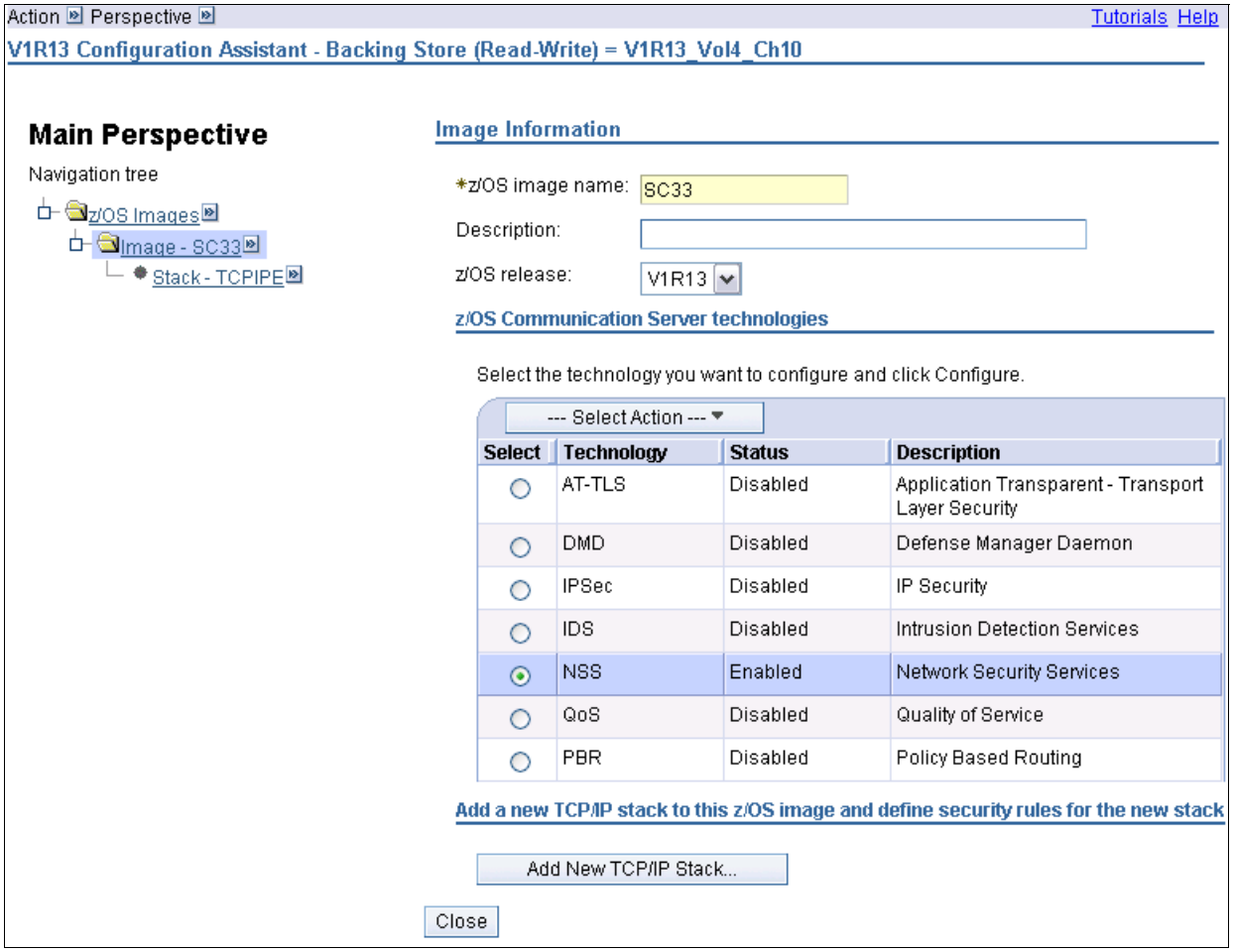


Figure 10-16 MVS image enabled for NSS, but not yet enabled for AT-TLS

AT-TLS policy

To enable the AT-TLS policy, complete the following steps:

1. Highlight AT-TLS for the SC33 image, and select **Enable**. The image changes to a status of *Incomplete*. Next, click **Configure** to proceed with the AT-TLS policy for the secured NSSD-NSS client connection. In the Required AT-TLS Image Level Settings window shown in Figure 10-17, enter the name of the AT-TLS key ring that you will use for the NSS connections, and click **OK**.

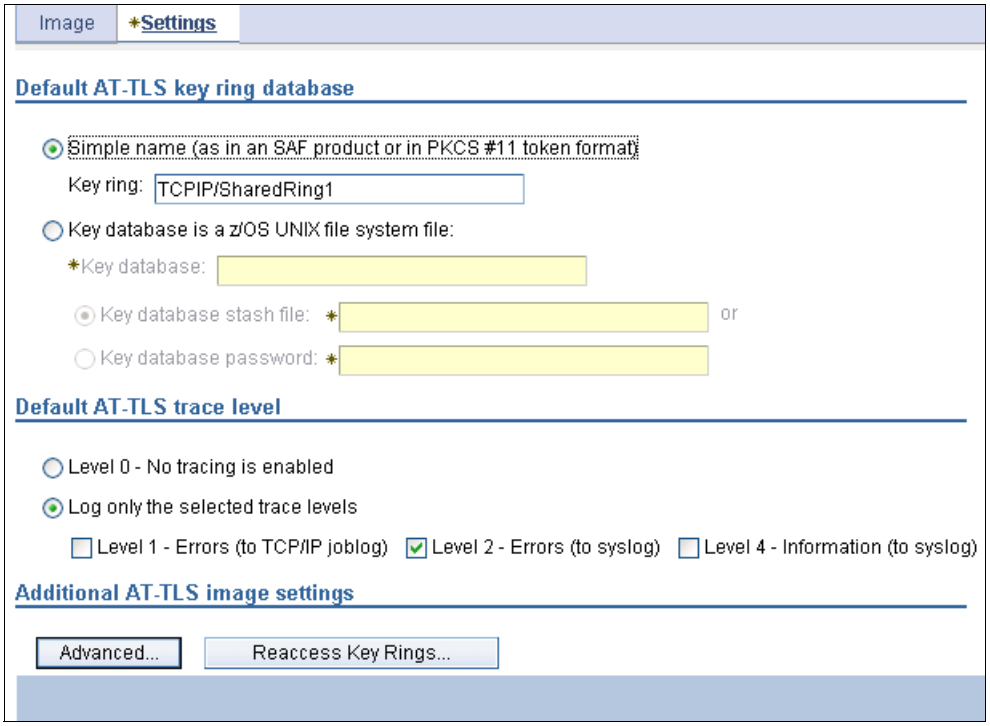


Figure 10-17 Name of the key ring or key database

2. In the AT-TLS perspectives window, highlight the TCPIPE stack underneath the MVS image. Observe that the stack is not enabled for AT-TLS yet. Click **Enable AT-TLS**.
3. The AT-TLS Perspective window opens. The Navigation tree on the left side of this window shows that this stack is still an *Incomplete Stack*.

4. Select the AT-TLS rule that you want to enable for the stack. Notice, we highlighted the `Default_NSS_Server` rule as shown in Figure 10-18. Right-click this rule and select **Enable Rule**.

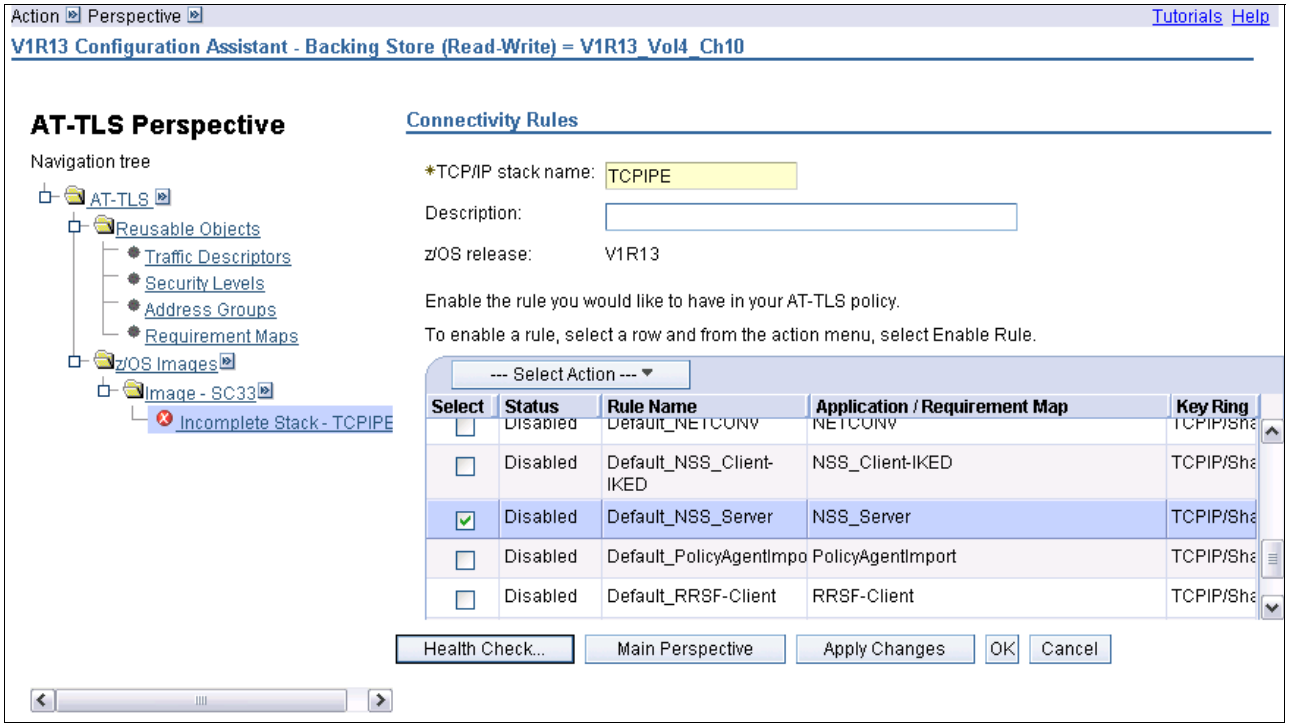


Figure 10-18 Enabling NSS TLS rule

5. When prompted, click **OK** in the Verify Rule message window shown in Figure 10-19.

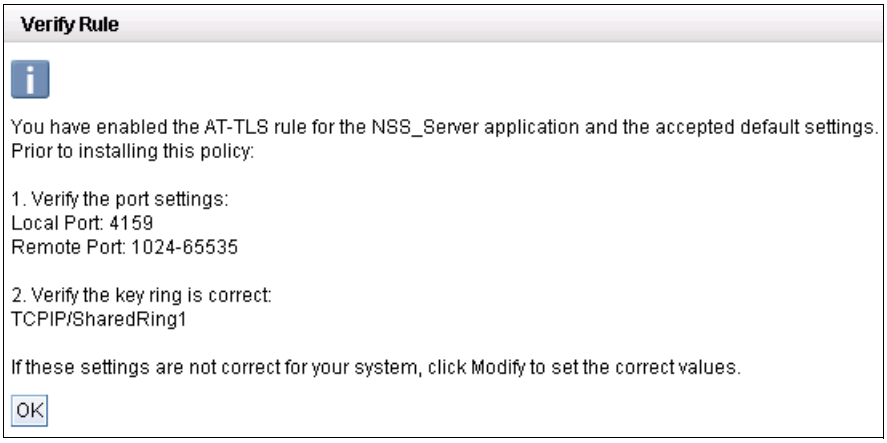


Figure 10-19 After AT-TLS rule enablement: Verify and possibly modify the default settings

6. Click **Apply Changes**. The Navigation tree no longer indicates that the AT-TLS rules for the stack are incomplete.
7. Click **Modify** to view all the definition values that we created for our NSS implementation.

8. In the Modify Rule window, shown in Figure 10-20, notice that the values on each of the tabs are acceptable. Click **OK**. Then, on the AT-TLS Perspective window, click **OK** again.

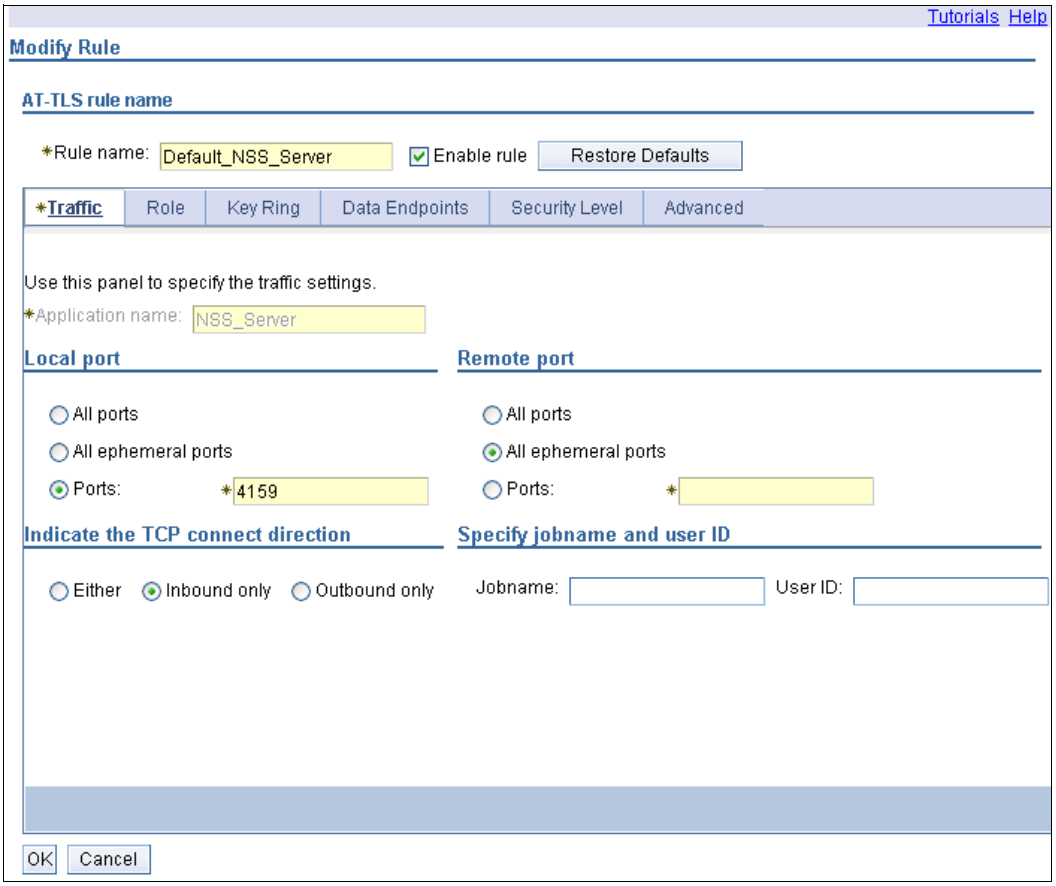


Figure 10-20 Reviewing default NSS Server rule

9. In the AT-TLS Perspective, click **Application Setup Tasks** to view of all the related AT-TLS tasks.
10. Complete all the tasks as shown in Figure 10-21.

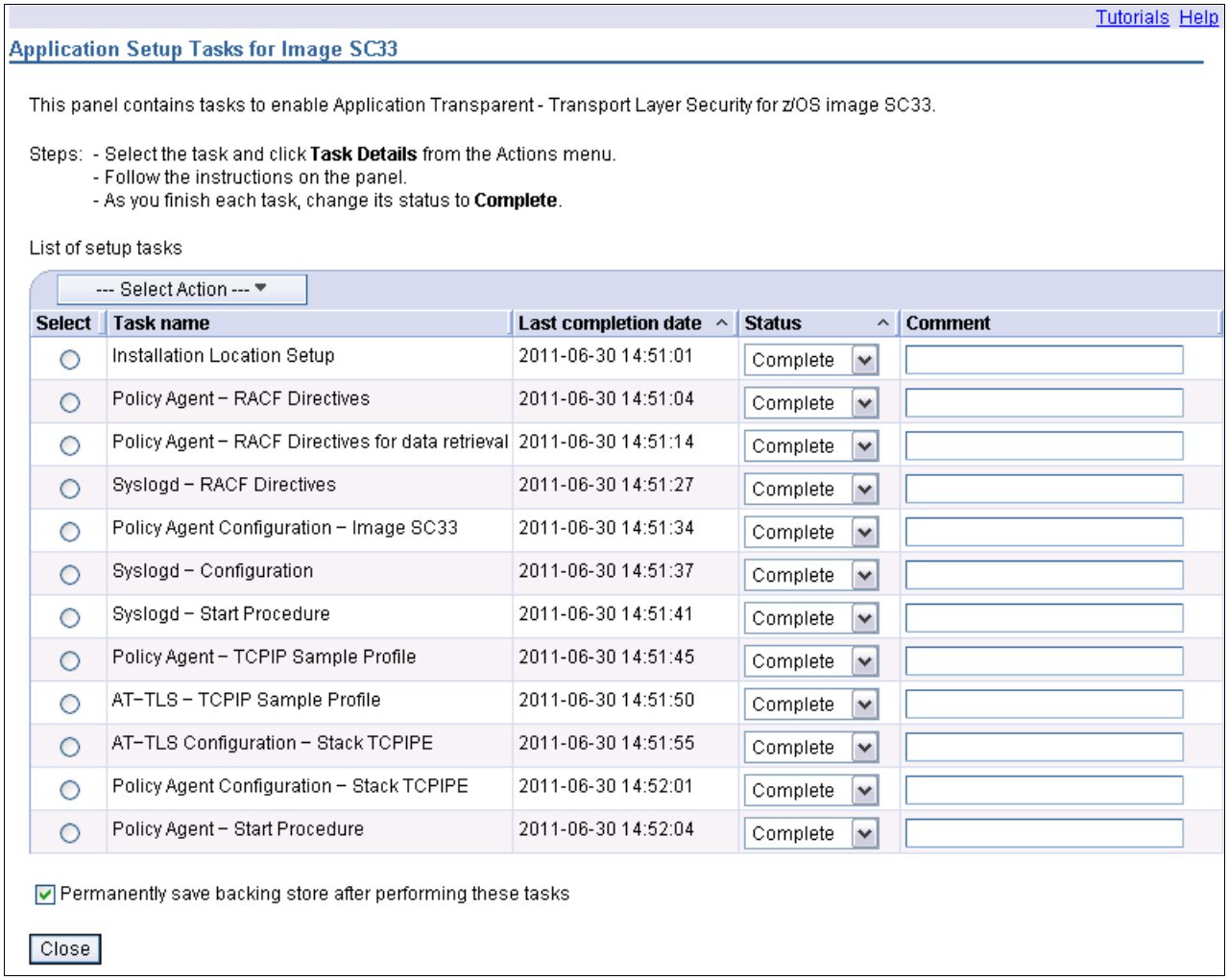


Figure 10-21 Application Setup Task for AT-TLS on behalf of NSS

- Examine the various options under the Task Details for the Policy Agent - RACF Directives. As the *z/OS Communications Server: IP Configuration Guide*, SC31-8775 also advises, this task detail reminds you to define the SERVAUTH class profile for INITSTACK and gives you the RACF commands to create this profile.
- Examine the various options under Task Details for the AT-TLS TCPIP Sample Profile. As the *z/OS Communications Server: IP Configuration Guide*, SC31-8775 also advises, this task detail reminds you to define the TCPCONFIG statement in the TCP/IP profile that enables AT-TLS.
- Examine the various options under Task Details for the AT-TLS Policy for NSS. This task shows you the contents of the AT-TLS policy file for the NSSD-NSS client connection, as shown in Example 10-31 on page 478.

Example 10-31 AT-TLS Policy for the NSS server connections

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPE
.............................................................. Lines
deleted
## TLS default rules: Default_NSS_Server
## End TLS default rules
##
## End of Configuration Assistant information
TTLSRule                          Default_NSS_Server~1
{
  LocalAddr                       ALL     1
  RemoteAddr                      ALL
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Direction                       Inbound
  Priority                        255
  TTLSGroupActionRef              gAct1~NSS_Server
  TTLSEnvironmentActionRef        eAct1~NSS_Server
  TTLSConnectionActionRef         cAct1~NSS_Server
   }
TTLSGroupAction                   gAct1~NSS_Server
{
  TTLSEnabled                     On
}
TTLSEnvironmentAction             eAct1~NSS_Server
{
  HandshakeRole                   Server     2
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~SC33
}
TTLSConnectionAction              cAct1~NSS_Server
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~Default_Ciphers
  TTLSConnectionAdvancedParmsRef  cAdv1~NSS_Server
  CtraceClearText                 Off
  Trace                           2
}
TTLSConnectionAdvancedParms       cAdv1~NSS_Server
{
  ApplicationControlled           On
  SecondaryMap                    Off
}
TTLSKeyringParms                  keyR~SC33
{
  Keyring                         TCPIP/SharedRing1     3
}
TTLSCipherParms                   cipher1~Default_Ciphers     4
{
  V3CipherSuites                  TLS_RSA_WITH_AES_256_CBC_SHA
  V3CipherSuites                  TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  V3CipherSuites                  TLS_DH_RSA_WITH_AES_256_CBC_SHA
```

```
  V3CipherSuites                  TLS_DHE_DSS_WITH_AES_256_CBC_SHA
     V3CipherSuites                  TLS_DH_DSS_WITH_AES_256_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
  V3CipherSuites                  TLS_DHE_RSA_WITH_AES_128_CBC_SHA
  V3CipherSuites                  TLS_DH_RSA_WITH_AES_128_CBC_SHA
  V3CipherSuites                  TLS_DHE_DSS_WITH_AES_128_CBC_SHA
  V3CipherSuites                  TLS_DH_DSS_WITH_AES_128_CBC_SHA
}
PortRange                         portR1
{
  Port                            4159     5
}
PortRange                         portR2
{
     Port                            1024-65535
}
```

In this example, the numbers correspond to the following information:

- The name of the rule (1), which indicates that we enabled the IBM default rule.
- This rule defines the SSL handshake role as `Server` (2).
- Identify the SSL/TLS key ring (3).
- Identify the list of default ciphers (4), from which one is selected for encrypting the traffic flows.
- Identify the port to which the client connects (5).

11. When you complete these tasks and the object that they create, exit the window by clicking **Close**.

12. Back in the AT-TLS Perspective window, select the MVS image (we use SC33) as shown in Figure 10-22 and click **Install Configuration Files**. Use FTP to copy the AT-TLS policy to the System z host. Incorporate the policy into the policy agent configuration and test.

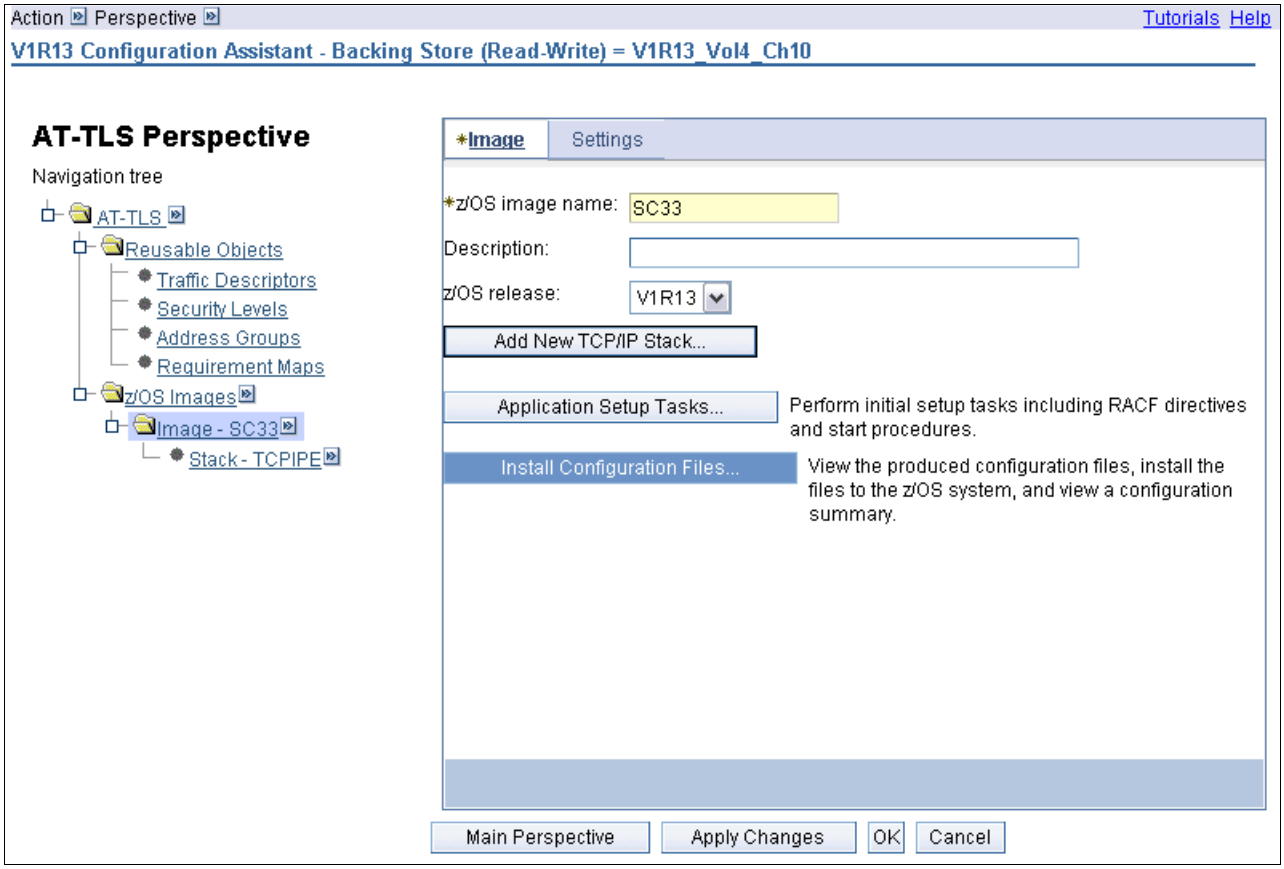


Figure 10-22 The AT-TLS Perspective window

Tip: If IP filter policies or IPSec policies are required for NSS, consult Chapter 7, “IP filtering” on page 217 and Chapter 8, “IP Security” on page 245 for information about how to produce the policies.

10.2.5 Configuring the NSS environment at the WebSphere DataPower SOA Appliance to support the SAF access service

Figure 10-23 represents the portion of the entire NSS environment that includes the NSS XMLAppliance client in the DataPower SOA appliance.

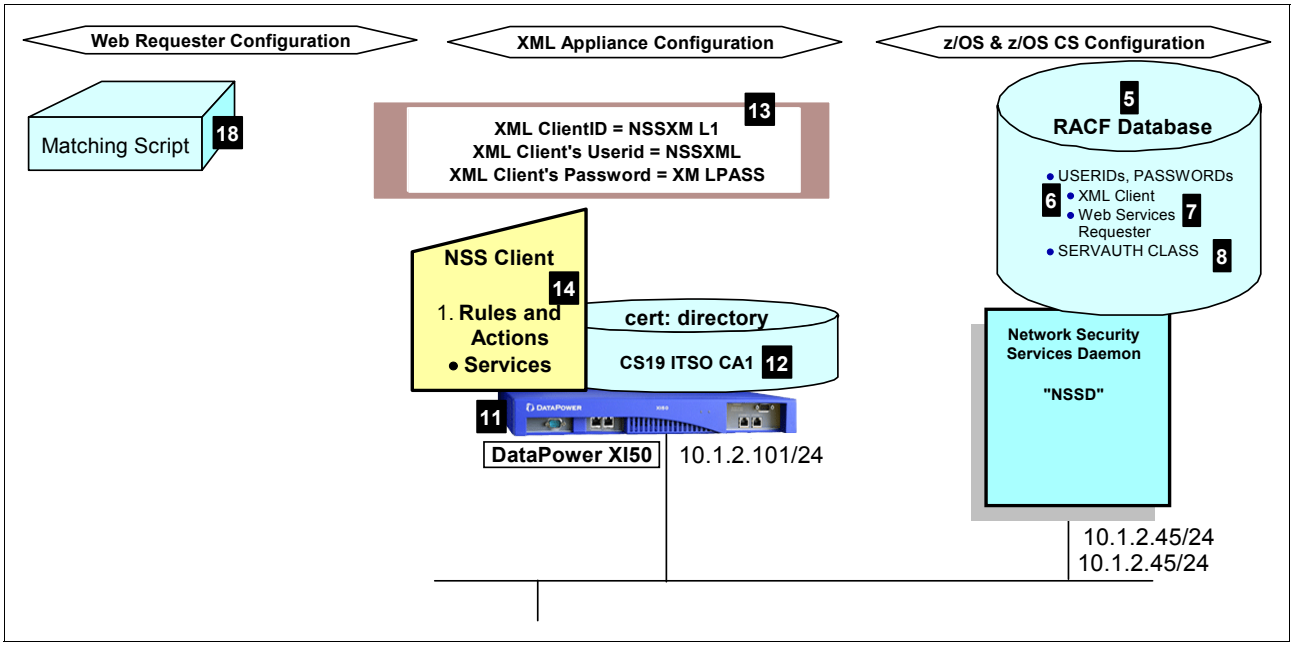


Figure 10-23 Client Configuration Details for an NSS implementation

We performed the following steps when we connected and tested the DataPower appliance with NSS:

1. DataPower SOA Appliance Customization
   a. Customize the DataPower appliance to connect to the network and System z.
   b. Create the NSS Client configuration in the appliance.
   c. Create the crypto profile and key ring in the appliance. Incorporate the CA certificate that signed the NSS server certificate into the key database.
   d. Create an Authentication, Authorization, and Audit (AAA) rule for testing the integration of the DataPower appliance with the NSS server.
2. Verification of NSS Server with NSS XMLAppliance Client
   a. Establish secure connection between NSS Server and NSS XMLAppliance Client
   b. Issue NSS commands to verify state of the connection.

Logging in to the management interface for DataPower appliance

We used the DataPower Web GUI to perform the administration tasks for the DataPower appliance. When you connect to the web GUI using the management port number that is assigned by the DataPower administrator, the login window shown in Figure 10-24 opens.

Figure 10-24 The XI50 WEB GUI console login window

Log in to the default domain with the user ID and password that the DataPower administrator provides. In our testing, we later switched to a new domain, *ITSO*, that the administrator created for us to isolate our work from the work of others. When you log in, the Control Panel window opens as shown in Figure 10-25.

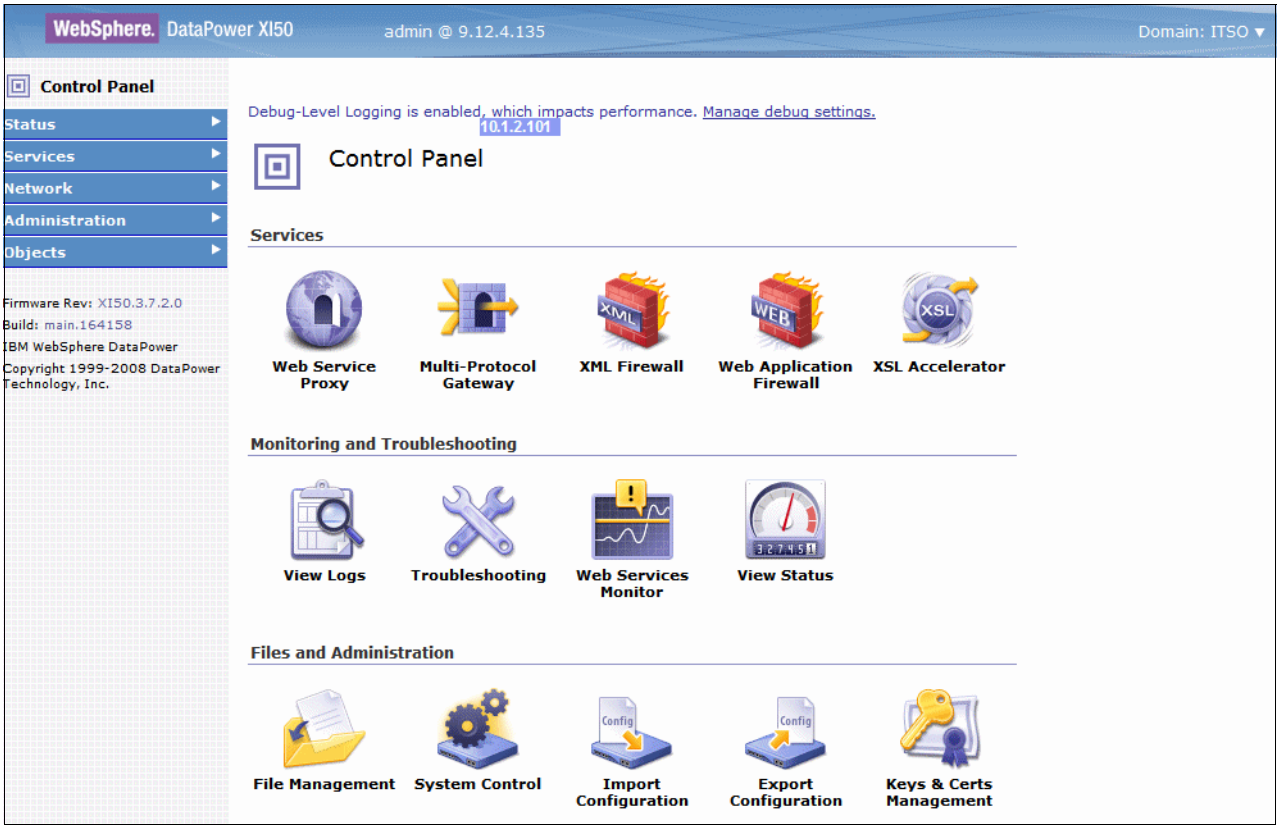


Figure 10-25 Control Panel window and navigation bar of the XI50 DataPower appliance

The navigation bar is divided into multiple areas:

- Status
- Services
- Network
- Administration
- Objects

When you configure the DataPower appliance for NSS integration on System z, you use several of these control window functions.

You need to complete the following sets of definitions on the DataPower SOA appliance:

- The Cryptographic configuration
- The SSL Proxy Profile object
- The NSS client configuration
- The XML firewall configuration

> **Tip:** For our configuration, we employed the XML firewall. However, in lieu of the firewall configuration, you might want to implement the Multi-Protocol Gateway configuration.

The Cryptographic objects configuration

The cryptographic security configuration for the NSS client requires the creation of several profile objects as shown in Figure 10-26.

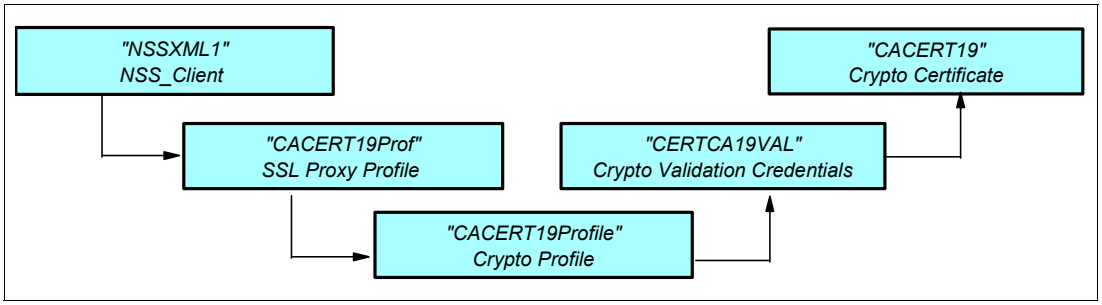


Figure 10-26 Cryptographic profiles for the NSS XMLAppliance Client (DataPower XI50)

The NSS client must be associated with a certificate directory that contains at least the CA certificate that has signed the NSS server certificate. Figure 10-26 shows that our *NSS XMLAppliance client* is associated with an *SSL Proxy Profile*, which identifies a *Crypto Profile*, which then points to a set of *Crypto Validation Credentials*, in which the *Crypto Certificate* itself (that is, the x.509 certificate) is named.

You can configure the objects in Figure 10-26 in any order. However, if you start by defining the Crypto Certificate on the right side of the visual, your configuration steps are executed more efficiently. With this sequence of steps, you create the prerequisite definitions for each configuration object as you proceed. Otherwise, you might have to return to the configurations and modify them to enter the names of any missing prerequisite objects.

We start the configuration process by defining the Crypto Certificate Object.

Before you begin the configuration on the DataPower appliance, export the z/OS CA certificate that resides on the NSSD key ring into a file in CERTB64 format. You need to load this ASCII file into the DataPower appliance later. In our testing, the CA certificate that signed the server certificate that the NSS server presents to the NSS Client during SSL negotiation was *CS19 ITSO CA1*. We exported the certificate from the z/OS RACF database in BASE64 encoding (ASCII) with the JCL shown in Example 10-32 at z/OS SC33.

Example 10-32 Exporting a certificate in ASCII format to use FTP to copy to another platform

```
//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  *
//*     certificates when passed in the SSL exchange.                 *
//*********************************************************************
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  racdcert certauth export(label('CS19 ITSO CA1')) -
    format(certb64) dsn('TCPIP.CS19.CACERT')
/*
```

Example 10-33 shows the data set that contains the certificate.

Example 10-33 ASCII (Base64 encoding) certificate output from an export job

```
-----BEGIN CERTIFICATE-----
MIICkzCCAfygAwIBAgIBADANBgkqhkiG9w0BAQUFADBMMQswCQYDVQQGEwJVUzEY
MBYGA1UEChMPSUJNIENvcnBvcmF0aW9uMSMwIQYDVQQLExpJVFNPIENlcnRpZmlj
YXRlIEF1dGhvcml0eTAeFw0wODExMTAwNTAwMDBaFw0xMjExMTEwNDU5NTlaMEwx
CzAJBgNVBAYTAlVTMRgwFgYDVQQKEw9JQkOgQ29ycG9yYXRpb24xIzAhBgNVBAsT
GklUUO8gQ2VydGlmaWNhdGUgQXVOaG9yaXR5MIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQCz8tJiJR/7UZnCdiFm+4PnDVFpUnHa4EMtXqfNwBcTcH5tjVj14c+7
5ze/FJ1aZ7aerTU37HBOvRO+eSL2TnZsfVxS5qFS9p64nPgdMXhOc4ixtc8t1mCm
jHJJl319lS7YpZMlK89PUfxLyNWb9xJv7gjiaOMDL9k1SYeDgJescwIDAQABo4GE
MIGBMD8GCWCGSAGG+EIBDQQyEzBHZW5lcmFOZWQgYnkgdGhlIFNlY3VyaXR5IFNl
cnZlciBmb3Igei9PUyAoUkFDRikwDgYDVROPAQH/BAQDAgEGMA8GA1UdEwEB/wQF
MAMBAf8wHQYDVROOBBYEFBqDiVGzNNpeOMTxMVOrD4tHO8a1MAOGCSqGSIb3DQEB
BQUAA4GBAGB3D4WYFizgJkCXSrDw+7lO93zFdwIKzk7IPa9xzL72Ra7tK/Avd5/O
vVzv7NAH4aNXh/u3RBTjpoDjLIyCcygy/1elpauBUJuTTxeNyI1wFTM2t/hqOMlT
CaI1G3KB41ORAPt6XksLYbBQw3uooFsz/7MOnjS7Q/o2IivAzQdR
-----END CERTIFICATE-----
```

Use FTP to send the resulting certificate in ASCII format to a workstation where it is stored (in our case, with the name `cacert.txt`). From this workstation, you can open a web browser and connect to the management port on the DataPower appliance.

You can use the cryptography windows of the DataPower appliance to upload the ASCII file into the key database of the DataPower appliance:

1. In the navigation tree of the DataPower appliance (Figure 10-27), click **Objects** → **Crypto Configuration** → **Crypto Certificate**.

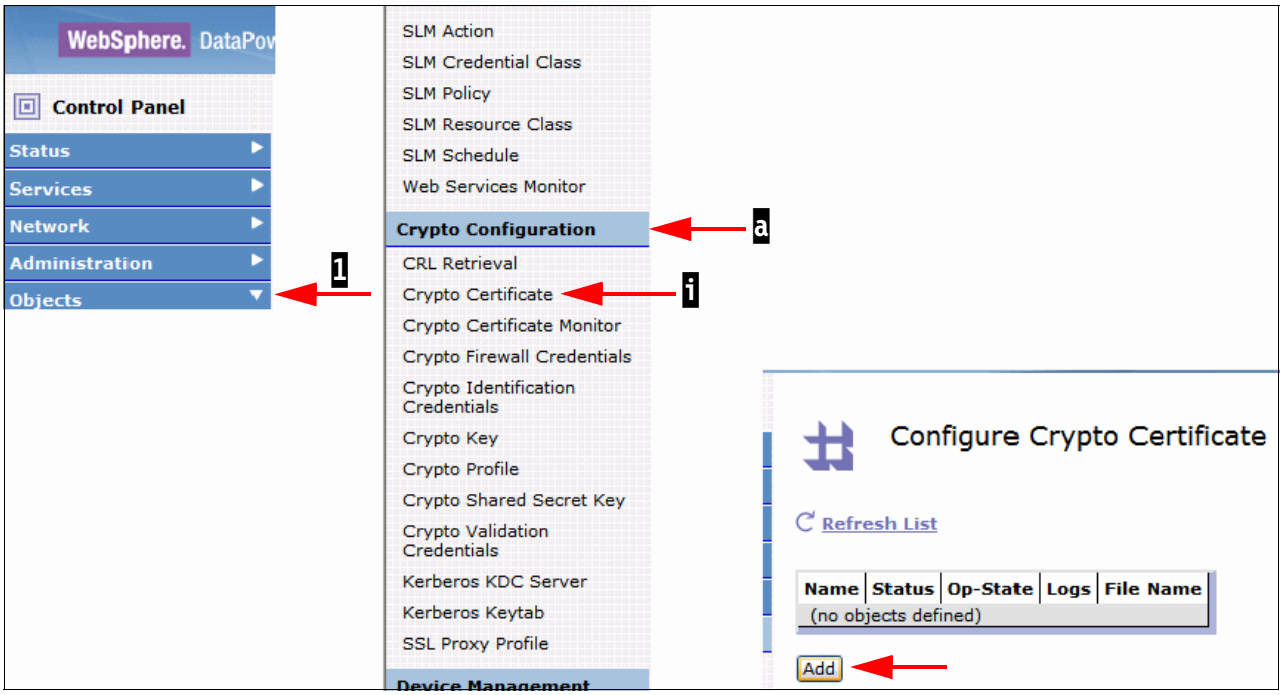


Figure 10-27 Beginning to configure a Crypto Certificate

2. Click **Add** to open the Configure Crypto Certificate window shown in Figure 10-28. Notice the file name in which certificates are stored (the `cert:` directory).

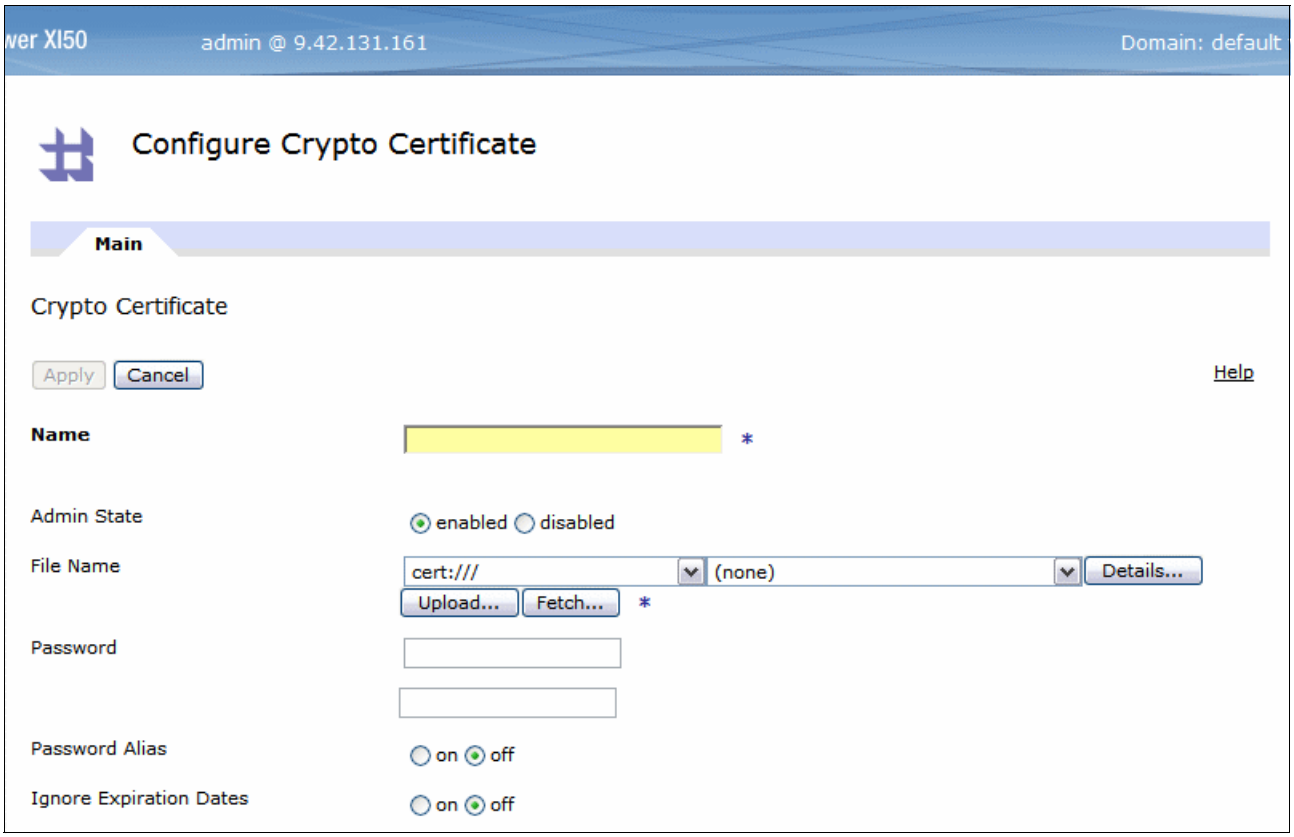


Figure 10-28 Defining the Crypto Certificate object

Tip: Each platform has its own type of repository for storing x.509 certificates. For example, on z/OS, certificates can be stored either in a RACF repository on a key ring or in a UNIX file system in a key database. On the XI50 DataPower appliance, the certificates are stored in a `cert` directory.

3. Select **Upload** and use the two segments shown in Figure 10-29 to browse for the certificate that you stored on your workstation. Enter the name of the certificate stored on the workstation, and provide the name under which you want to save the Certificate Certificate object.

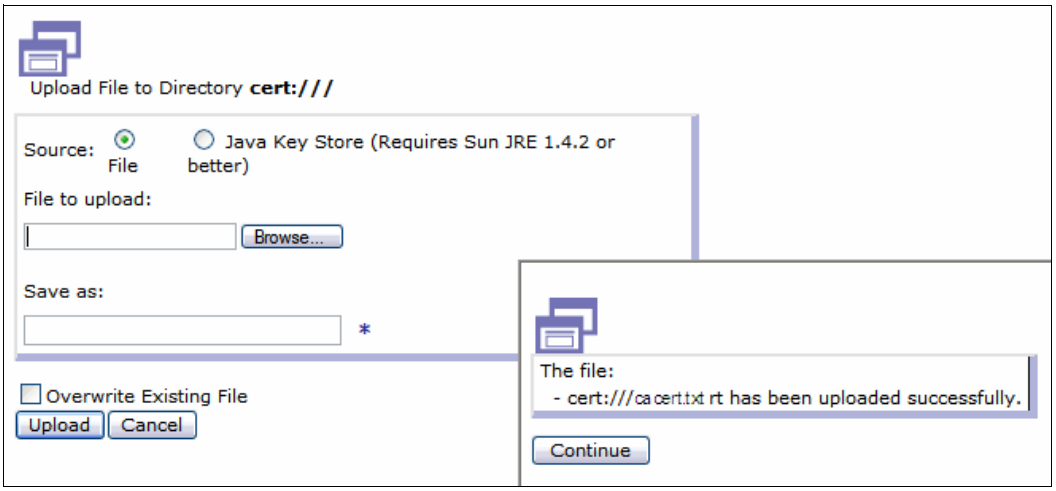


Figure 10-29 Uploading ASCII certificate file from workstation to DataPower appliance

We entered the name `cacert.txt` as the file to upload and saved it as `CA19CERT`.

4. You can select **Upload** to see that the Base64-encoded ASCII certificate is saved successfully in the `cert:` directory. Select **Continue** to open the window shown in Figure 10-30.

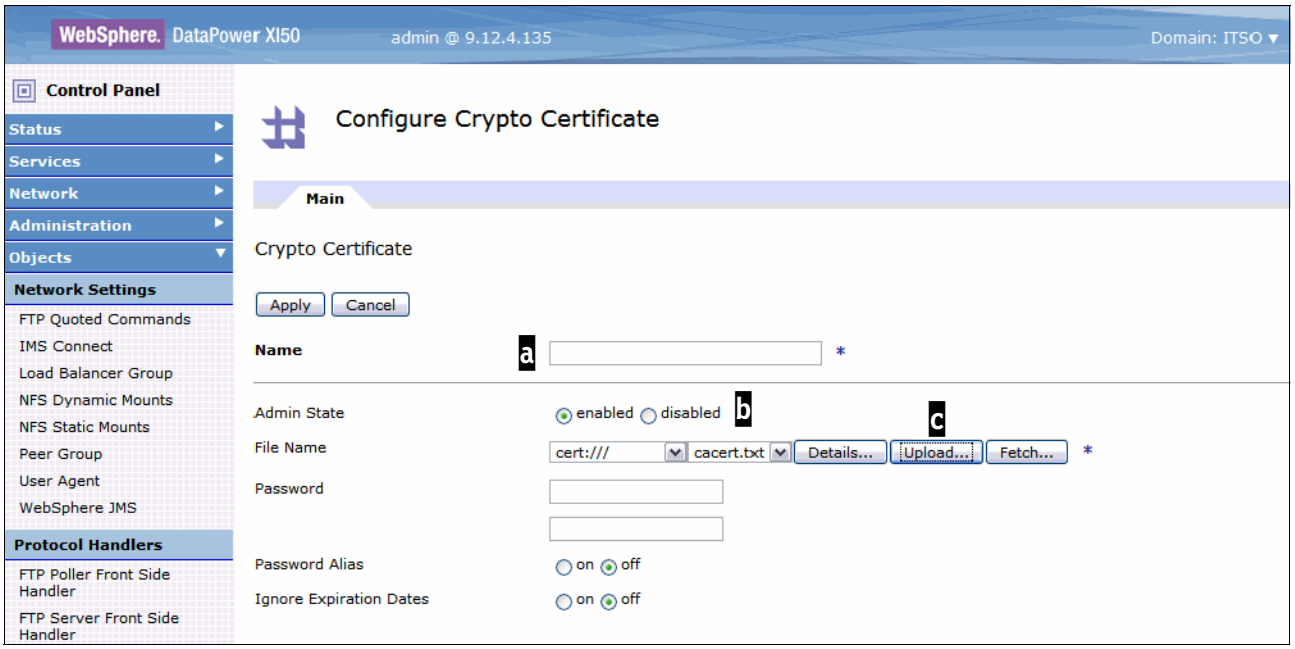


Figure 10-30 Uploading the CA certificate from workstation to DataPower appliance

5. Provide a name for the Crypto Certificate object (a). We entered `CACERT19`. Select **Enabled** for the Admin State (b). The certificate is stored in the `cert:` directory (c). Click **Apply**.

Figure 10-31 confirms that the Crypto Certificate is saved under the name of *CACERT19*.

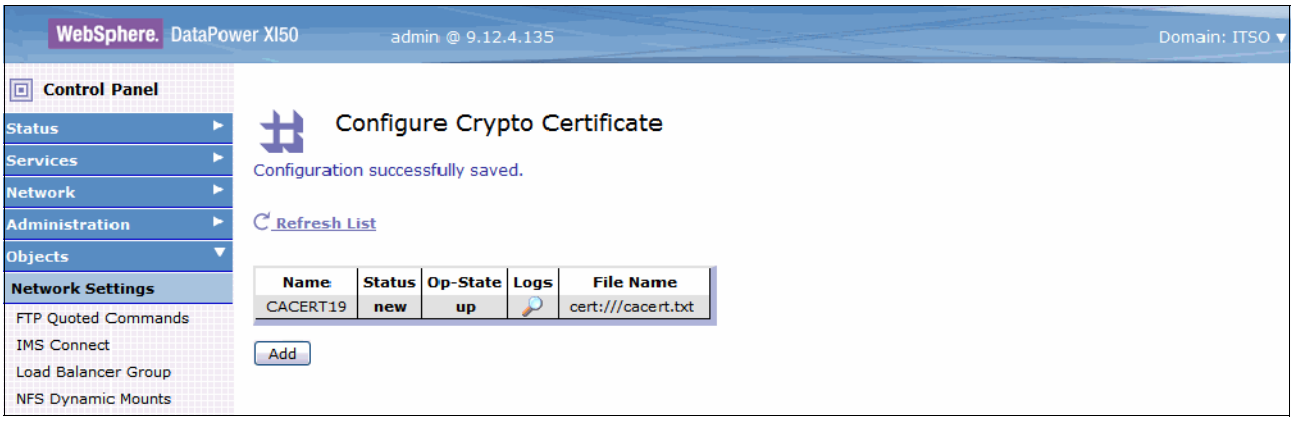


Figure 10-31 Successful upload of CA certificate into the XML cert: directory

6. Save the configuration to ensure that you do not lose your Crypto Certificate object by selecting **Save Config** in the upper, right corner of the window.

The SSL Proxy Profile object

Next, you create an SSL Proxy Profile object for the secured communication between the NSS XMLAppliance Client and the NSS Server. The SSL Proxy Profile object allows you to use the Crypto Certificate object that you just created.

To create the SSL Proxy Profile object, complete the following steps:

1. In the navigation tree of the DataPower appliance (Figure 10-32), expand the Objects view of potential definition objects and click **SSL Proxy Profile**. Then, in the window that opens, click **Add**.

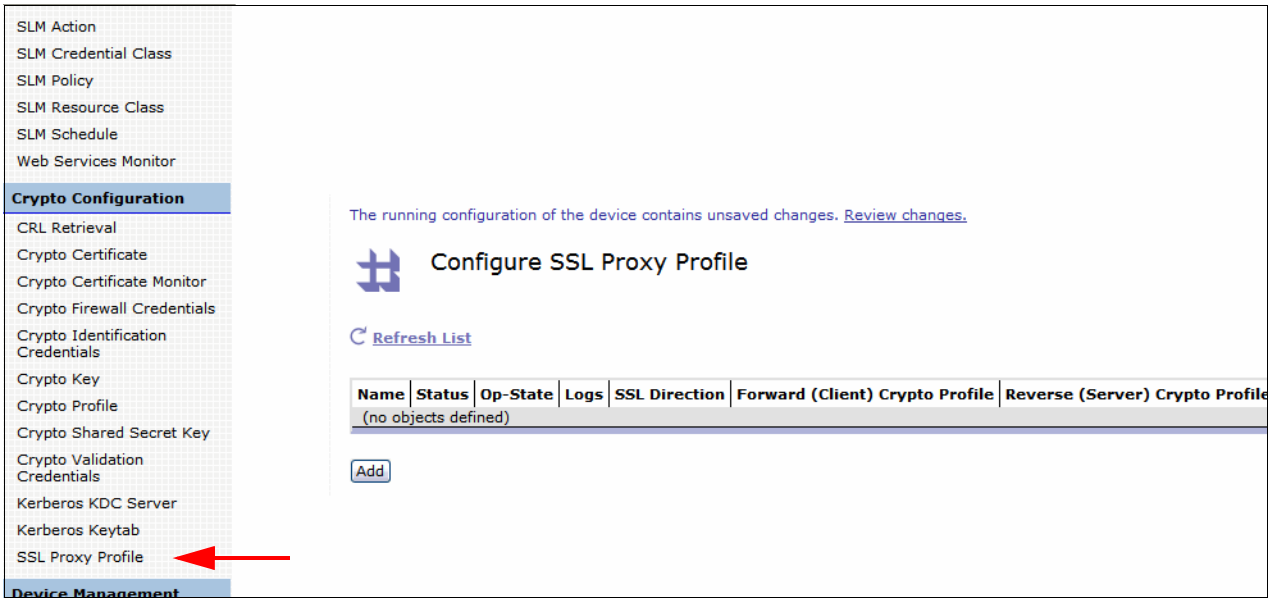


Figure 10-32 Configuring the SSL Proxy Profile object

2. In the Configure SSL Proxy Profile window (shown in Figure 10-33), change the Admin State to **Enabled**. Then change the SSL Direction to **Forward**.

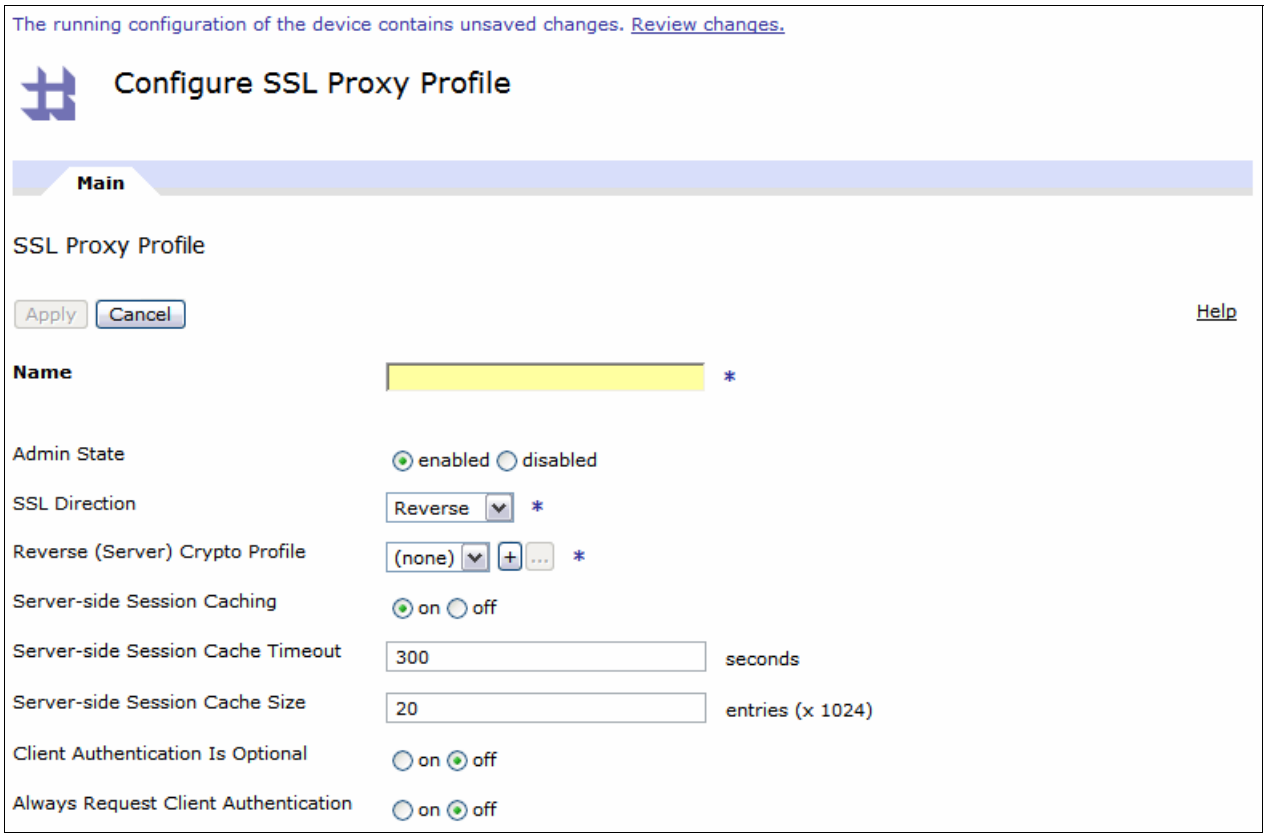


Figure 10-33 Begin to configure the SSL Proxy Profile

These settings reduce the number of configurable fields to those shown in Figure 10-34.

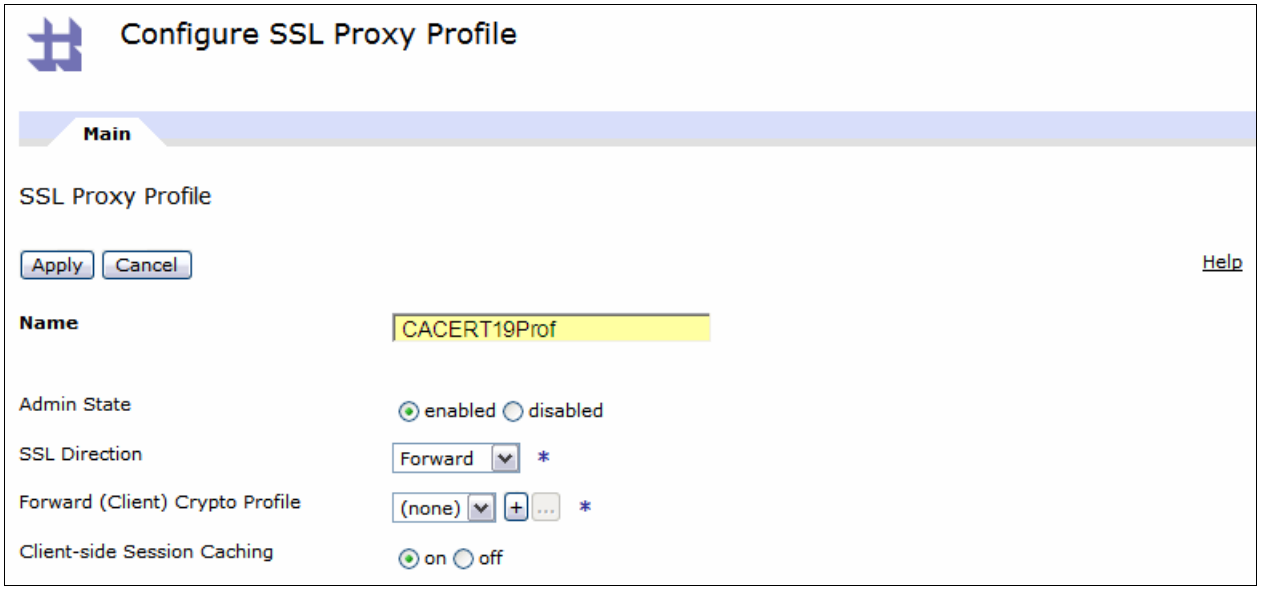


Figure 10-34 Providing SSL Proxy Profile name and definition of Client Crypto Profile

3. Ensure that the Admin State is *Enabled*. Enter the name for the SSL Proxy Profile (in our case, `CACERT19Prof`).
4. To complete the SSL Proxy Profile, you need to create:
   - A crypto profile object
   - A set of validation credentials

   On the line labeled “Forward (Client) Crypto Profile,” click the plus sign (+) to create a Client Crypto Profile using the information shown in Figure 10-35.

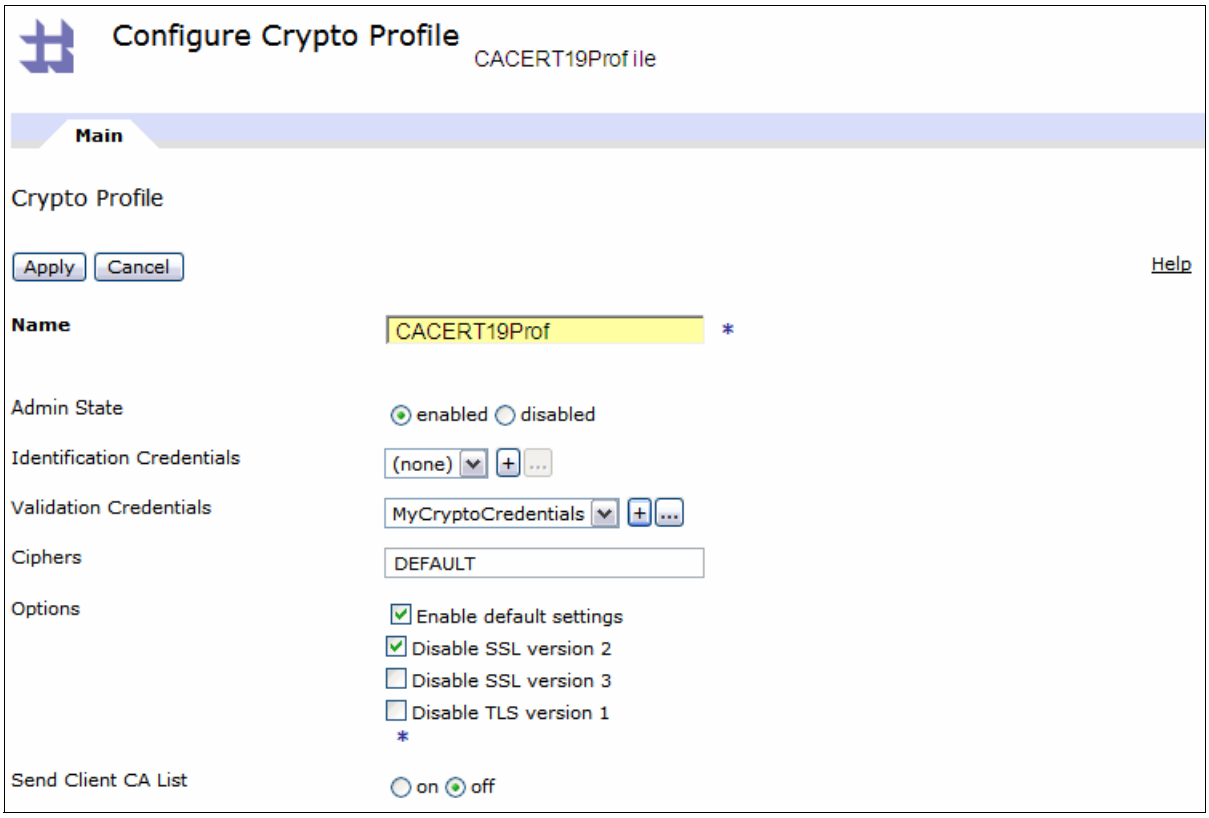


Figure 10-35 Configuring the Crypto Profile

We assigned the name `CACERT19Profile` to the Crypto Profile, and **Enabled** the object.

5. The Crypto Profile references the following security objects:
   - Identification Credentials
   - Validation Credentials
   - Ciphers
   - Options

   Select the security objects as shown in Figure 10-35.

6. Because we did not require client certificates in our SSL/TLS negotiations, we skipped the definition of the Identification Credentials. However, we did need to validate incoming credentials. Therefore, we created an object for Validation Credentials. Click the plus sign for this object to open the Configure Crypto Validation Credentials window shown in Figure 10-36.

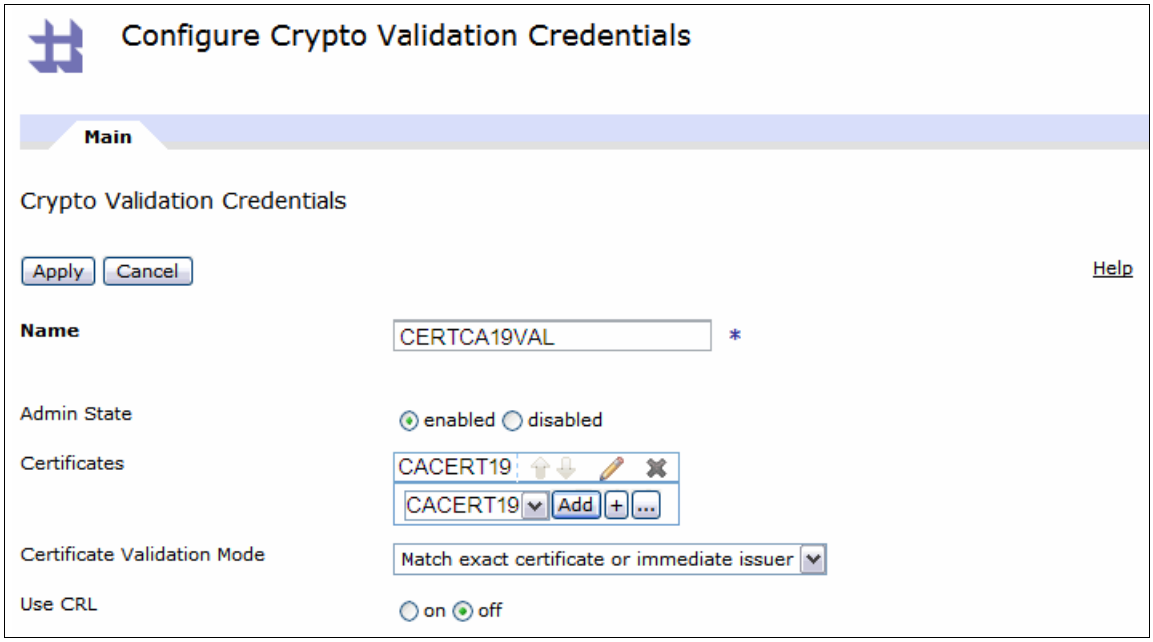


Figure 10-36 Creating the Crypto Validation Credentials

7. Enter the name of the Validation Credential that you are defining (in our case, `CERTCA19VAL`). For the Admin State, select **Enable**. Select the Validation Credential from the menu and click **Add**. Finally, select **off** for the Use CRL option. Remember that CACERT19 is the CA certificate that is used to authenticate the NSSD server certificate. Click **Apply**.

8. In the Configure Crypto Profile window shown in Figure 10-37, click **Apply**.

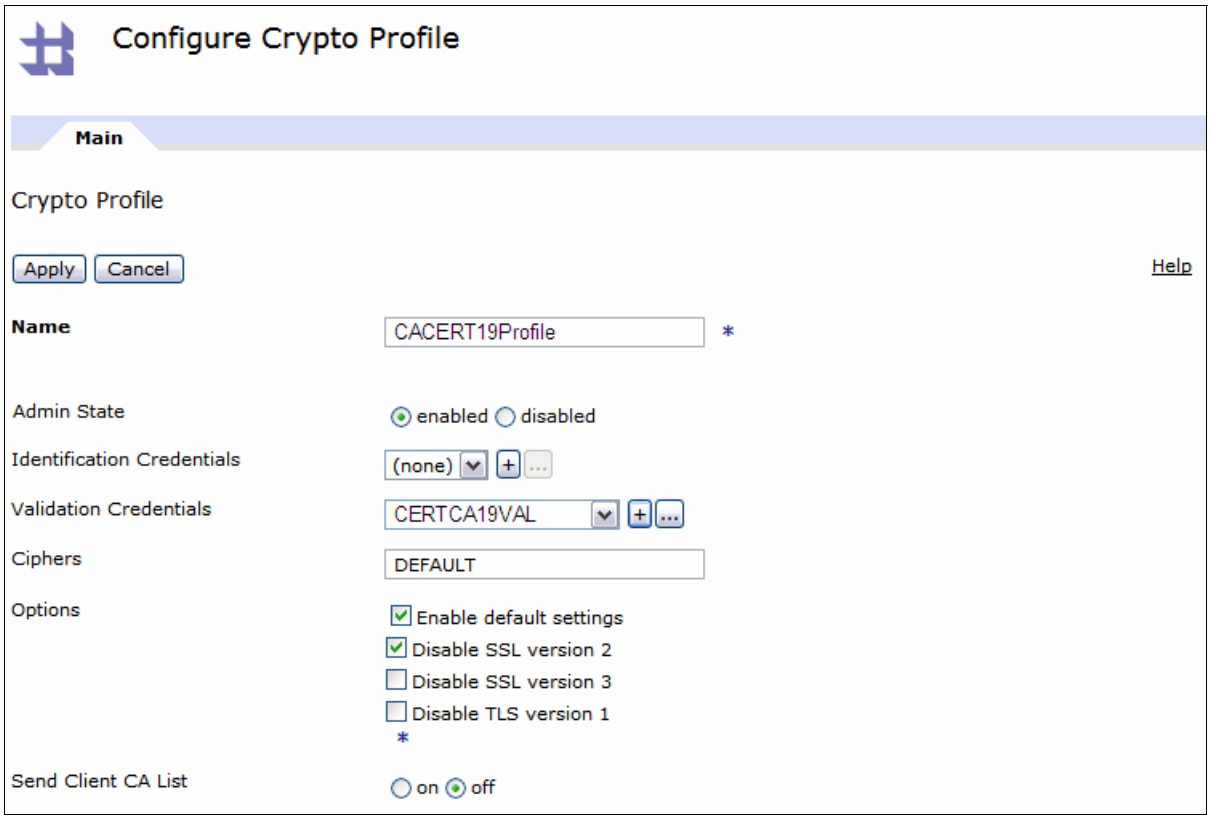


Figure 10-37 Completing the Crypto Profile

9. Click **Apply** to return to the Configure SSL Proxy Profile window shown in Figure 10-38.

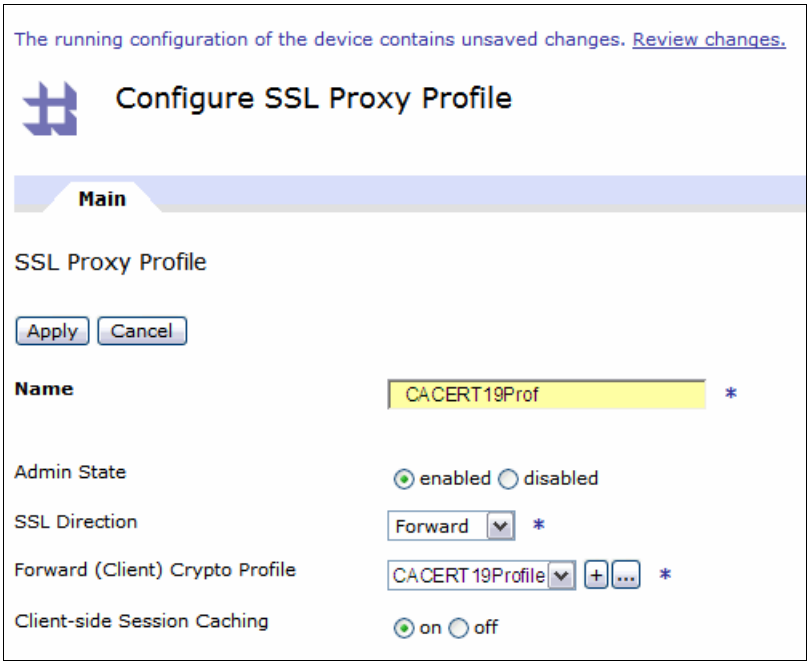


Figure 10-38 Nearly completed SSL Proxy Profile

10. Click **Apply** to return to the completed SSL Proxy Profile object shown in Figure 10-39.

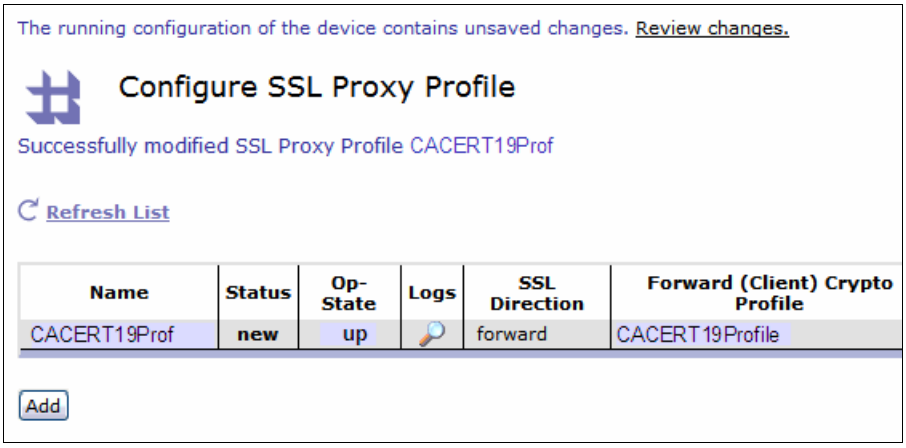


Figure 10-39 Completed SSL Proxy Profile

11. The SSL Proxy Profile in Figure 10-39 points to a Crypto Profile, which points to the Validation Credentials, which point to the Certificate object. Save the configuration using the option in the upper right corner.

Tip: Our test scenario does not exploit a connection to a back-end server. Therefore, we require no additional SSL Proxy Profiles that can be used for an SSL/TLS connection between the Web Services Requester and the back-end server application.

The NSS client configuration (XML Appliance Discipline)

Now, you configure the NSS client object on the DataPower appliance:

1. Begin by expanding the Objects view in the DataPower appliance and selecting the z/OS NSS Client object from the expanded window. Then click **Add** as shown in Figure 10-40.

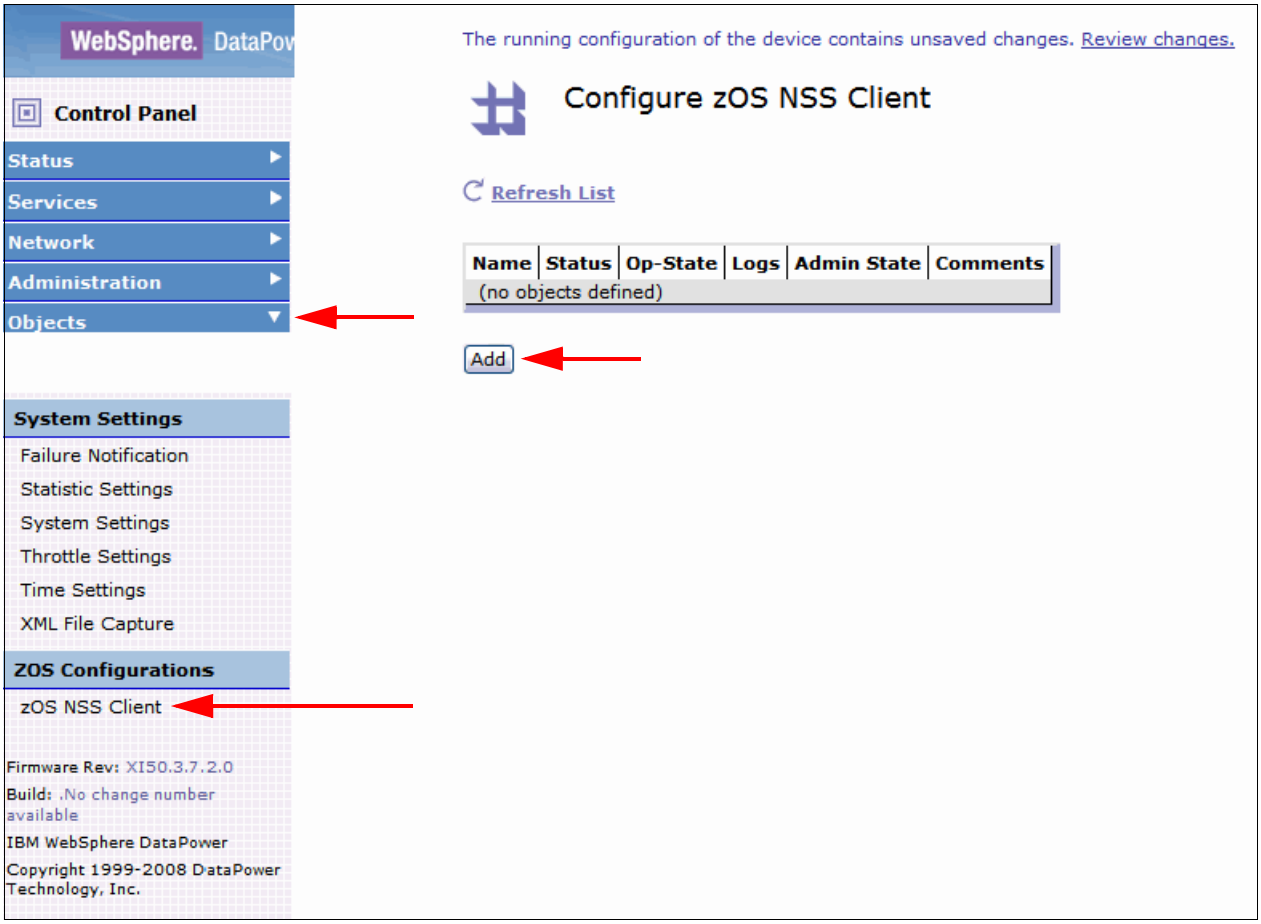


Figure 10-40 Configuring the NSS Client in the XI50

2. In the Configure z/OS NSS Client window, shown in Figure 10-41 on page 495, enter the name of the z/OS NSS Client object. In our testing, we gave the object the same name as the NSS Client ID, but the two names do not have to be identical. Select **Enabled** for the Admin State. Enter the remote address (`10.1.2.45`) and listening port (`4159`) of the NSS server. Identify the NSS Client ID (`NSSXML1`), the system name on which the NSS server is running (`SC33`), and the RACF user ID and password (`NSSXML` and `XMLPASS`) that represent this Client ID.

   Tip: The NSS Client ID must be a unique name within the NSS server. The user name can be shared, although we chose to make it a unique RACF user ID in our scenario.

   Use the pull-menu down for SSL to identify the SSL Client Proxy Profile name that you defined earlier in this scenario (in our case, `CACERT19Prof`).

3. Click **Apply** to confirm these values.

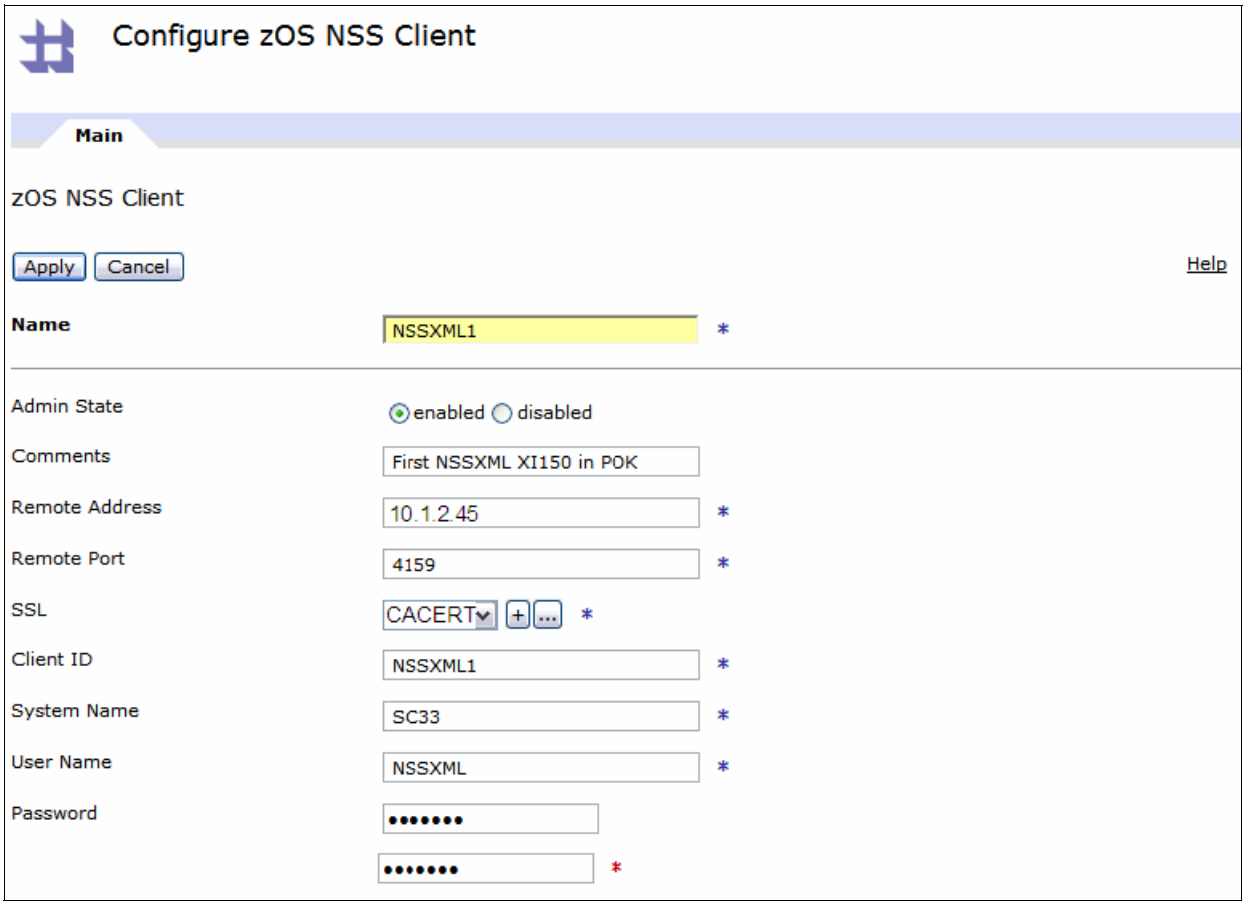


Figure 10-41 Configuring the z/OS NSS Client object

4. You receive a message that the configuration was modified successfully as shown in Figure 10-42. Click **Save Config** to save your changes to the configuration.

Figure 10-42 Completed NSS Client object

The XML firewall configuration

Next, configure the XML firewall service configuration.

When the NSS client requires security services from the NSS server, two types of security calls are made:

- The **zosnn-authen** calls, which authenticate the NSS Client's user ID with the security services at z/OS
- The **zosnss-author** calls, which authorize the NSS Client's user ID to access XMLAppliance services

The DataPower appliance can employ either of the following methodologies to issue these security calls to the NSS server:

- Direct Method

  The DataPower appliance can make the calls for NSS services directly by using either hard-coded values or variables in the authorization and authentication calls. The variables are replaced with values that are passed into the calls by the Web Services Requester or the HTTP server for DataPower.

- Indirect Method

  The DataPower appliance can invoke an AAA action from a service policy rule to issue authentication and authorization calls.

We chose to test with the Indirect Method with an AAA policy invoked through the XML firewall.

Figure 10-43 shows an enhanced view of our test scenario.

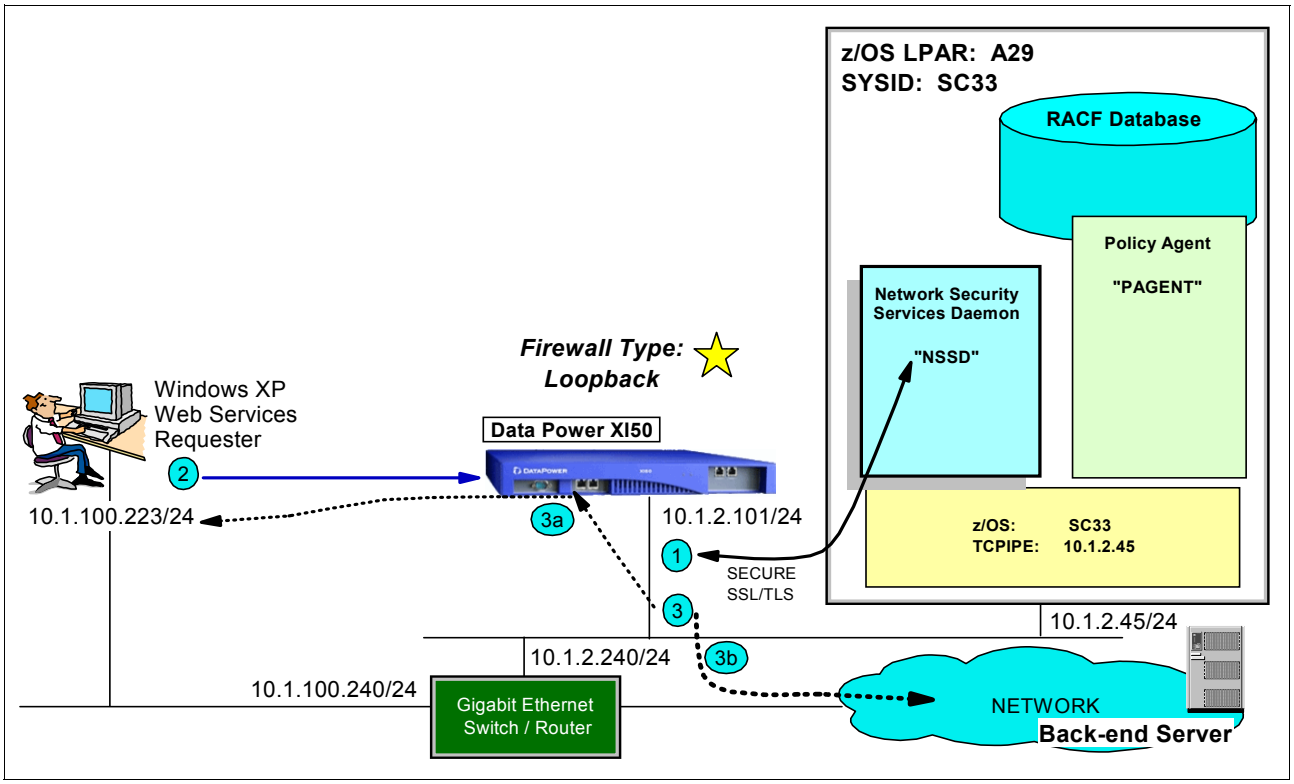


Figure 10-43 Test of DataPower XI50 as NSS Client to NSS on z/OS SC33

Notice the dotted line at 3a. In our test scenario, we did not use a back-end server to interconnect the Web Services Requester with an application in the network. Instead, our Web Services Requester requires the invocation of a service within the DataPower appliance itself. The XML firewall service operates here as a *loopback firewall*, with the service invoked within the DataPower appliance itself. Note how the response to the Web Services Request (3a) is sent by the loopback firewall and not by a back-end server in the network.

The XML firewall service operates against the NSS Client object that you defined when a service that requires the NSS Client arrives at the DataPower appliance. Services are composed of many components, as shown in Figure 10-44.

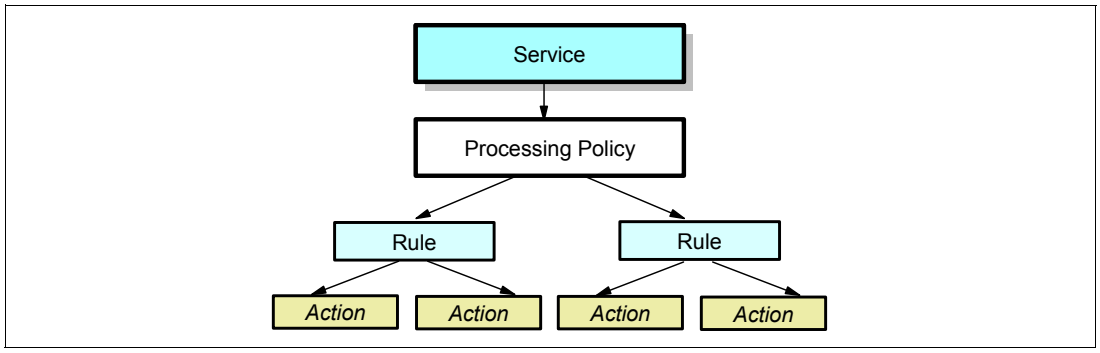


Figure 10-44 Relationship of services, processing policies, rules, and actions

A *service* is associated with only one processing policy. Each *processing policy* in the DataPower appliance can be mapped to one or more rules. Each *rule*, in turn, can cause one or more *actions* to be invoked, depending on how the incoming Web Services Request maps to the rule conditions.

In the case of the NSS XMLAppliance client in our scenario, the service that we invoked is an XML firewall service that we named `NSS_XML_Firewall`. This service (1) points to many components as shown in Figure 10-45.

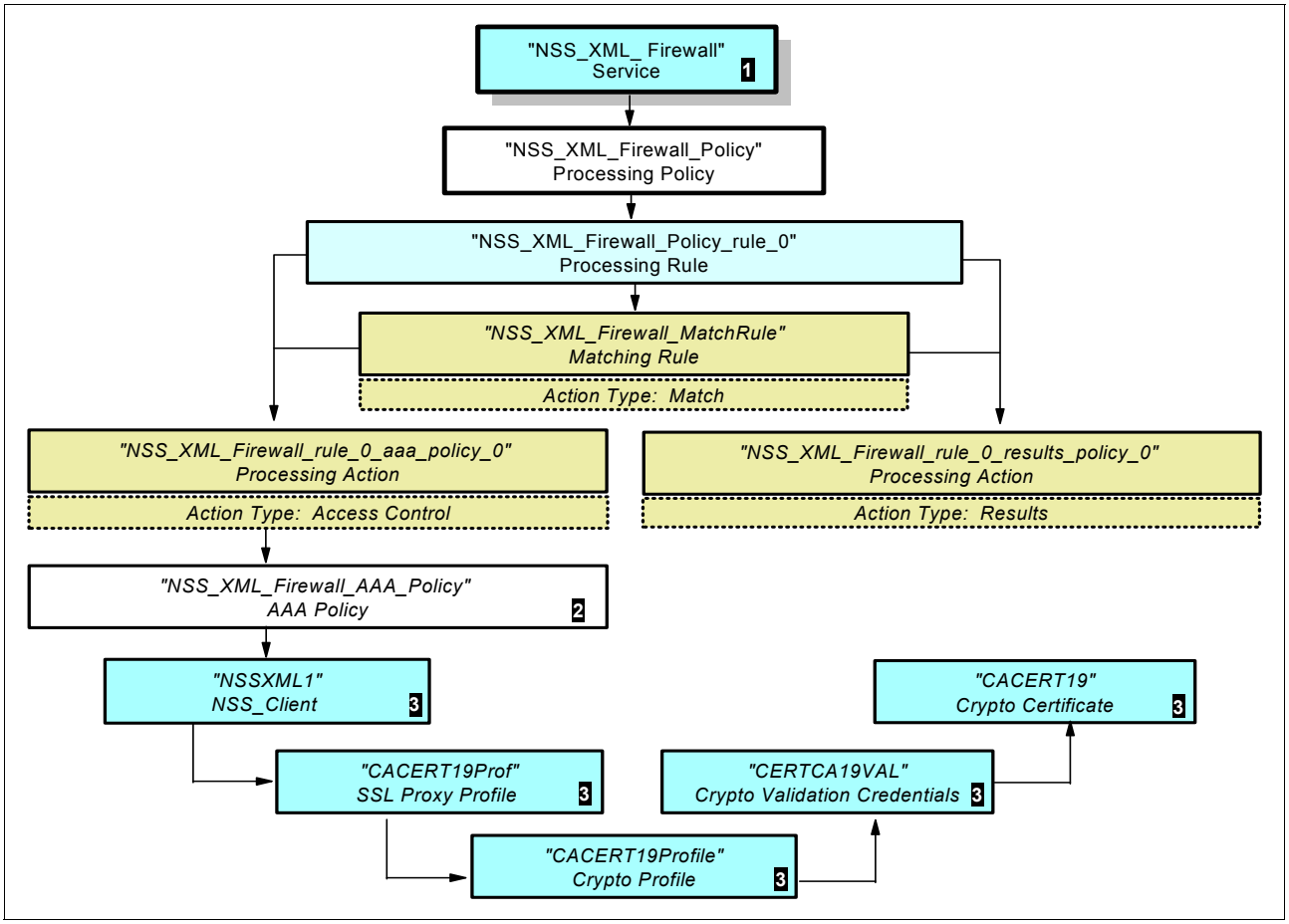


Figure 10-45 Specifications for the NSS configuration

Note in Figure 10-45 that we have already completed the NSSXML1 client component together with its cryptographic profiles (3). We next configure the firewall service (1), its rules, actions, and its AAA policy (2), which points to the NSS client, which is already configured.

To configure the firewall service, follow these steps:

1. From the navigation menu on the DataPower appliance, expand **Services** and select **New Advanced Firewall** as shown in Figure 10-46.

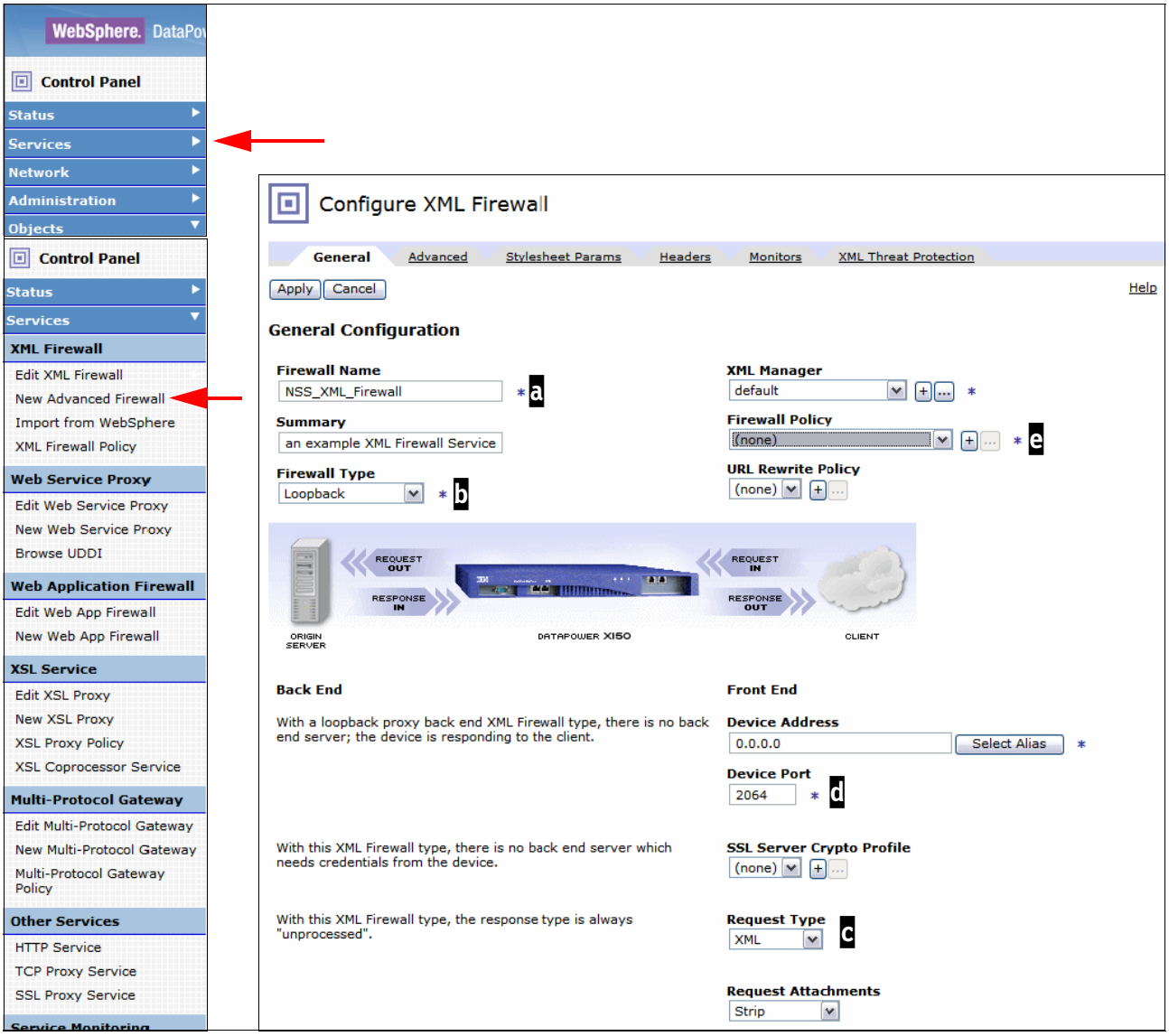


Figure 10-46 Configuring the XML firewall service

2. In the Configure XML Firewall window, enter the name of the Firewall Service at **a** (in our case, `NSS_XML_Firewall`). Then, at **b** select a Firewall Type of **Loopback** from the menu. For our scenario, we do not require a back-end server. Therefore, we allow the DataPower appliance to respond to the Web Services Requester client depicted in the figure. Because there is no back-end server, we require no SSL Server Crypto Profile for the Web Services Requester client. The only SSL activity that we performed in our test scenario is that between the NSS Client and the NSS Server.
3. Select a Request Type of **XML** at **c**. We retained the Device Port number at **d** of 2064, which is the listening port for the firewall that we defined.
4. Click the plus sign (+) in the Firewall Policy field at **e** to add a Firewall policy.

5. The window shown in Figure 10-47 opens. In this window, enter the name of the firewall policy at a (in our case `NSS_XML_Firewall_Policy`). Then, click **New Rule** at b to automatically assign a rule name based on the firewall policy name (in our case, `NSS_XML_Policy_rule_0`). At c, select **Both Directions** to indicate that the firewall policy rule is bidirectional and not different in each direction.

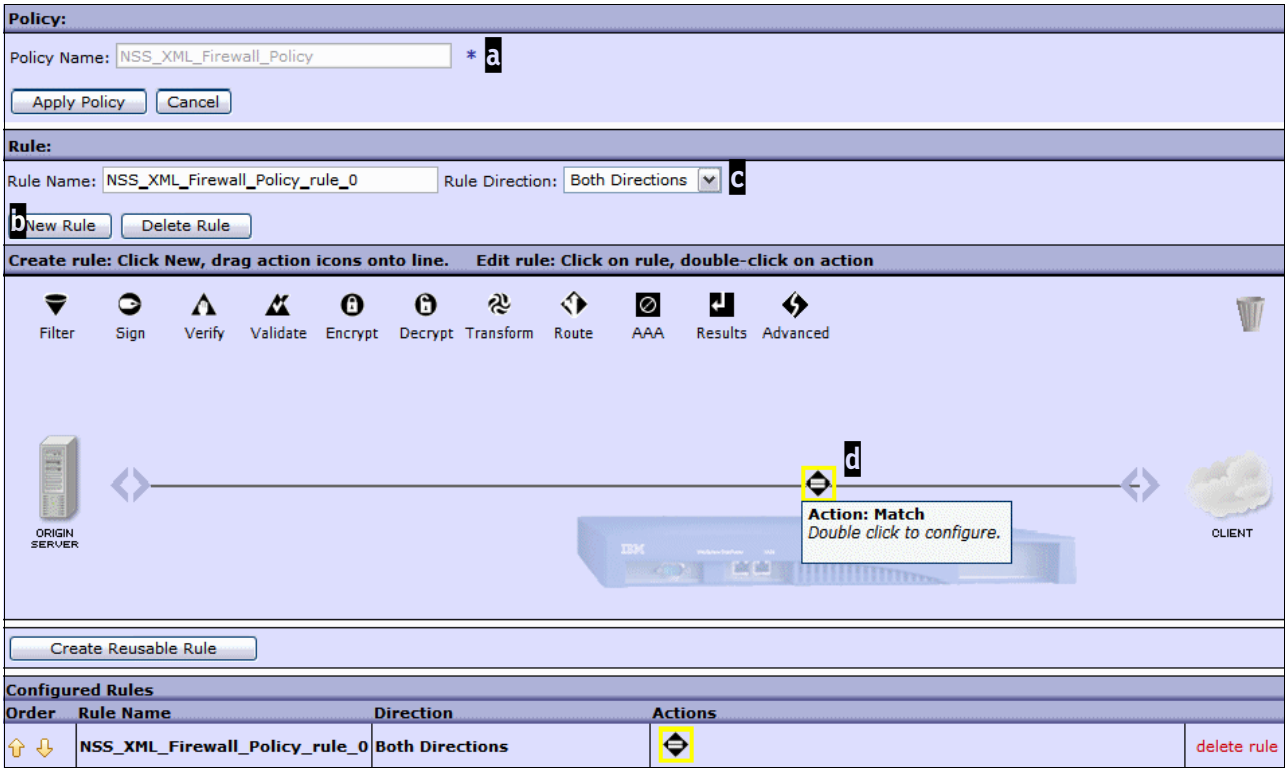


Figure 10-47 Building a Firewall Policy

At this point a diamond-shaped icon displays on the rule line (d) to indicate that you need to define a *Match Action* and *Match Rule*. Hover the mouse pointer over the icon to determine the meaning of the icon and how to proceed. In our scenario, we double-clicked the icon and opened the window shown in Figure 10-48. Click the plus sign (+) to configure the Matching Rule.

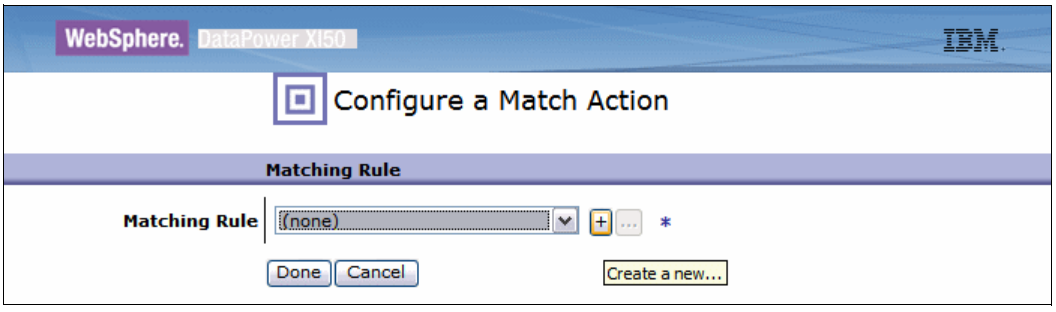


Figure 10-48 Configuring the Match Action

6. In the Configure Matching Rule window, shown in Figure 10-49, on the Main tab, enter the name of the firewall policy matching rule (in our case, `NSS_XML_Firewall_MatchRule`) at a. Click **Enable** for the Admin State at b. Then, at c, click the Matching Rule tab.

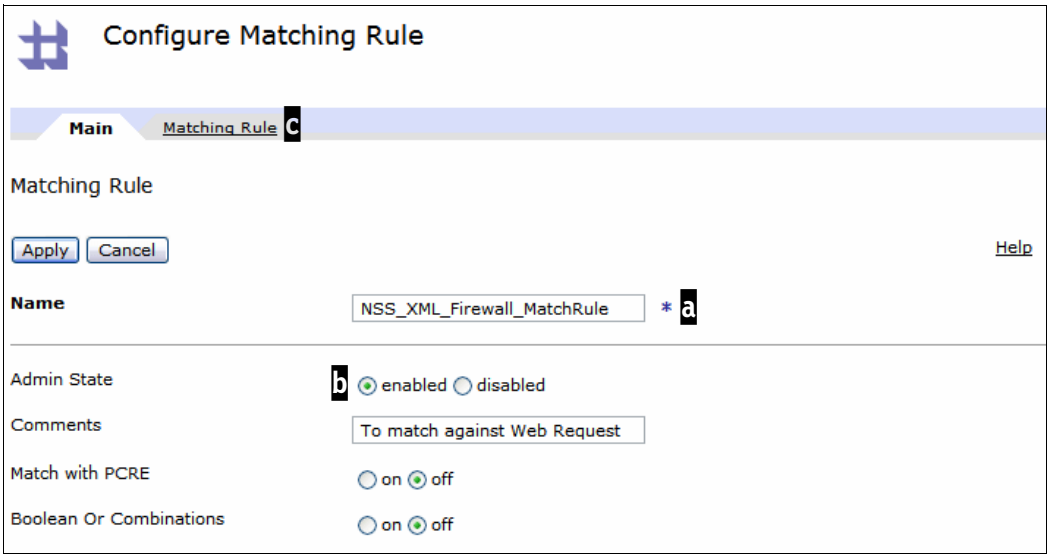


Figure 10-49 Configuring the Matching Rule (Part 1)

7. On the Matching Rule tab, click **Add** (at a).

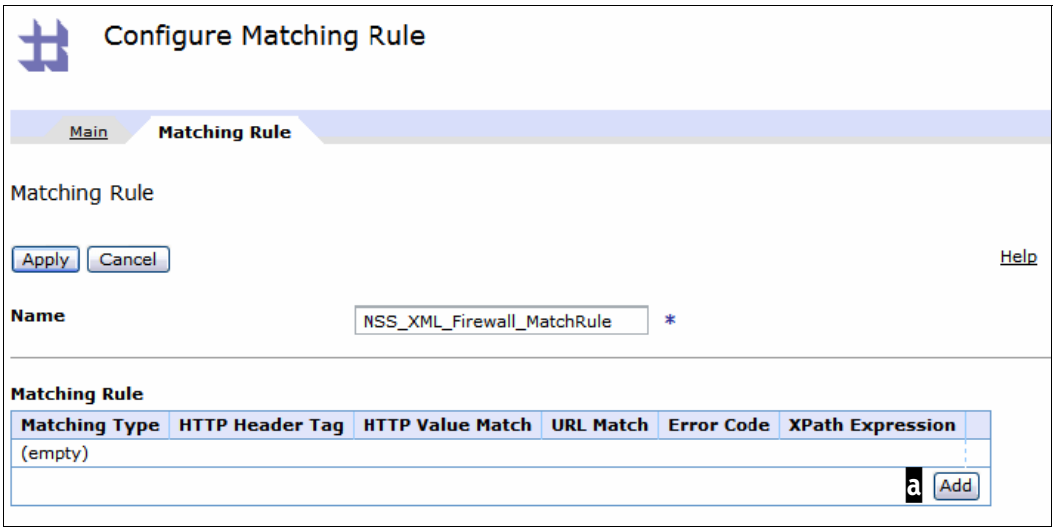


Figure 10-50 Configuring the Matching Rule (Part 2)

8. Complete the Matching Rule conditions as shown in Figure 10-51. In our testing, we selected a Matching Type of **URL** because the `curl` command submitted by the Web Services Requester uses the URL address to direct the request to the DataPower appliance. Allow any valid URL address by typing an asterisk (*) in the URL Match field. Click **Apply**.

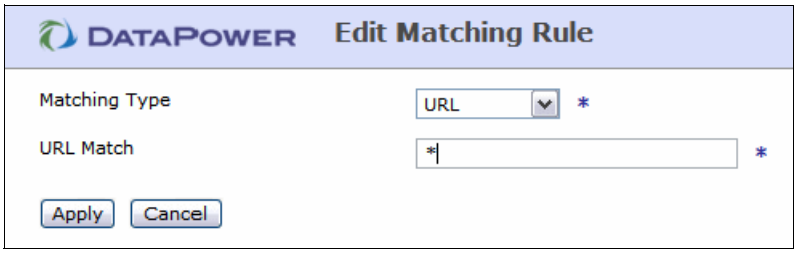


Figure 10-51 Specifying Conditions for the Matching Rule

9. Back in the Configure Matching Rule window, which is complete, click **Apply**.

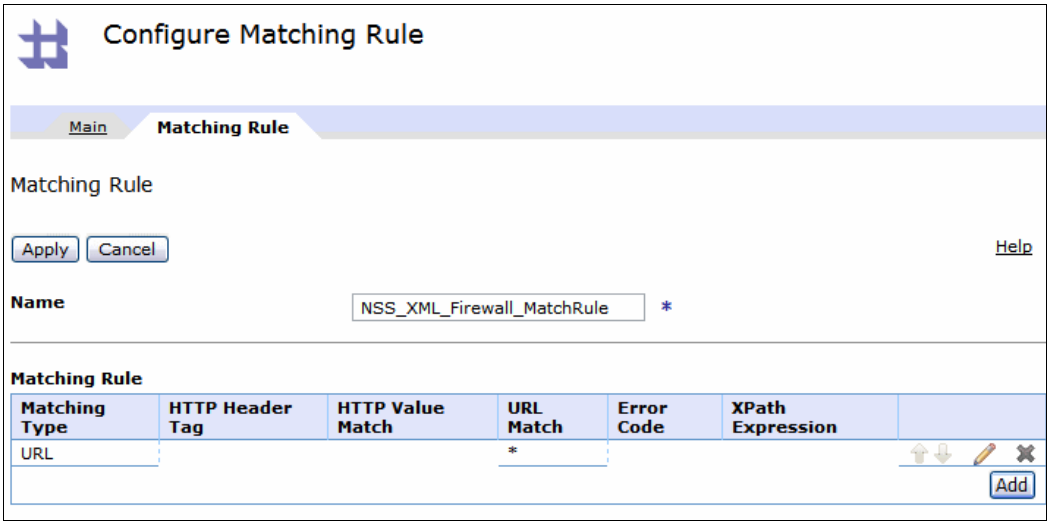


Figure 10-52 Conditions completed for the Matching Rule

10. Then, back in the Configure a Match Action window shown in Figure 10-53, click **Done**.

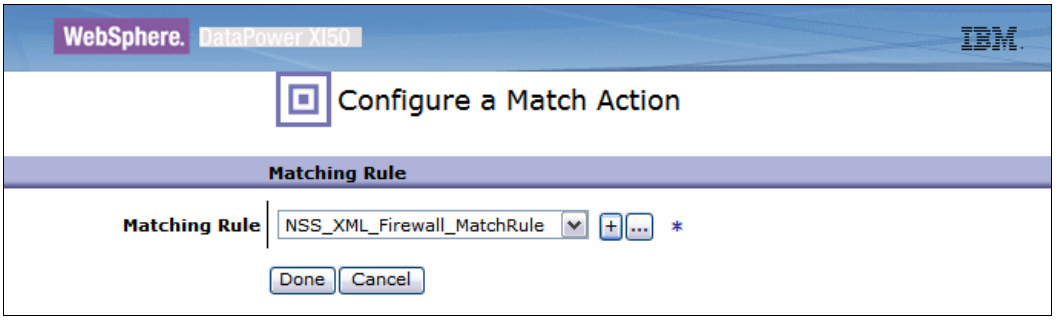


Figure 10-53 Completed Match Action

11. Next, you need to add an AAA action to the Firewall Policy Rule. Select the highlighted icon in Figure 10-54 and drag it onto the Rule line, resulting in the window shown in Figure 10-55.

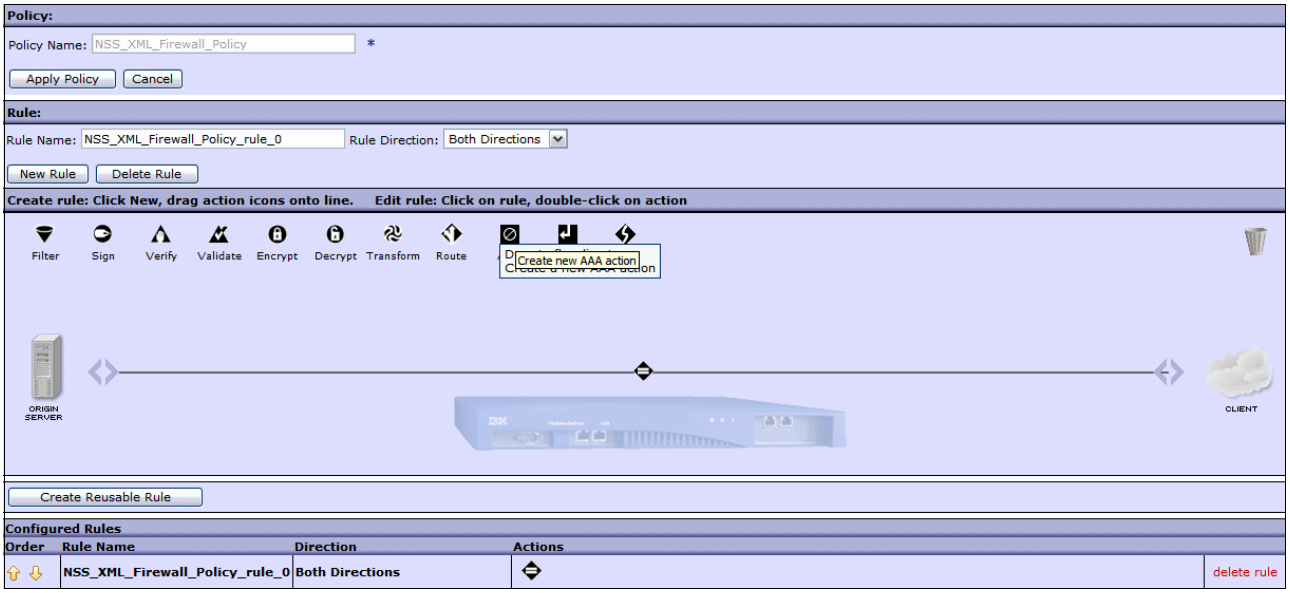


Figure 10-54 Completed Match Action/Rule for the NSS Firewall Policy Rule

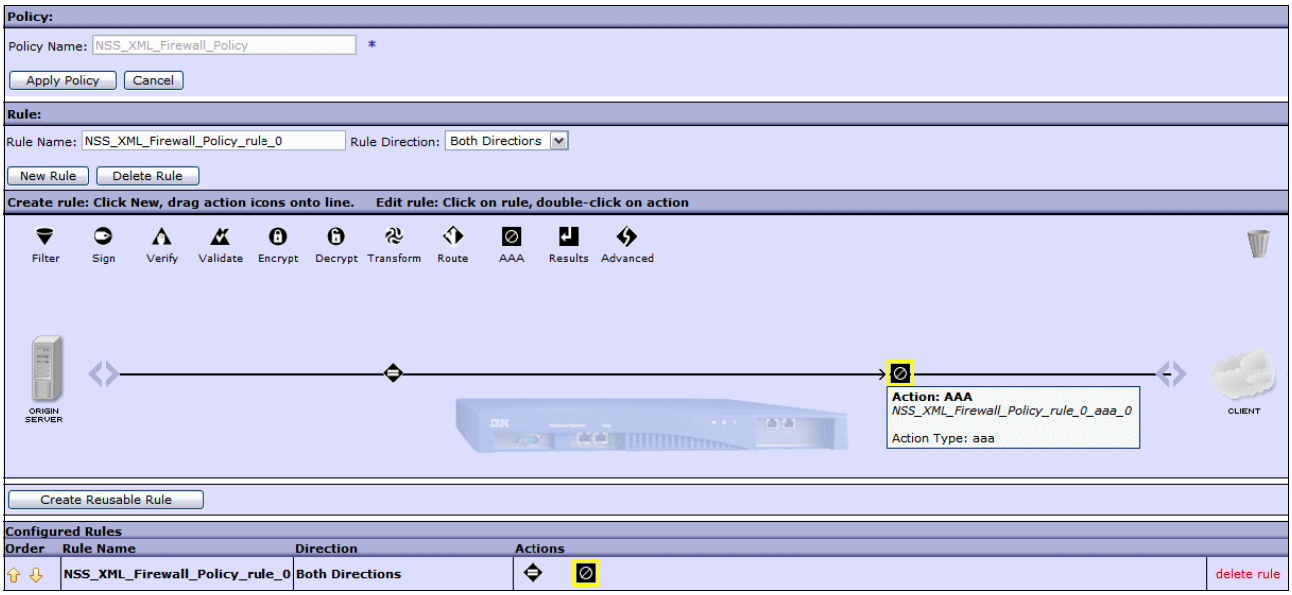


Figure 10-55 Configuring an AAA action and rule

12. Double-click the AAA action icon to specify the conditions and actions for the AAA policy. Use the menus shown in Figure 10-56 to specify INPUT and OUTPUT. Then click the plus sign (+) to define the AAA Policy object.

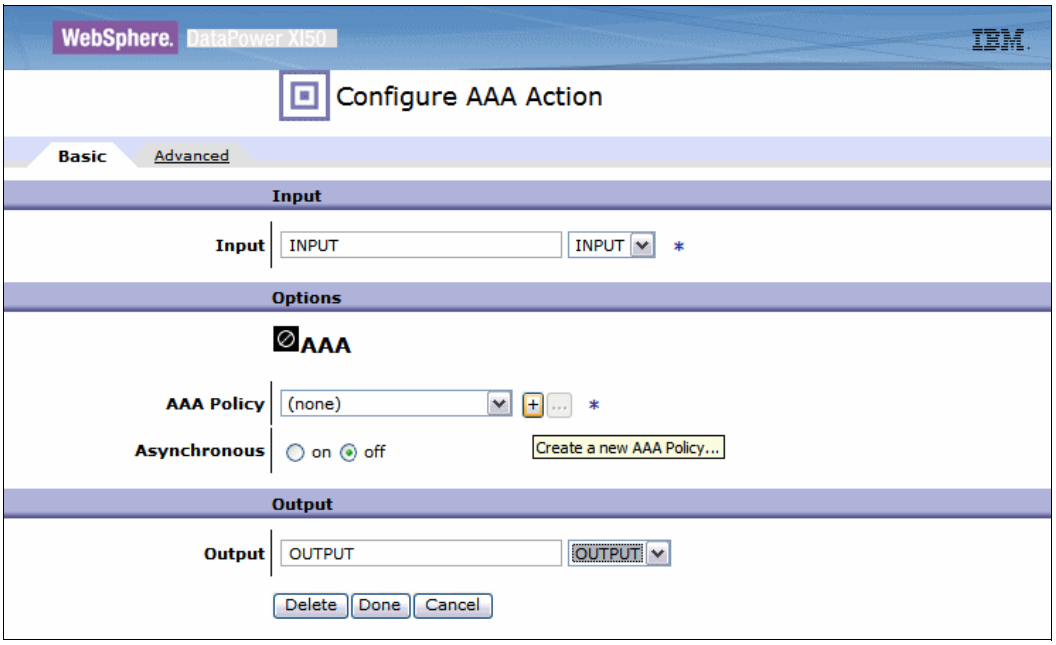


Figure 10-56 Configuring the AAA action, rule, policy

13. Enter a name for the AAA policy (in our case, `NSS_XML_Firewall_AAA_Policy`) and click **Create** as shown in Figure 10-57.

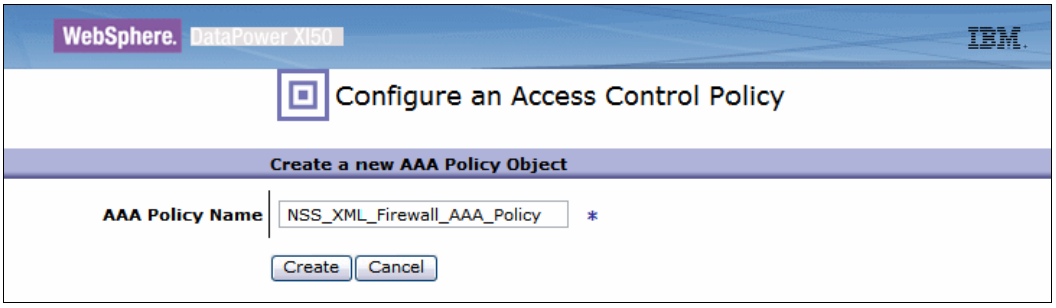


Figure 10-57 Creating the Access Control Policy object for an AAA policy

14. In Figure 10-58, select **HTTP Authentication Header** to extract a user's identity from an incoming request from the Web Services Requester user ID and password (in our case, CS03 and CS03) from the HTTP header that is presented (again, in our scenario) by the `curl` command content. Click **Next**.

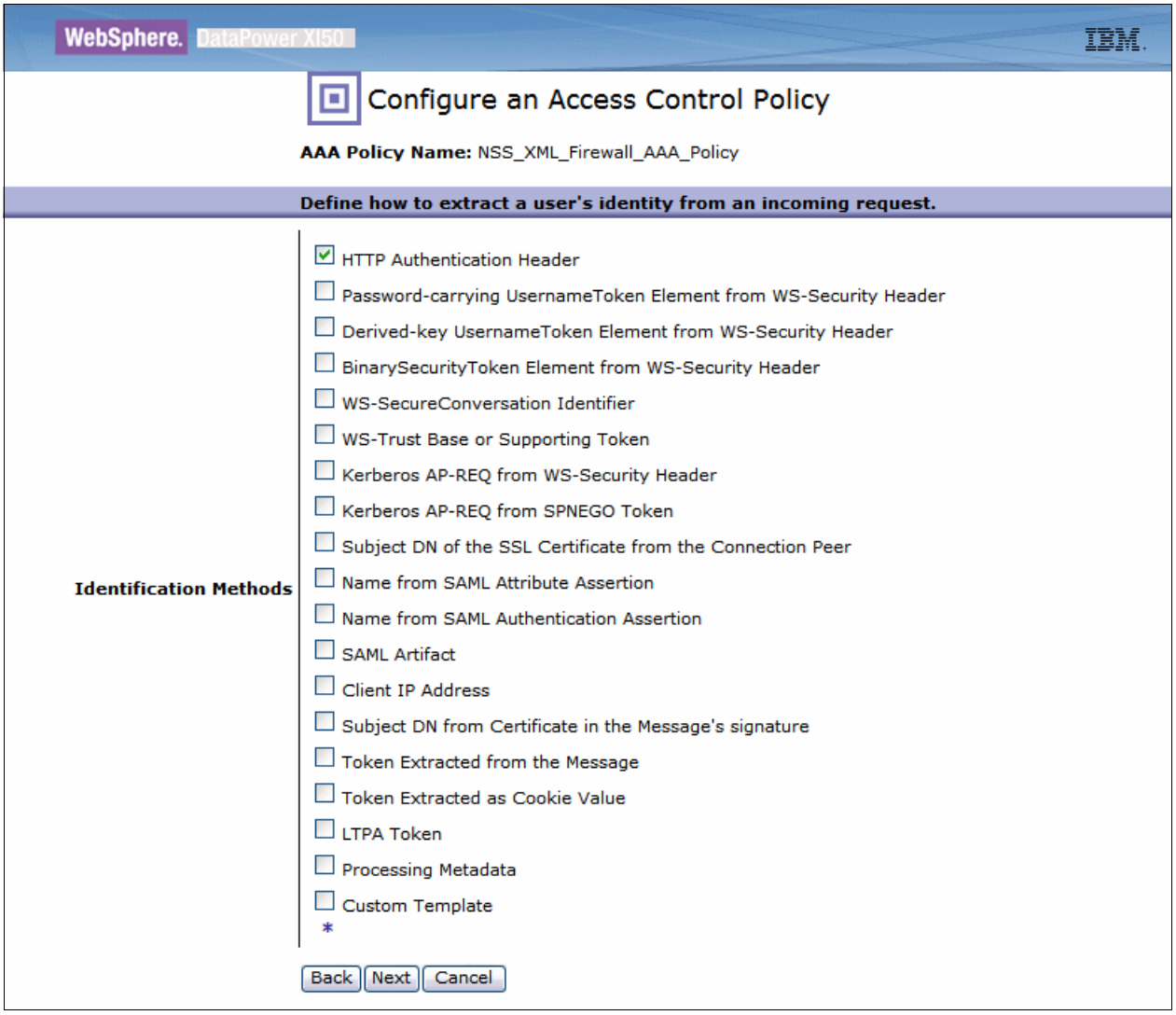


Figure 10-58 Identification methods for an Access Control Policy

15. Next, you need to define how you will perform the authentication (Figure 10-59). In our testing, we intended to integrate the DataPower appliance with System z and the NSS Server. The user ID and password that is included in the HTTP request contained in the **curl** command is presented to the NSS Server for SAF Authentication by the NSS Client. Therefore, in Figure 10-59, we selected **Contact NSS for SAF Authentication**. Then we selected the SAF Client Configuration named NSSXML1 and clicked **Next**.

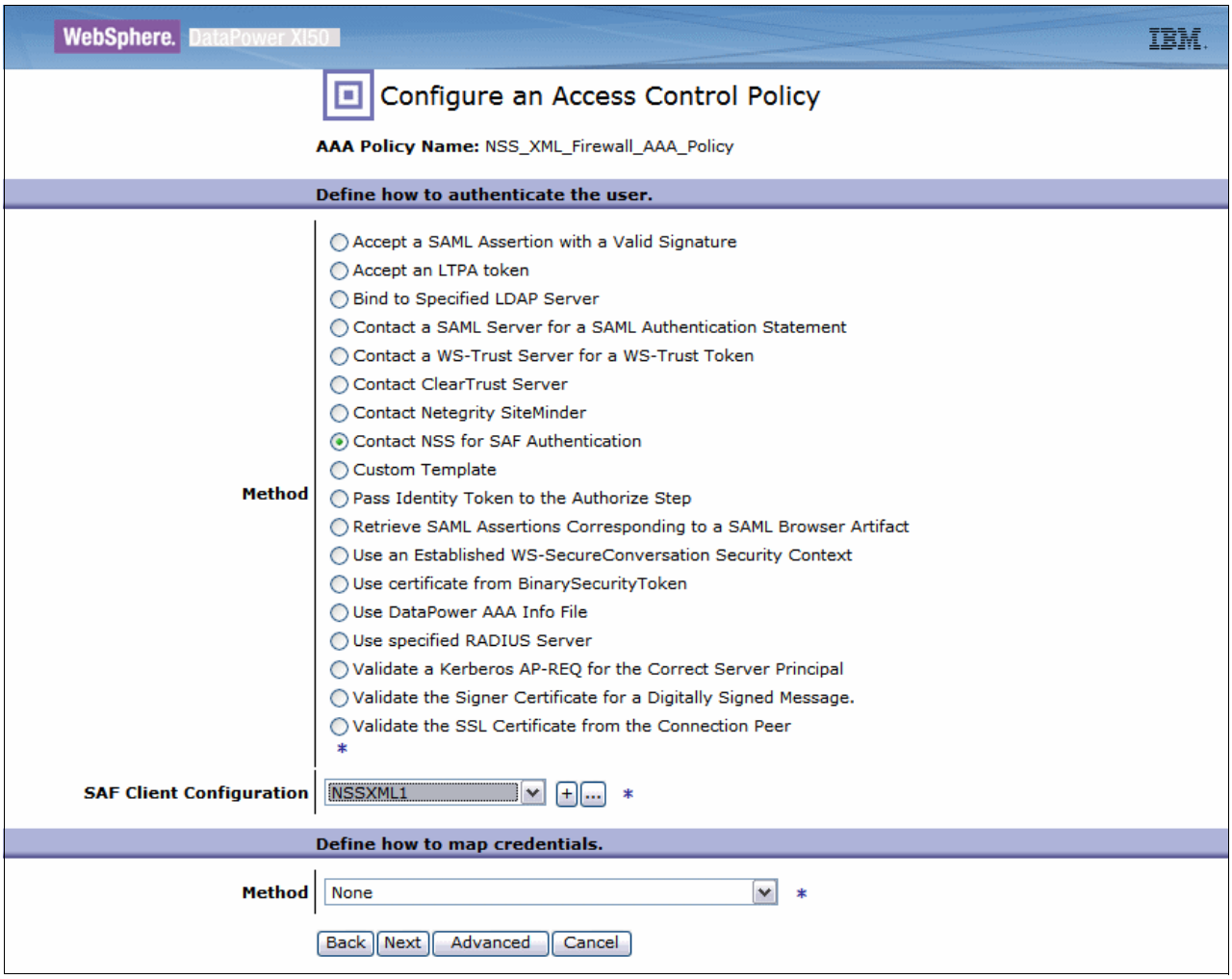


Figure 10-59 The authentication method to be used against the user ID and password

16. You need to create an HTTP header at the Web Services Requester and include in its contents a value that can be compared with the AAA Policy that is defined in the DataPower appliance. If the value matches, then the AAA Policy triggers an XML request to the NSS server.

    In Figure 10-60, identify a matching value that can trigger an authorization request. In our testing, we selected **XPath Expression** from the Resource Identification Methods. Enter the wording for which the DataPower appliance needs to scan in the XPath Expression text box (in our case, `/AuthroizationResource`). Then, when the DataPower appliance finds the expression, it can identify the string containing the SERVAUTH resource name for which the authenticated user ID requires authorization.

    Click **Next**.

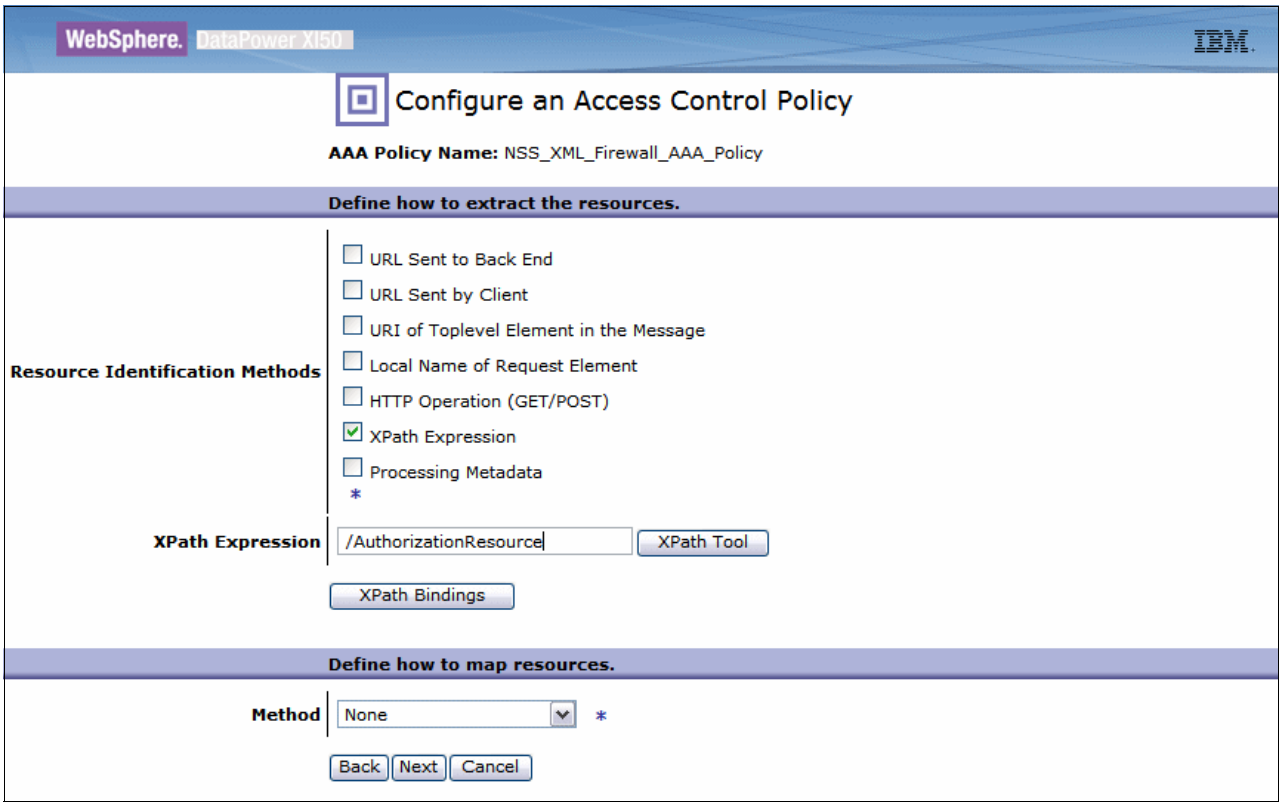


Figure 10-60 How to extract the resource for authorization

17. In Figure 10-61, define the authorization method by selecting **Contact NSS for SAF Authorization**. Select the SAF Client Configuration named **NSSXML1**. In our testing, we needed only Read authorization to the resource, so we selected that as the Default Action. Click **Next**.

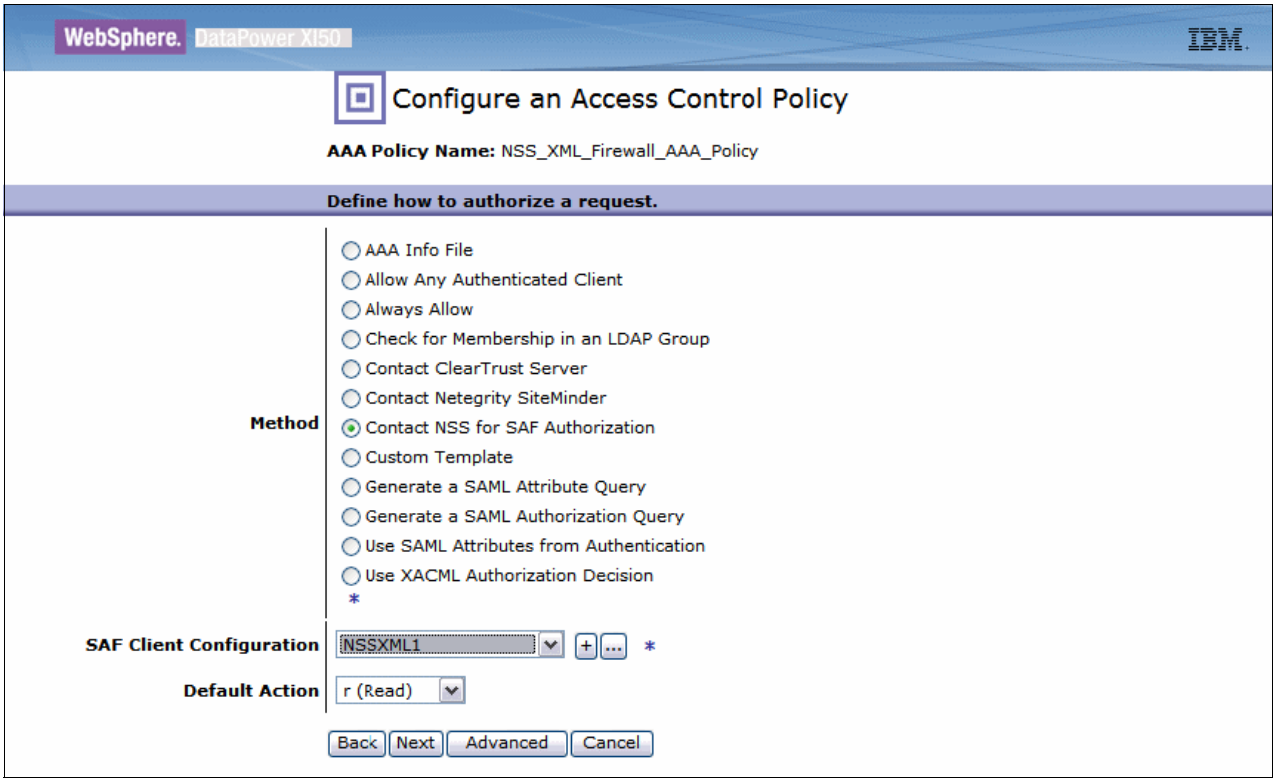


Figure 10-61 How to authorize the request for a SERVAUTH resource

18. You do not want to change any options for Monitoring or Logging activity because you do not need post-processing. Accept the defaults shown in Figure 10-62, and click **Commit**.

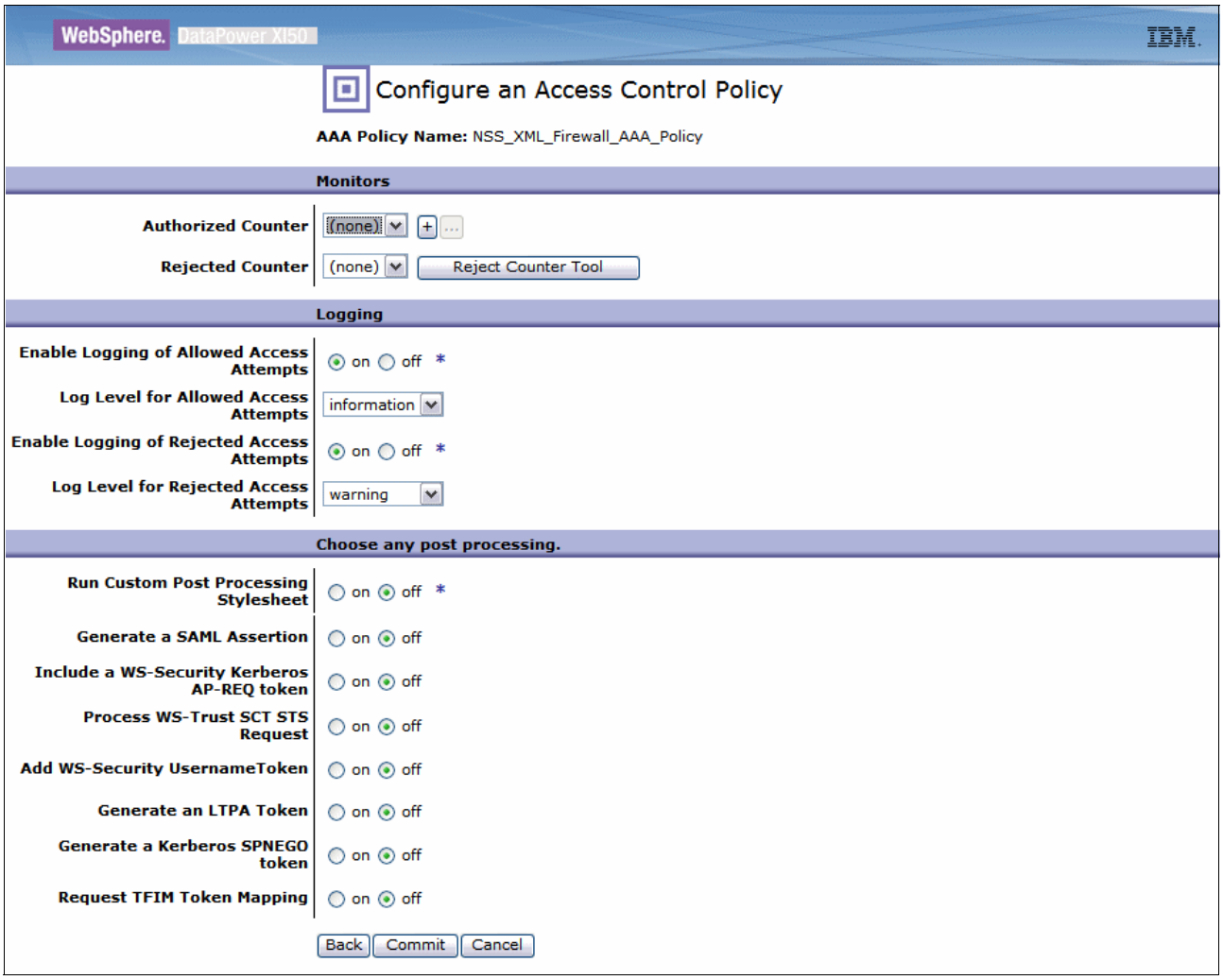


Figure 10-62 Monitors, logging, and post-processing for an AAA Policy

19. You receive a message that the AAA Policy was created successfully (Figure 10-63). Click **Done** in this message window and in the subsequent window.

Figure 10-63 Successful AAA Policy definition

20. Back at the XML Firewall Policy window, drag the Results icon (highlighted in Figure 10-64) to the Policy Rule line (as shown in Figure 10-65 on page 511).

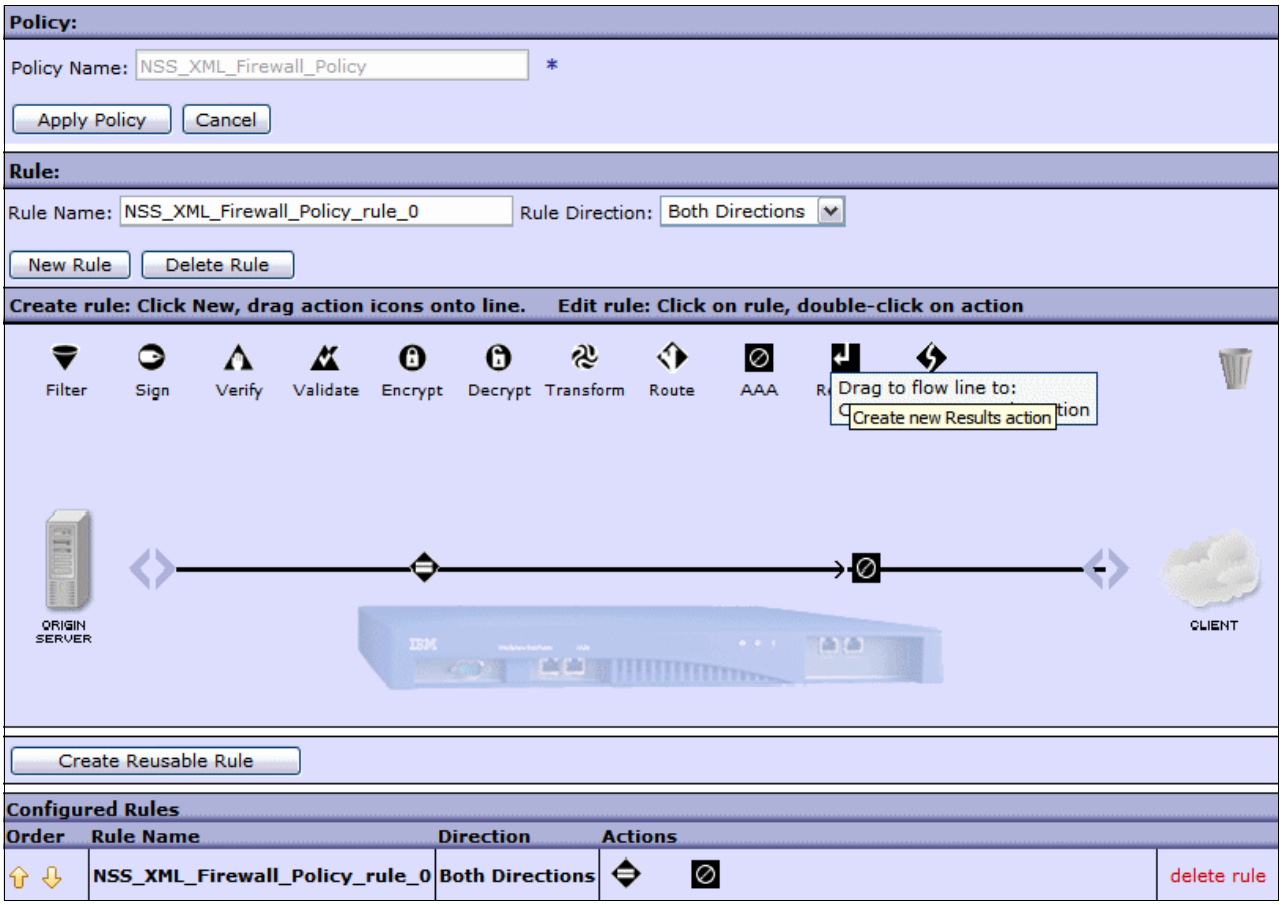


Figure 10-64 Completing the policy rule with a results action (Part 1)

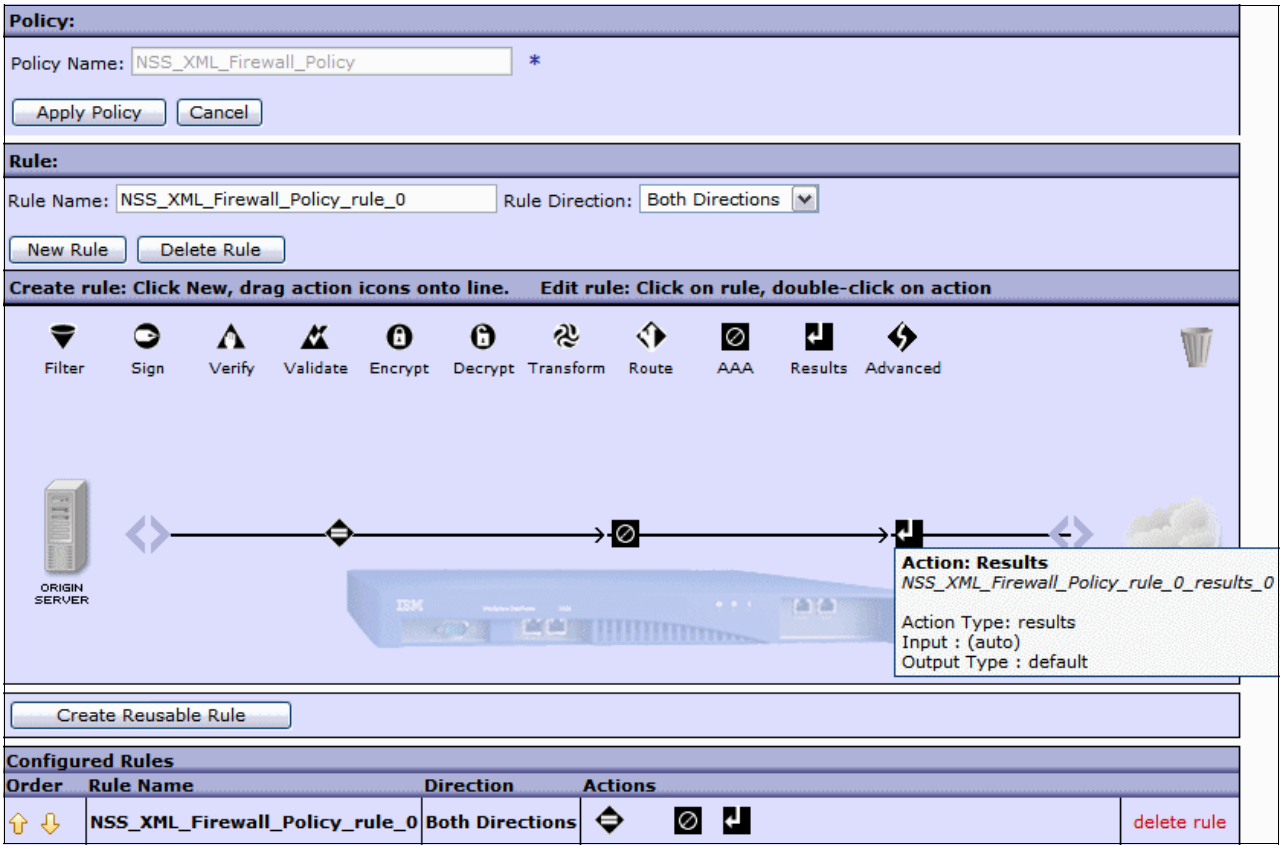


Figure 10-65 Completing the policy rule with a results action (Part 2)

21. Double-click the Results icon on the Policy Rule line to define the action to which the authentication and authorization process should lead (Figure 10-66). Enter `INPUT` for Input and `OUTPUT` for Output, and then click **Done**.

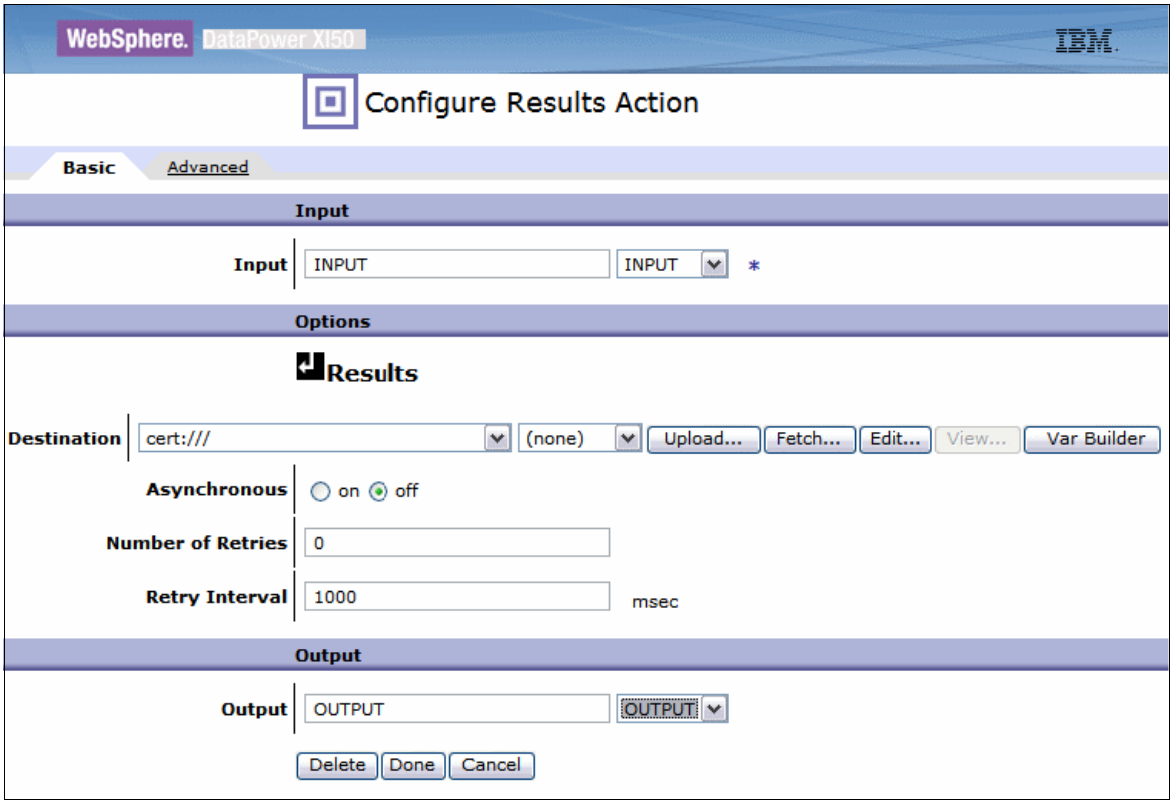


Figure 10-66 Configuring the Results Action

22. Click **Apply Policy** in the next window that displays (Figure 10-67). Then close this window using a button in the upper-right of the window.

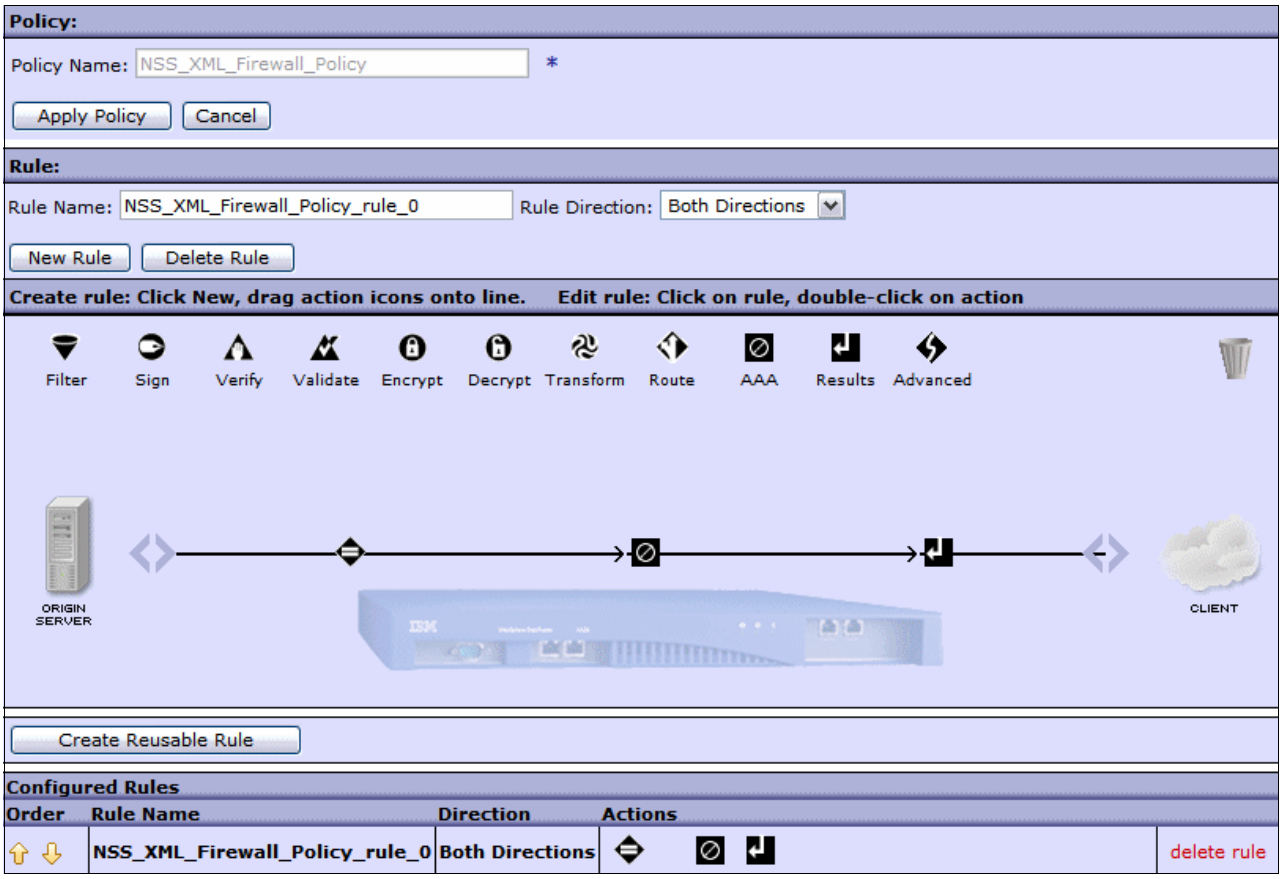


Figure 10-67 Completed policy: Match action, AAA action, and Results action

23. Back in the window where you started (shown in Figure 10-68), click **Apply** and then **Close Window**.

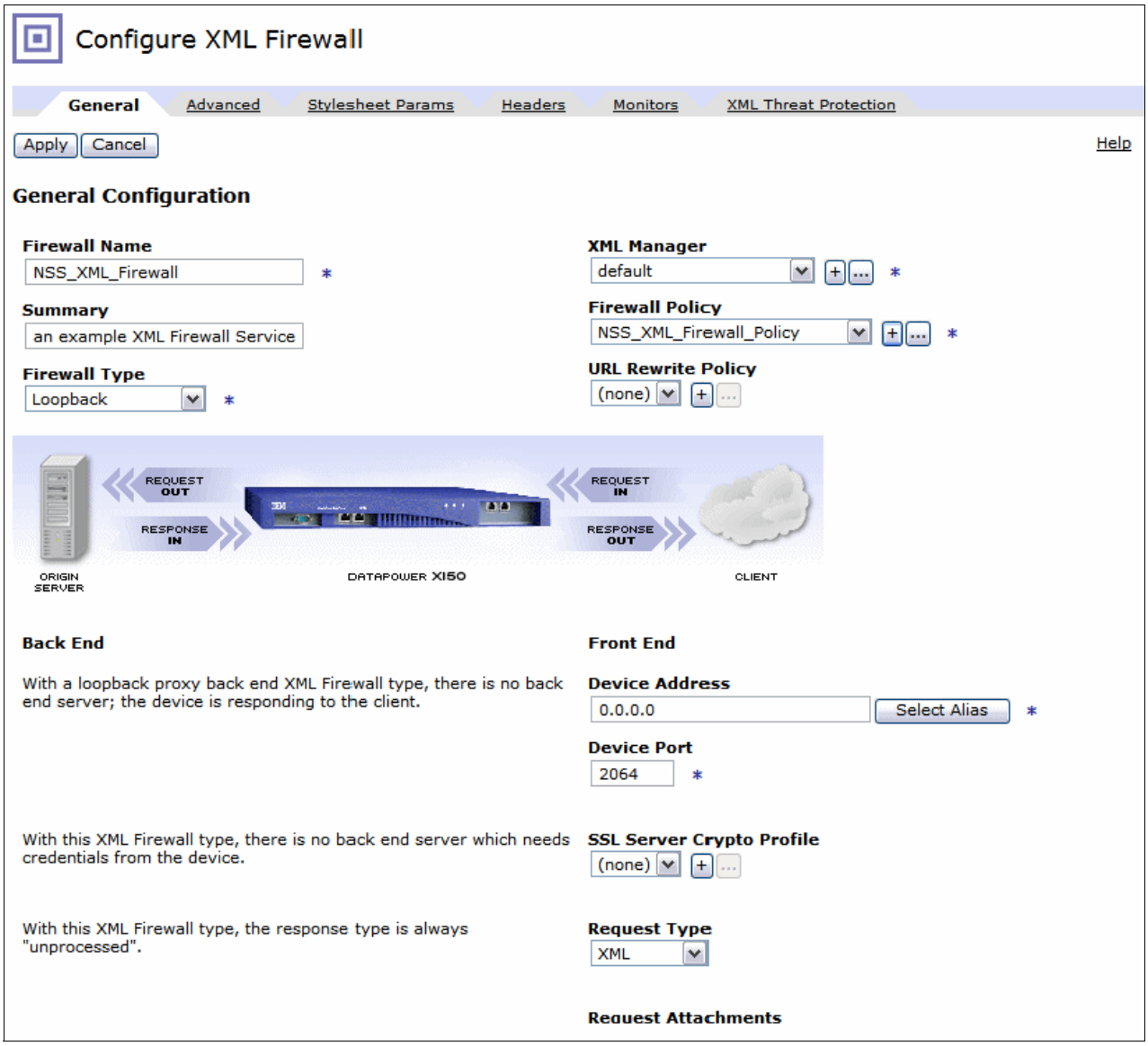


Figure 10-68 Completed XML firewall configuration

10.2.6 Configuring the NSS environment at the Web Services Requester

The platform that we used for testing the entire NSS XML environment was a Windows XP system running on an IBM x platform. You can use any platform and operating system that can issue web services requests, such as Linux, HP, MAC, IBM System p® series, and so forth. However, the platform that you use must be capable of executing a Web Services Requester program.

Figure 10-69 represents the portion of the entire NSS environment that depicts the Web Services Requester environment and its relationship to other components.

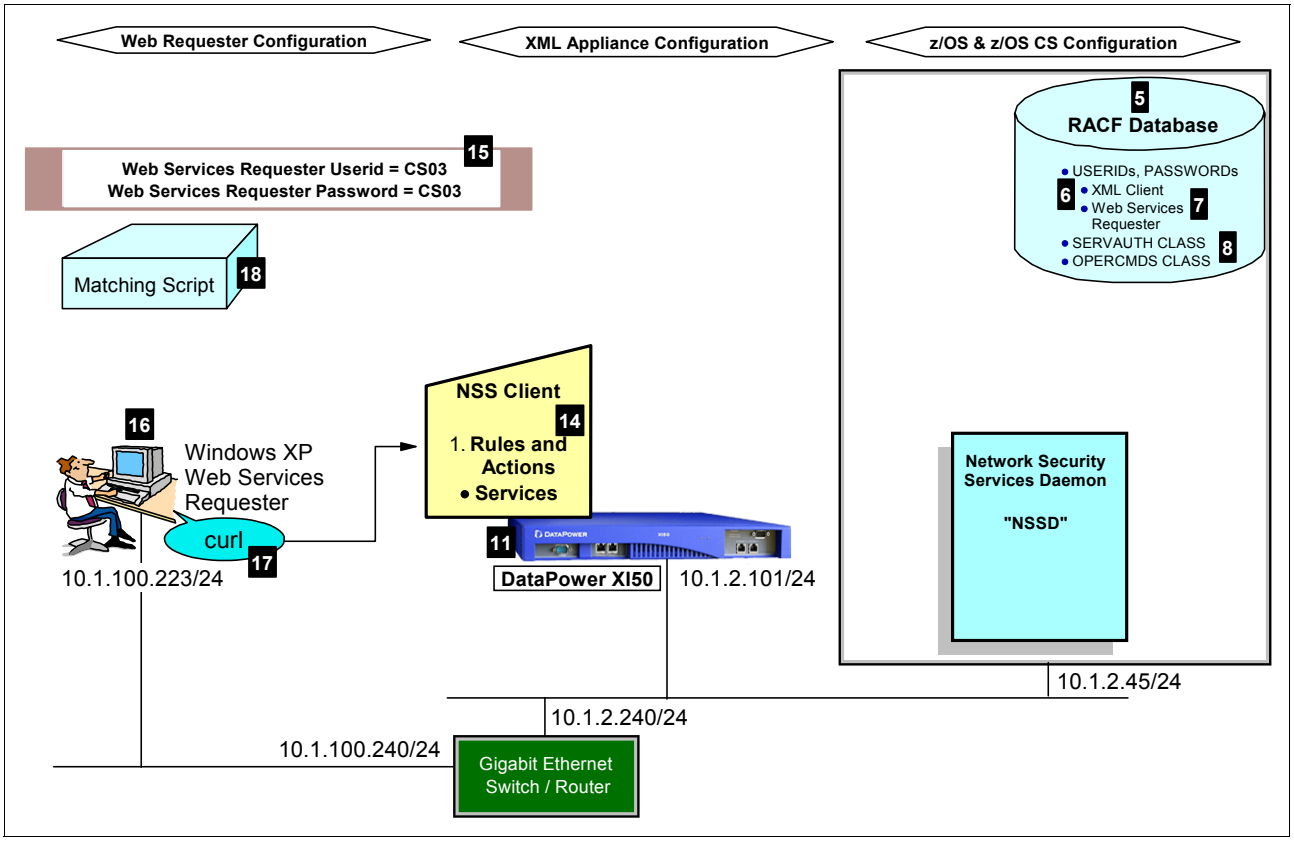


Figure 10-69 Configuration for Web Services Requester platform in an NSS environment

For our purposes, we used the **curl** command (17) instead of a WebSphere application as the Web Services Requester program. The **curl** command allows you to employ URL syntax to transfer files and support SSL certificates. Because it is simple to use, the command is a useful tool for quick testing in an environment such as ours.

You can download **curl** as freeware onto the Web Services Requester platform (16) from any number of web sites. We downloaded **curl** from the following website:

`http://curl.haxx.se/download.html`

We selected the Windows Generic WIN32 binary SSL Version 7.19.1 package that is 260 KB and downloaded it to the Windows XP platform. We experienced installation issues with the WINXP2000 version on our platform after our download and installation, but we had no problems with the Generic WIN32 package.

We then edited a Matching Script file on the workstation as shown in Example 10-34.

Example 10-34 Matching Script to match Match Rule and Action in the DataPower appliance

```
<AuthorizationResource>EZB.DB2.DB2PROCA.SALES</AuthorizationResource>
                                              A                       B
```

Note how the contents of the script contain a pattern, `/AuthorizationResource` (B), that matches the XPATH expression coded in the DataPower appliance. That XPATH expression represents filter or condition that is used to invoke a subsequent action to authenticate the user ID and password (15) and the resource (A) at the System z RACF SERVAUTH definitions (8).

Note: The NSS Server authenticates the user ID and password that submits the Web Services Request and also authorizes that user to a specific SERVAUTH resource that is defined to RACF. In our scenario, we authorized our user to access a SERVAUTH resource named `EZB.DB2.DB2PROCA.SALES`.

Each application process to which the Web Services Requester requires access might need a different type of SERVAUTH resource authorization. You need to be prepared to define and authorize that resource at RACF prior to any testing. We define this SERVAUTH resource during the verification process in 10.3, “Verifying the NSS configuration with the NSS Client (XML Appliance Discipline)” on page 516.

We stored the script named `newhttppayload.xml` in the following directory, which is the same directory as the **`curl`** command that we downloaded from the web:

```
C:\Program Files\CURL\curl-17.19.0-devel-mingw32\bin\
```

We use the POST method of the **`curl`** command to test the authentication of a Web Services Requester user ID (15) and its authorization to a System z resource named *EZB.DB2.DB2PROCA.SALES*.

10.3 Verifying the NSS configuration with the NSS Client (XML Appliance Discipline)

Verify the following types of operations for the NSS Server relationship to the NSS Client, as shown in Figure 10-69 on page 515:

- z/OS and z/OS Communications Server and NSSD configuration and operations
- DataPower appliance configuration and operations
- Web Services Requester configuration and operations

10.3.1 Operations with z/OS NSS Server

To verify the operations for the z/OS NSS Server, follow these steps:

1. Log on to the z/OS system and from the console and start the policy agent with the new AT-TLS policy for the NSS Server (Example 10-35).

Example 10-35 Initializing policy agent with policy for NSS Server and NSS 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
 EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS
 EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : IPSEC
 EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : TTLS   a
 EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER
```

TCPIPE (a) shows that the AT-TLS policy is now active.

If policy agent was already executing, you can simply update the running policies with the following command:

```
F PAGENT,UPDATE
```

2. Start the NSSD server on z/OS system ID of SC33 as shown in Example 10-36.

Example 10-36 NSSD Initialization at z/OS SC33

```
S NSSD
$HASP100 NSSD     ON STCINRDR
IEF695I START NSSD     WITH JOBNAME NSSD     IS ASSIGNED TO USER NSSD
   , GROUP TCPGRP      a
$HASP373 NSSD     STARTED
EZD1357I NETWORK SECURITY SERVICES (NSS) SERVER INITIALIZATION
SEQUENCE HAS BEGUN
IEE252I MEMBER CTINSS00 FOUND IN SYS1.PARMLIB
EZD1373I NSS IPSEC DISCIPLINE ENABLED      b
EZD1373I NSS XMLAPPLIANCE DISCIPLINE ENABLED      c
EZD1353I NSS SERVER CONFIG PROCESSING COMPLETE USING FILE
/etc/security/nssd33.conf      d
EZD1358I NSS SERVER INITIALIZATION SEQUENCE HAS COMPLETED
```

The NSS server initializes under the OMVS segment associated with user ID NSSD (a). This is the user ID that was defined previously. Notice messages EZD1373I at b and c. These messages are informational messages that explain that both discipline types for the NSS server are available. Observe in message EZD1353I (d) that we used the NSSD configuration file that was customized earlier (`/etc/security/nssd33.conf`).

You can display more information about the NSS server configuration with the **`modify`** command:

```
F NSSD,DISPLAY
```

Example 10-37 shows the results of this command.

Example 10-37 Results of NSSD, DISPLAY command

```
F NSSD,DISPLAY
EZD1386I DISPLAY NSS CONFIGURATION 726
DISPLAY Network Security Server Configuration Parameters:
    Port        = 4159                    1
    SyslogLevel = 255     (0x00ff)        2
    KeyRing     = "NSSD/NSSD_keyring"     3
    ---------------------------------
    Discipline IPSec        = Enabled     4
    Discipline XMLAppliance = Enabled     4
```

In this example, the numbers correspond to the following information:

- Displays the listening port for the NSS server (1).
- Displays the SYSLOGLEVEL of 255 (2). This is a high level that is used for problem determination. After it is verified that the NSS server is operating as desired, you can reduce this logging level to 15.
- Displays the key ring owner and key ring name for the IPSec discipline (3). Note that the key ring name for the AT-TLS process is not displayed because that is under the control of the AT-TLS policy in policy agent.
- Displays the status of the available NSS disciplines (4) (IPSec and XMLAppliance).

3. Issue the **netstat** command to determine which ports the NSS server is connected to as shown in Example 10-38.

Example 10-38 Displaying connections for the NSS server

```
D TCPIP,TCPIPE,N,CONN,CLIENT=NSSD
USER ID  CONN     STATE
NSSD     00000F7D LISTEN
  LOCAL SOCKET:   ::..4159
  FOREIGN SOCKET: ::..0
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

The only active connection is the connection of the server to the listening port of 4159.

Remember that in the customization of the DataPower appliance, you always enable each object as you define it. If those objects are still enabled in the appliance and if the z/OS definitions for the NSS server with an NSS client are correct, you can see an active connection between the server and the NSS client. Because there is no established connection, you return to the DataPower appliance and enable all the objects that are associated with the NSS client. Then you can view an established connection between the client and the server when we execute a command from the SDSF z/OS console.

4. Execute the following command to see the status of the server-client relationship as shown in Example 10-39 on page 519.

```
D TCPIP,TCPIPE,N,CONN,CLIENT=NSSD
```

Example 10-39 Connection Status with the NSS server filtered on client ID of NSSD

```
D TCPIP,TCPIPE,N,CONN,CLIENT=NSSD
USER ID  CONN     STATE
NSSD     00001603 ESTBLSH    1
  LOCAL SOCKET:   ::FFFF:10.1.2.45..4159
  FOREIGN SOCKET: ::FFFF:10.1.2.101..20366
NSSD     00000013 LISTEN      2
  LOCAL SOCKET:   ::..4159
  FOREIGN SOCKET: ::..0
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the numbers correspond to the following information:

– Displays that there is a connection established with a partner at address 10.1.2.101, the address of the DataPower appliance (1). The Connection ID is 1603.

– Displays the listening socket for the NSS server (2).

You need the details about the connection between the NSS server and the NSS client. Use the **netstat** command from the SDSF console and filter on the connection ID that was extracted at 1 in Example 10-39. Note that the example also specifies to see only a TTLS connection because we know that the connection should not have been established without AT-TLS.

Example 10-40 shows the results of the following command:

```
D TCPIP,TCPIPE,N,TTLS,CONN=1603,DETAIL
```

Example 10-40 Detailed Connection Status with the NSS server filtered on Connection ID

```
D TCPIP,TCPIPE,N,TTLS,CONN=1603,DETAIL 3
CONNID: 00001603
  JOBNAME:      NSSD
  LOCALSOCKET:  ::FFFF:10.1.2.45..4159
  REMOTESOCKET: ::FFFF:10.1.2.101..20366
  SECLEVEL:     TLS VERSION 1 4
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA 5
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
TTLSRULE: NSSSERVERRULEAT-TLS~1 6
  PRIORITY:       255 7
  LOCALADDR:      0.0.0.0/0
  LOCALPORT:      4159
  REMOTEADDR:     0.0.0.0/0
  REMOTEPORTFROM: 1024               REMOTEPORTTO: 65535
  DIRECTION:      INBOUND
  TTLSGRPACTION:  GACT1~NSS_SERVER     8
    GROUPID:                    00000005
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      255
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
  TTLSENVACTION:  EACT1~NSS_SERVER 8
    ENVIRONMENTUSERINSTANCE:    1
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1
```

```
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    CLIENTAUTHTYPE:             REQUIRED
  TTLSCONNACTION: CACT1~NSS_SERVER 8
    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
                                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
```

Example 10-40 on page 519 displays exactly what was coded in the AT-TLS policy that we discussed in Example 10-27 on page 462.

- The command that we executed to obtain the output for this particular TTLS session (3).
- TLS V1 was negotiated for this secured session between the server and the client (4).
- The negotiated encryption algorithm (5).
- The AT-TLS policy rule name and the associated actions that correspond to the session (6).

  These are the policy rules and actions that we defined with the IBM Configuration Assistant in Chapter 7, “IP filtering” on page 217 and Chapter 8, “IP Security” on page 245.
- The Rule Priority (7).

  This is significant because multiple, applicable TTLS policies have been loaded into the running stack. However, this is the TTLS policy rule with the highest priority and is therefore the one that is selected.
- The AT-TLS policy rule action names (8) that are associated with the selected policy rule (6).

  These are the policy rules and actions that we defined with the IBM Configuration Assistant in Chapter 7, “IP filtering” on page 217 and Chapter 8, “IP Security” on page 245.

5. Now that because a secure connection is established between the NSS Server and the NSS client, you can view this connection in the z/OS UNIX System Services environment from the UNIX perspective. From the ISPF command line, enter the following command:

   ```
   TSO OMVS
   ```

   Switch to superuser mode to view the data in the syslog directory:

   ```
   su
   ```

   Eventually, you will want to look at log messages in SYSLOGD. To do so, go to the `syslog` directory and execute the following command without filtering on NSS client name:

   ```
   nssctl -d
   ```

Example 10-41 shows the output of this command.

Example 10-41 Monitoring NSS with the nssctl command

```
CSO3 @ SC33:/SC33/tmp/syslog>nssctl -d

CS nssctl  SystemName: SC33      Mon Nov 10 18:47:29 2008
Function: Display           NSSClientName: n/a

ClientName:                      NSSXML1   1
ClientAPIVersion:                3
StackName:
SystemName:                      SC33
ClientIPAddress:                 ::ffff:10.1.2.101 2
ClientPort:                      20366
ServerIPAddress:                 ::ffff:10.1.2.45 3
ServerPort:                      4159
UserID:                          NSSXML 4
ConnectState:                    connected
TimeConnected:                   2008/11/10 18:39:44
TimeOfLastMessageFromClient:     2008/11/10 18:39:44
Discipline:                      XMLAppliance 5
  CertificateServiceSelected:    No 6
  CertificateServiceEnabled:     No 6
  PrivateKeyServiceSelected:     No 7
  PrivateKeyServiceEnabled:      No 7
  SAFAccessServiceSelected:      Yes 8
  SAFAccessServiceEnabled:       Yes                                    8
************************************************************

1 entries selected
CSO3 @ SC33:/SC33/tmp/syslog>
```

In this example, the numbers correspond to the following information:

- The symbolic NSS Client name (1) that was established when the NSS middleware appliance configuration was built in 10.2.5, “Configuring the NSS environment at the WebSphere DataPower SOA Appliance to support the SAF access service” on page 481.
- The NSS Client’s IP address (2).
- The NSS Server’s IP address (3).
- The RACF User ID (4) that has been assigned to the client. This is the name that the client presented to the NSS server to establish the secure connection.
- The discipline that this NSS client is using with the NSS server (5).
- Shows that the Certificate Service has neither been selected nor enabled (6).
- Shows that the Private Key Service has neither been selected nor enabled (7).
- Shows that the Client has requested SAF Access Service (8).
- Shows that the connection has successfully negotiated SAF Access Service (9).

6. You can explore the various syslog daemon logs and find the entries that confirm what you see in the displays. That is, you can see that the negotiations are successful.
Example 10-42 shows the output of SYSLOG daemon (and also in the isolated NSSD log of `LOCAL44.log`).

Example 10-42 Successful connection establishment for NSS Daemon and NSS Client

```
NSSD: DBG0031I NSS_VERBOSE UseridCheck(NSSXML,******) passed   1
NSSD: DBG0038I NSS_VERBOSE (00001.ezd.nss.connID....) connID 1 renamed to (NSSXML1 ) 2
NSSD: DBG0032I NSS_VERBOSE
ProfileCheck(NSSXML,EZB.NSS.SC33.NSSXML1.XMLAPPLIANCE.SAFACCESS) rc 0 (AUTH) racfRC 0
racfRsn 0 3
NSSD: DBG0103I NSS_LIFECYCLE  NSS connID  1 ::ffff:10.1.2.101..57602 - NSSXML1
,SC33    ,        ,NSSXML  ,0x02,0x02 - NSS_ConnectClientReqToSrv request accepted -
Cert=No RemoteMgmt=No SAFAccess=Yes 4
NSSD: DBG0035I NSS_VERBOSE ConnectClientReqToSrv clientname (NSSXML1      ) connID 1 rc
0 reason 0 CertService N RemoteMgmtService N SAFAccessService YBB2B83A845D16AC 5
```

In this example, the numbers correspond to the following information:

- The RACF user ID presented by the NSS client passed authentication (1).
- The client name of NSSXML1 has now been assigned to the connection (2).
- The RACF user ID is authorized to access the SERVAUTH profile that we created earlier (3).
- The type of access (4) that the NSS client user ID (NSSXML) is requesting (`SAFAccess=Yes`).
- The type of service (5) that the NSS client name (NSSXML1) is requesting, `SAFAccessService Y`, which is the type of service required by an NSS client and not by an NSS IPSec client.

7. To see the SSL/TLS negotiation, you can peruse the general syslog daemon log. Example 10-43 shows the messages that are written to the log when a high level of debugging is enabled.

Example 10-43 AT-TLS negotiation for NSS connection in z/OS log

```
a
TTLS Map   CONNID: 00001603 LOCAL: ::FFFF:10.1.2.45..4159 REMOTE:
::FFFF:10.1.2..101..20366 JOBNAME: NSSD USERID: NSSD TYPE: InBound STATUS: Appl Control
RULE: NSSServerRuleAt-TLS~1 ACTIONS: gAct1~NSS_Server eAct1~NSS_Server cAct1~NSS_Server
TTLS Event GRPID: 00000005 ENVID: 00000000 CONNID: 00001603  RC:    0 Connection
b
TTLS Start GRPID: 00000005 ENVID: 00000005 CONNID: 00001603 Initial Handshake ACTIONS:
gAct1~NSS_Server eAct1~NSS_Server cAct1~NSS_Server HS-Server
c
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Call
GSK_SECURE_SOCKET_OPEN - 7E7E6718
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Set
GSK_SESSION_TYPE -  SERVER
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Set
GSK_V3_CIPHER_SPECS -  090A2F
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Set
GSK_KEYRING_LABEL -  CS19 ITSO SharedSite1
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Set GSK_FD -
00001603
TTLS Flow  GRPID: 00000005 ENVID: 00000005 CONNID: 00001603  RC:    0 Set GSK_USER_DATA
- 7E8C7250
d
TTLS Data  CONNID: 00001603 RECV CIPHER 160301007F

TTLS Data  CONNID: 00001603 RECV CIPHER
0100007B03014931BBFACA6F5576FE22B01DB28D6C704F06E3D4CA27F2AED1FF61336215035B20000400FD00
00000000000000000FFFF0A010265B

TTLS Data  CONNID: 00001603 SEND CIPHER
160301004A02000046030149 31BC3A598A017F2A5C9E114DA15A0C169ACBCB4B25804B754A9A37C401CF9320
000400FD000000000000000000000FFF
TTLS Data  CONNID: 00001603 SEND CIPHER 140301000101
e
```

In this example:

- a shows a connection (#1603) that is mapped to a set of AT-TLS rules and actions.
- b is the start of the SSL handshake.
- c shows the secure API calls with information about handshake roles, cipher specs, and certificate name that should be used by the NSS server.
- From d to e shows the exchange of cipher specs for the SSL/TLS connection.

10.3.2 Operations with the DataPower appliance and Client

As discussed in 10.2.5, “Configuring the NSS environment at the WebSphere DataPower SOA Appliance to support the SAF access service” on page 481, each object for the NSS client must be enabled and operational to initiate the secure connection between the NSS server and the NSS client. Loading the appropriate x.509 certificate chain into the DataPower appliance and configuring the NSS Client object is a fairly simple task, as discussed in “The Cryptographic objects configuration” on page 484 and “The NSS client configuration (XML Appliance Discipline)” on page 494.

In this section, we examine the log of the DataPower appliance to view entries for the NSS CLient-Server connection. Before testing, we changed the logging value to debug using the Control Window Troubleshooting icon (Figure 10-70).

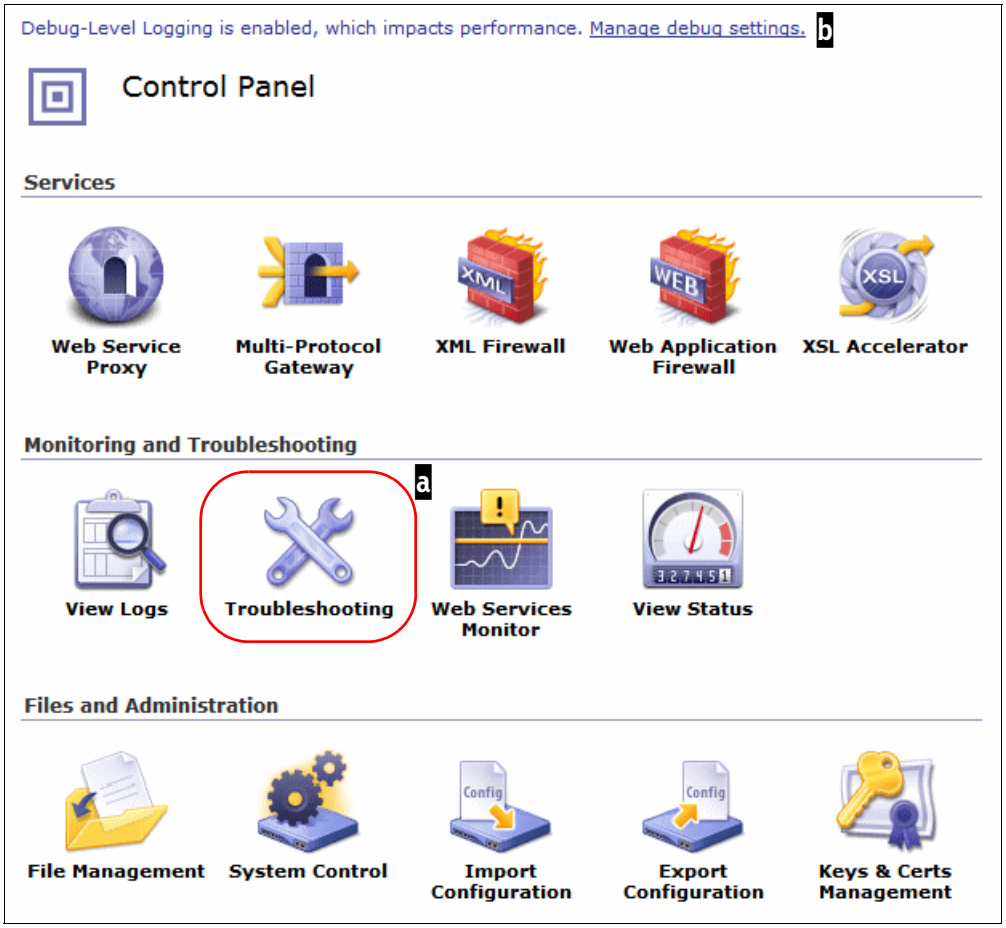


Figure 10-70 Managing logging levels for debugging

In the figure:

- Choose the icon (a) to manage tracing and logging.
- As long as debug logging is enabled, the warning message (b) reminds you that you need to disable debug logging when it is no longer needed because it adversely affects performance.

Example 10-44 shows the results of debug-level logging.

Example 10-44 Establishing connection between NSS Server and NSS XMLAppliance Client

```
zos-nss (NSSXML1): Operational state up                7
zos-nss (NSSXML1): StateChange(0x9ad913e0): finishing [event=0connection
established] for NSSXML1                6
zos-nss (NSSXML1): StateChange(0x9ad913e0): finishing [event=0NSS handshake in
progress] for NSSXML1
valcred (CERTCA19VAL): certificate validation succeeded for '/C=US/O=IBM
Corporation/OU=ITSO CS19 Shared SITE/CN=ITSO.IBM.COM' against
'CERTCA19VAL'           5
zos-nss (NSSXML1): StateChange(0x9ad913e0): finishing [event=0TCP connection in
progress] for NSSXML1          4
zos-nss (NSSXML1): NSSXML1(0x9ad913e0) DNS returned ip 10.1.2.45
zos-nss (NSSXML1): NSSXML1 (0x9ad913e0) resolving hostname - 10.1.2.45     3
zos-nss (NSSXML1): commit-ssl profile(0xa5c42c8) for NSSXML1            2
zos-nss (NSSXML1): config cleared NSSXML1
zos-nss (NSSXML1): Pending
zos-nss (NSSXML1): config commit called NSSXML1           1
```

The log entries are listed from latest to oldest, so we examine the log reading from the bottom:

1. We disabled and re-enabled the NSS Client named NSSXML1.
2. The SSL profile configured for NSSXML1 is applied to the activation of the connection with the NSS server.
3. The NSSXML1 client begins its activation with the NSS server at IP address `10.1.2.45`.
4. The TCP connection between the server and the client is in the final stages.
5. The SSL/TLS negotiation is in progress, with the server having sent a copy of its server certificate to the NSS client for validation.
6. The SSL/TLS handshake is in progress and the SSL/TLS connection is successfully completed.
7. The NSSXML1 client is now up.

Figure 10-71 shows an excerpt from the full log that is displayed in the XML Management GUI in tabular format.

current time: 18:13:49 on 2008-11-28

time	category	level	tid	dir	client	msgid	message	Show last 50 100 all
Fri Nov 28 2008								
18:13:23	mgmt	notice	33823			0x00350014	zos-nss (NSSXML1): Operational state up	
18:13:23	zosnss	debug	33839			0x82a0000f	zos-nss (NSSXML1): StateChange(0x9ad913e finishing [event=0connection established] for NSSXML1	
18:13:23	zosnss	debug	33839			0x82a0000f	zos-nss (NSSXML1): StateChange(0x9ad913e finishing [event=0NSS handshake in progress] NSSXML1	
18:13:23	crypto	info	69027			0x8060010b	valcred (CERTCA19VAL): certificate validati succeeded for '/C=US/O=IBM Corporation/OU CS19 Shared SITE/CN=ITSO.IBM.COM' agai 'CERTCA19VAL'	

Figure 10-71 Excerpt of log from DataPower appliance

The Firewall configuration and all of its rules and actions can become quite complex, so you need to verity that configuration. Using a web browser, connect to the XML console and click **Control Window** → **Status** → **Firewall** to reach the window shown in Figure 10-72. The information here explains whether the Firewall configuration is complete and whether the z/OS NSS client configuration is associated correctly with the cryptographic and firewall policies.

NSS_XML_Firewall [XML Firewall Service]	New	up	enabled		
default [XML Manager]	Saved	up	enabled		
default [User Agent]	Saved	up	enabled		
NSS_XML_Firewall_Policy [Processing Policy]	New	up	enabled		
NSS_XML_Firewall_MatchRule [Matching Rule]	New	up	enabled		
NSS_XML_Firewall_Policy_rule_0 [Processing Rule]	New	up	enabled		
NSS_XML_Firewall_Policy_rule_0_aaa_0 [Processing Action]	New	up	enabled		
NSS_XML_Firewall_AAA_Policy [AAA Policy]	New	up	enabled		
NSSXML1 [zOS NSS Client]	Saved	up	enabled		
CACERT19Prof [SSL Proxy Profile]	Saved	up	enabled		
CACERT19Profile [Crypto Profile]	Saved	up	enabled		
CERTCA19VAL [Crypto Validation Credentials]	Saved	up	enabled		
CACERT19 [Crypto Certificate]	Saved	up	enabled		
NSSXML1 [zOS NSS Client]	Saved	up	enabled		
CACERT19Prof [SSL Proxy Profile]	Saved	up	enabled		
CACERT19Profile [Crypto Profile]	Saved	up	enabled		
CERTCA19VAL [Crypto Validation Credentials]	Saved	up	enabled		
CACERT19 [Crypto Certificate]	Saved	up	enabled		
NSS_XML_Firewall_Policy_rule_0_results_0 [Processing Action]	New	up	enabled		

Figure 10-72 View of NSS Client services and objects and their dependencies

Figure 10-72 on page 526 shows the hierarchy shown in Figure 10-73.

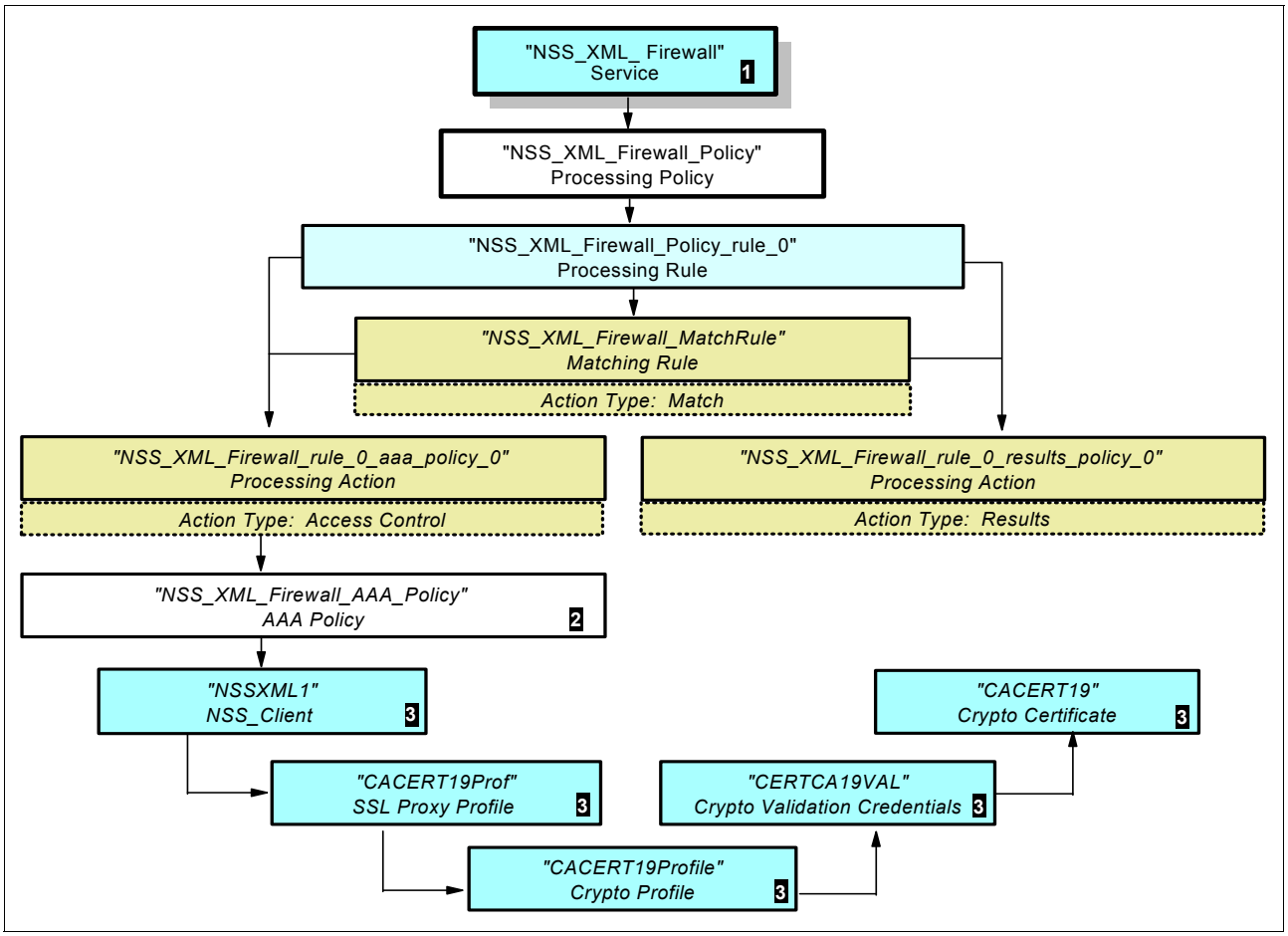


Figure 10-73 Specifications for the NSS configuration

All of the objects are associated correctly. The testing of the NSS client and its security configuration, depicted in the lower sections the figure, is complete.

Next, you need to test the Firewall policy and AAA policy when you verify operations at the Web Services Requester platform.

10.3.3 Operations with the Web Services Requester platform

Now, you can submit a web services request from the workstation. This last step verifies the complete operation of using the NSS Server as a centralized security authentication and authorization service for middleware appliances.

Prerequisite verification

To begin, you must have an active and secured connection between the NSS Server and the NSS Client. You must also have a SERVAUTH class defined for the Web Services Requester against which to authorize. We defined the SERVAUTH class and permitted our Web Services Requester user to this class, as shown in Example 10-45 on page 528. Our requester user ID is `CS03` and belongs to the RACF group of `SYS1`.

Example 10-45 Defining a SERVAUTH class

```
RDEFINE SERVAUTH EZB.DB2.DB2PROCA.SALES UACC(NONE)
PERMIT EZB.DB2.DB2PROCA.SALES CLASS(SERVAUTH) ID(SYS1) ACCESS(READ)
setropts generic(servauth) refresh
setropts raclist(servauth) refresh
```

This class (EZB.DB2.DB2PROCA.SALES) is named in the Matching Script that was built on the Web Requester platform in Example 10-34 on page 516.

Submitting an HTTP Request from the Web Services Requester

At the workstation that represents the Web Services Requester platform, move to the **curl** command directory and execute the following command:

```
C:\Program Files\CURL\curl-7.19.0-devel-mingw32\bin>curl -v -X POST
-d@newhttppayload.xml -u cs03:cs03 http://10.1.2.101:2064
```

In our testing, we wanted the output of the **curl** command to be verbose (**-v**) and the data contained within the file newhttppayload.xml to be sent in the HTTP header together with the user ID of CS03 (whose password is CS03). The HTTP request is to be sent to a service running on port 2064 at URL 10.1.2.101. (2064 is the port that is associated with the XML firewall service that was defined earlier.) Example 10-46 shows the resulting output at the workstation.

Example 10-46 Results of HTTP Request directed to the DataPower appliance

```
C:\Program Files\CURL\curl-7.19.0-devel-mingw32\bin>curl -v -X POST
-d@newhttppayload.xml -u cs03:cs03 http://10.1.2.101:20641
* About to connect() to 10.1.2.101 port 2064 (#0)
*   Trying 10.1.2.101... connected
* Connected to 10.1.2.101 (10.1.2.101) port 2064 (#0)    2
* Server auth using Basic with user 'cs03'               3
> POST / HTTP/1.1
> Authorization: Basic Y3MwMzpjczAz
> User-Agent: curl/7.19.0 (i386-pc-win32) libcurl/7.19.0 OpenSSL/0.9.8h zlib/1.2.3
libssh2/0.18
> Host: 10.1.2.101:2064
> Accept: */*
> Content-Length: 69
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 Good
< Authorization: Basic Y3MwMzpjczAz
< User-Agent: curl/7.19.0 (i386-pc-win32) libcurl/7.19.0 OpenSSL/0.9.8h zlib/1.2.3
libssh2/0.18
< Host: 10.1.2.101:2064
< Content-Type: application/x-www-form-urlencoded
< Via: 1.1 NSS_XML_Firewall      4
< X-Client-IP: 10.1.100.223      5
< Transfer-Encoding: chunked
<
<?xml version="1.0" encoding="UTF-8"?>
<AuthorizationResource>EZB.DB2.DB2PROCA.SALES</AuthorizationResource><?xml
version="1.0" encoding="UTF-8"?>      6
<AuthorizationResource>EZB.DB2.DB2PROCA.SALES</AuthorizationResource>* Connection #0
to host 10.1.2.101 left intact       7
* Closing connection #0
```

In this example, the numbers correspond to the following information:

1. We issue the **curl** command from the appropriate directory of the Web Services Requester platform.
2. The connection is established to the firewall service on port 2064 of the DataPower appliance platform.
3. We request user authentication services for user CS03.
4. The name of the service at port 2064 is NSS_XML_Firewall.
5. The client issuing the web request is at IP address 10.1.100.223.
6. The data resource for which CS03 must be authorized is EZB.DB2.DB2PROCA.SALES.
7. The connection is left open for a bit, but it is then closed.

The example here shows no blatant error messages. However, there are no messages to indicate that authentication and authorization have taken place either. Therefore, we need to peruse the logs at the middleware appliance and at the NSS server platform as described in the next section.

Reviewing logs at the DataPower appliance

The logs (still at a logging level of DEBUG) reveal that the authentication and authorization of the Web Services Requester user ID were successful as shown in Example 10-47. Again, we read from the bottom up, because the latest messages display at the top of the log.

Example 10-47 DataPower Log for authenticating and authorizing Web Services Requester

```
xmlfirewall (NSS_XML_Firewall): rule (NSS_XML_Firewall_Policy_rule_0): #2 results:
'generated from INPUT results stored in OUTPUT' completed ok. 14
xmlfirewall (NSS_XML_Firewall): rule (NSS_XML_Firewall_Policy_rule_0): #1 aaa:
'INPUT NSS_XML_Firewall_AAA_Policy stored in OUTPUT' completed ok.   13
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Message
allowed
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): zosnss
authorization succeeded with credential 'cs03' for resource
'EZB.DB2.DB2PROCA.SALES'    12
.......................
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Calling
zosnss-author[NSSXML1,cs03,r,EZB.DB2.DB2PROCA.SALES]    11
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): The operation
for the SAF authorization request is r           10
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Authorizing
with "zosnss"       9
.......................
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): zosnss
authentication succeeded with (http-basic-auth,
username='cs03'     password='********'configured-realm='login' ) 8
.......................
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Using
zOS/NSS(NSSXML1)     7
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy):
Authenticating with "zosnss"     6
.......................
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Resource
"EZB.DB2.DB2PROCA.SALES" 5
xmlmgr (default): xpathexpr:////AuthorizationResource 4
```

```
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Extracting
resources using "XPath"    3
xmlfirewall (NSS_XML_Firewall): Policy(NSS_XML_Firewall_AAA_Policy): Extracting
identity using "http-basic-auth"     2
.......................
xmlfirewall (NSS_XML_Firewall): Finished parsing: http://10.1.2.101:2064/
xmlfirewall (NSS_XML_Firewall): rule (NSS_XML_Firewall_Policy_rule_0): selected
via match 'NSS_XML_Firewall_MatchRule' from processing policy
'NSS_XML_Firewall_Policy'
Matching (NSS_XML_Firewall_MatchRule): Match: Received URL [/] matches rule '*'
xmlfirewall (NSS_XML_Firewall): New transaction(conn use=1): POST
http://10.1.2.101:2064/ from 10.1.100.223 1
```

In this example, the numbers correspond to the following information:

1. A POST request has been received by the NSS_XML_Firewall at Port 2064 from 10.1.100.223. `NSS_XML_Firewall_Policy_rule_0` is selected because the URL matched the `NSS_XML_Firewall_MatchRule`.

2. The identity requiring authentication was extracted.

3. The resource requiring authorization was extracted using an XPath method.

4. The XPath expression is `/AuthorizationResource`.

5. The resource requiring authorization (through the AAA Policy) to the user ID of `CS03` is `EZB.DB2.DB2PROCA.SALES`.

6. The authentication method (through the AAA Policy) is with the NSS server (`zosnss`).

7. The authentication is to be processed by the NSS Client with a symbolic name of `NSSXML1`.

8. The authentication of user ID of CS03 with its password succeeded at the NSS server.

9. The authorization process starts with the NSS server.

10. The authorization is for READ access (`r`).

11. The call for READ authorization of CS03 to EZB.DB2.DB2PROCA.SALES is sent to the NSS server.

12. The NSS server responds positively to the authorization request.

13. The AAA action of the NSS_XML_Firewall_Policy_rule_0 executed successfully.

14. The results action of the NSS_XML_Firewall_Policy_rule_0 executed successfully.

You can now view messages in the NSSD log at the z/OS system. Messages are recorded in `nssd.log`, as shown in Example 10-48. These messages are recorded with the latest messages at the bottom. Therefore, begin the analysis at the top of the message series.

Example 10-48 NSS Log messages for the XML authentication and authorization

```
NSSD: DBGO031I NSS_VERBOSE UseridCheck(CS03,******) passed 1

NSSD: DBGO016I NSS_SAF_ACCESS_INFO Clientname (NSSXML1                ) requested
authentication for SAF ID (cs03    ) - user authenticated: (yes) 2

NSSD: DBGO032I NSS_VERBOSE ProfileCheck(CS03    ,EZB.DB2.DB2PROCA.SALES) rc 0 (A
AUTH (LEVEL CHECK)) racfRC 0 racfRsn 0 3

NSSD: DBGO017I NSS_SAF_ACCESS_INFO Clientname (NSSXML1                ) requested
access level (0x02) authorization for SAF ID (cs03    ) to SAF CLASS (SERVAUTH)
resource (EZB.DB2.DB2PROCA.SALES) - user authorized: (yes)   4
```

In this example, the numbers correspond to the following information:

1. NSS sends a request for authentication of the user ID of CS03 to the RACF database.
2. NSS sends a response to the NSS Client (NSSXML1) indicating that the user CS03 is authenticated.
3. NSS submits a request to RACF to validate the authorization of CS03 to the resource named EZB.DB2.DB2PROCA.SALES. RACF successfully validates the request with a return code and a reason code of 0.
4. The response sent to the NSS Client indicates that CS03 is authorized to access the requested resource.

Important: After you have concluded your testing, return to all configurations for NSS and the middleware appliance, and either disable or minimize tracing and logging.

10.4 Additional information

The testing that we described in this chapter confirms that the configuration of the NSS Server with an NSS Client that communicates with the XMLAppliance Discipline is correct. You need a good plan to build this environment. It imperative that someone devote time to designing the naming conventions and the topology of the solution, and that the z/OS Communications Systems Programmer work closely with the DataPower appliance specialist to build and test this environment. In addition, the application programmers who code the Web Services XML applications and requirements must work closely with the DataPower appliance specialist.

See the following resources:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775, for additional information regarding NSS configuration.
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787, for additional information about authorization.
- *z/OS Cryptographic Services ICSF Overview*, SA22-7519, for general information about ICSF.
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521, for information about working with hardware cryptographic devices and ICSF-protected private keys.

All DataPower product documentation is available from a link on the DataPower product support page:

`http://www.ibm.com/software/integration/datapower/support/`

See the following series of IBM Redbooks publications about DataPower appliances:

- *IBM WebSphere DataPower SOA Appliances Part I: Overview and Getting Started*, REDP-4327
- *IBM WebSphere DataPower SOA Appliances Part II: Authentication and Authorization*, REDP-4364
- *IBM WebSphere DataPower SOA Appliances Part III: XML Security Guide*, REDP-4365
- *IBM WebSphere DataPower SOA Appliances Part IV: Management and Governance*, REDP-4366
- *DataPower Architectural Design Patterns: Integrating and Securing Services Across Domains*, SG24-7620

10.5 NSS configuration worksheet for an NSS XMLAppliance client

Use the worksheet in this section to plan the definitions for an NSS XMLAppliance Client that will communicate with an NSS Server. You might find a helpful way is also to produce a diagram for your scenario, much like the one we used in Figure 10-6 on page 439.

1. What is the MVS System ID of the node that is to run the Network Security Services Daemon?

2. What is the superuser user ID associated with the OMVS segment under which the NSS Daemon is to run?

3. With which TCP/IP stack (or stacks) does the NSS daemon establish affinity?

4. What is the address or the DNS name that the NSS XMLAppliance client will use to connect to the NSS server?

5. What “symbolic client name” or ClientID do you want to assign to each NSS XMLAppliance client that is to request authentication of the NSS server?

6. What RACF “user ID” do you want to assign to each NSS ClientID that is to request authentication of the NSS server?

> **Rule:** Multiple NSS clients can use a single user ID. However, each NSS client must have a unique symbolic client name (for example ClientID).
>
> **Guideline:** Because SAF user IDs are used to authorize a client to the NSS services, avoid sharing a single user ID across NSS disciplines.

7. What type of authentication should our NSS client user ID present: a password or a passticket?
   - Password:
   - Passticket (Yes or No?):
8. What is the name of the key ring that NSS uses to manage the necessary certificates and private keys for NSS IPSec clients, and for XMLAppliance private key services and certificate services clients? Although the scenario to support NSS XMLAppliance SAF services clients does not require this key ring, the configuration of the NSS server includes this key ring name.

9. What are the labels of the certificates that need to reside on this NSS key ring? You need to document the labels of the certificates for the IPSec discipline and for the private key and certificate services of the XMLAppliance discipline.

   _____

   _____

   _____

10. What is the name of the key ring that NSS uses to manage the certificates for secure communications between itself and its NSS clients?

11. What are the labels of the certificates on the NSS node that are used for the TLS/SSL secure connection between the NSS server and the NSS XMLAppliance client?
    - Signing Certificate:
    - Server Certificate:

12. Do you plan to use x.500 distinguished name filtering in RACF for certificate mapping? If so you will need to activate RACF digital certificate mapping.

13. If the NSS server has implemented IP filtering and IPSec, what are the IP addresses of the NSS server and the NSS client for which you might have to establish IP filters on the NSS server node?
    - Client:
    - Server:

14. What are the TSO user IDs of the IPSec administrators who are authorized to manage NSS and its clients?

15. What are the RACF user IDs of the Web Services Requesters that might need authentication and authorization at z/OS RACF?

16. What are the SERVAUTH resources to which the Web Services Requester user IDs require authorization, and what type of access do they need (Read, Update, Control)?

    SERVAUTH class of _____ (access _____)

11

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.

Tip: z/OS Communications Server provides NAT traversal support only for IPv4 traffic.

We discuss the following topics in this chapter.

Section	Topics
11.1, “Network Address Translation (NAT)” on page 536	Basic functionality of NAT and Network Address Port Translation (NAPT)
11.2, “IPSec and NAT incompatibilities” on page 538	Brief description of the incompatibilities between IPSec and NAT
11.3, “NAPT traversal support for integrated IPSec/VPN” on page 539	The solution, testing environment, configuration examples, and problem determination suggestions

11.1 Network Address Translation (NAT)

NAT is a method defined in RFC 3022. NAT, as the name suggests, is a way of translating (or changing) a host's source IP address into a different IP address on an outbound packet. 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 used to represent a 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)

11.1.1 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 11-1 shows static NAT.

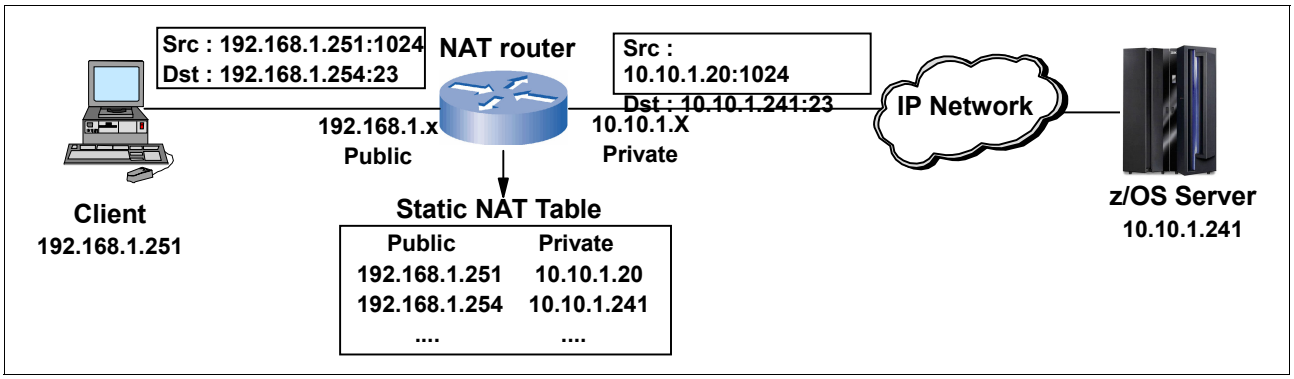


Figure 11-1 Static NAT

11.1.2 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.

Tip: NAPT translation is by far the most commonly found Network Address Translation. 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 11-2, 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.

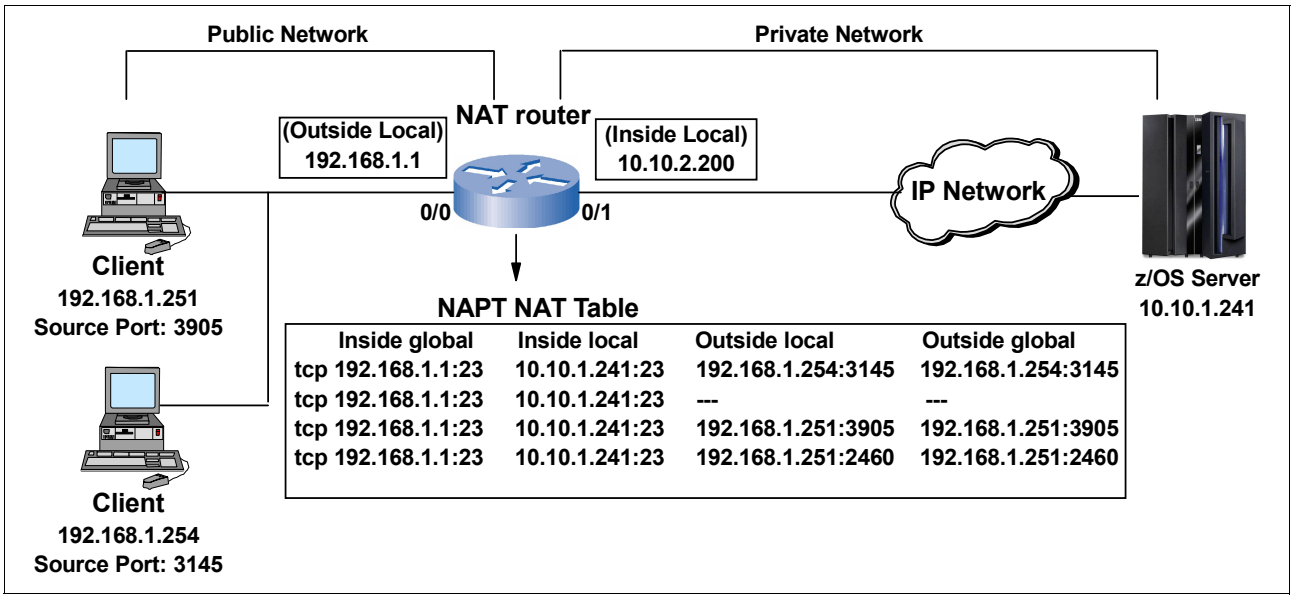


Figure 11-2 NAPT configuration example

Example 11-1 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 11-1 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
```

11.2 IPSec and NAT incompatibilities

This section discusses the incompatibilities between IPSec and NAT functions.

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, 3948, and 5996 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.
- RFC 5996, Internet Key Exchange Version 2 (IKEv2) Protocol, defines the IKEv2 protocols. Section 2.23 NAT Traversal specifies how to detect when IKEv2 peers are traversing one or more NATs. It also describes how UDP encapsulation of IKE and ESP packets is used to allow IPSec to interoperate with NAT when using IKEv2.

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.

UDP encapsulation: UDP encapsulation is *not* encapsulating a UDP packet. UDP encapsulation is inserting a UDP header between the IP header and the ESP or IKE header. The payload data can have a TCP, UDP, or other transport header.

11.3 NAPT traversal support for integrated IPSec/VPN

z/OS Communications Server extends IP security support to protect traffic that traverses a NAPT device. This 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), RFC 3948 (UDP encapsulation of IPSec ESP packets), and RFC 5996 (IKEv2 section 2.23) 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 between IPv4 addresses.
- If a UDP-encapsulated mode SA is negotiated using the IKEv2 protocol, then the SA and the connections over the SA are distributed to only a V1R12 or later target.
- In an SWSA environment, SWSA takeover functions are not available for the SA.

11.3.1 Enabling NAPT traversal support for IPSec

The objective was to successfully establish an IPSec dynamic tunnel from a Windows client, through a Router NAPT device to a TELNET server running on the z/OS mainframe. To test and verify the support, we made the following configuration changes:

- IBM Configuration Assistant: Enabled NAT traversal support for TCPIPA
- Router: Configured NAT and Port Address Translation (PAT) for port 23
- Windows XP: Installed and configured IPSec

Figure 11-3 represents our test environment.

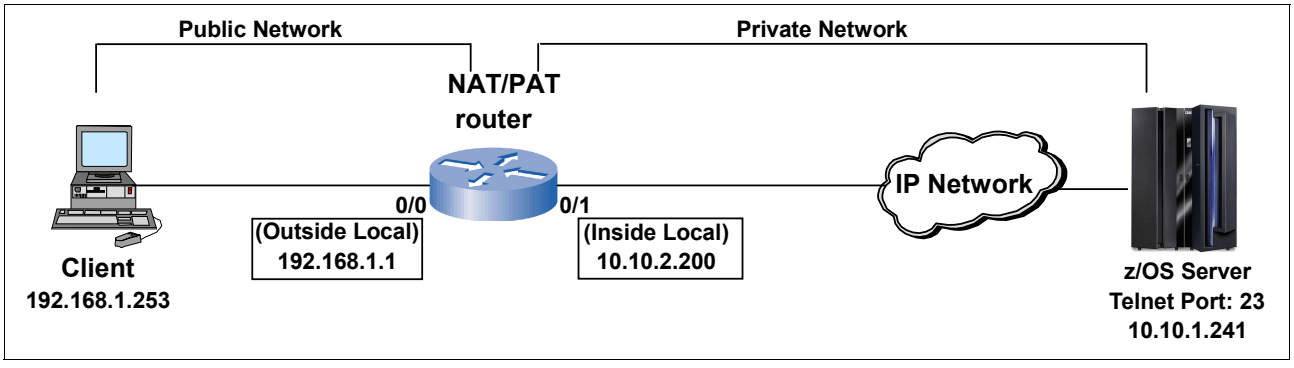


Figure 11-3 NAPT traversal diagram

Next, we explain the configuration changes we made to our test environment to successfully establish the IPSec dynamic tunnel using the NAPT traversal device.

IBM Configuration Assistant changes

In this section, we demonstrate the additional changes that are required on the IBM Configuration Assistant graphical user interface (GUI) to enable the NAPT support.

> **Important:** Before making these changes, refer to and complete the IPSec for Windows XP implementation using the examples supplied in Appendix C, “Configuring IPSec between z/OS and Windows” on page 859 as a guide.

To make changes in the IBM Configuration Assistant, complete the following steps:

1. From the main IBM Configuration Assistant panel, select **Stack - TCPIPA i**n the Navigation tree as shown in Figure 11-4. Click the Stack Settings tab.

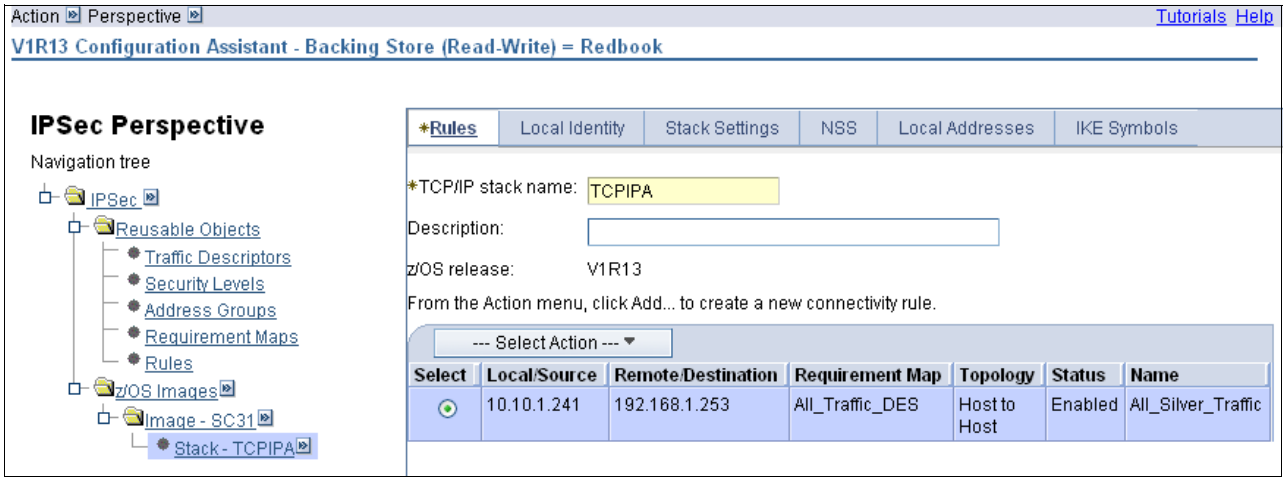


Figure 11-4 TCP/IP stack settings

2. On the Stack Settings tab, select to allow NAT, and to enable filter logging without including “implicitly deny” events, as shown in Figure 11-5.

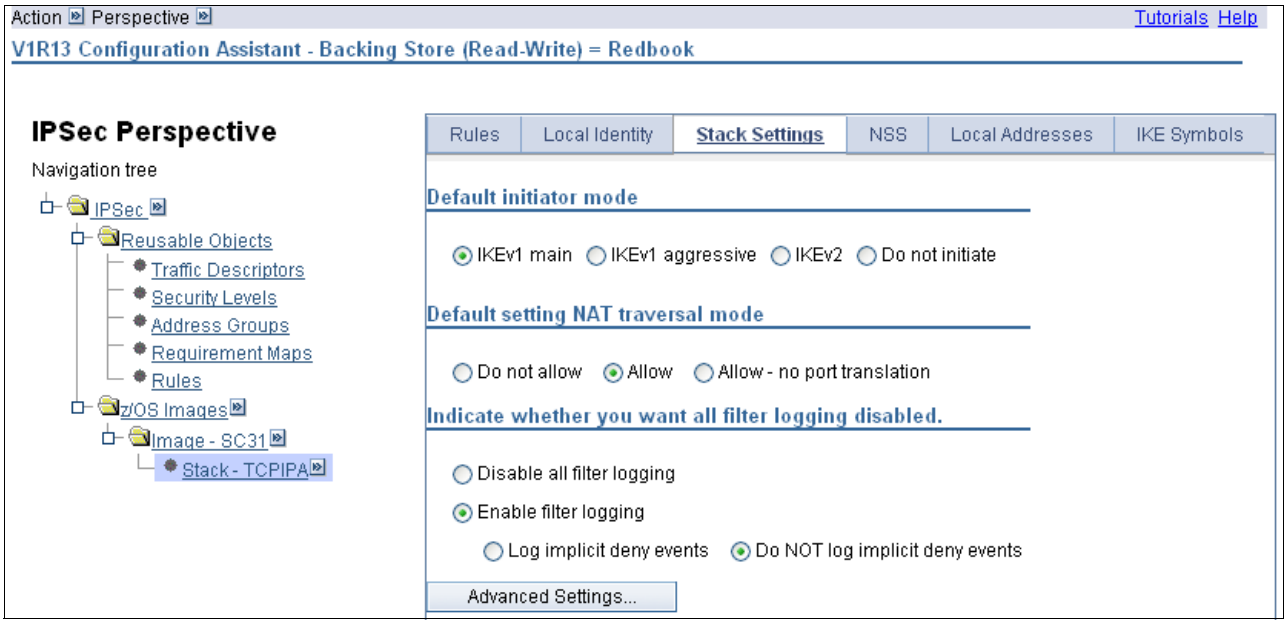


Figure 11-5 IPSec: Stack level settings

3. Click **Advanced Settings** to enable NAT keepalive messages, as shown in Figure 11-6

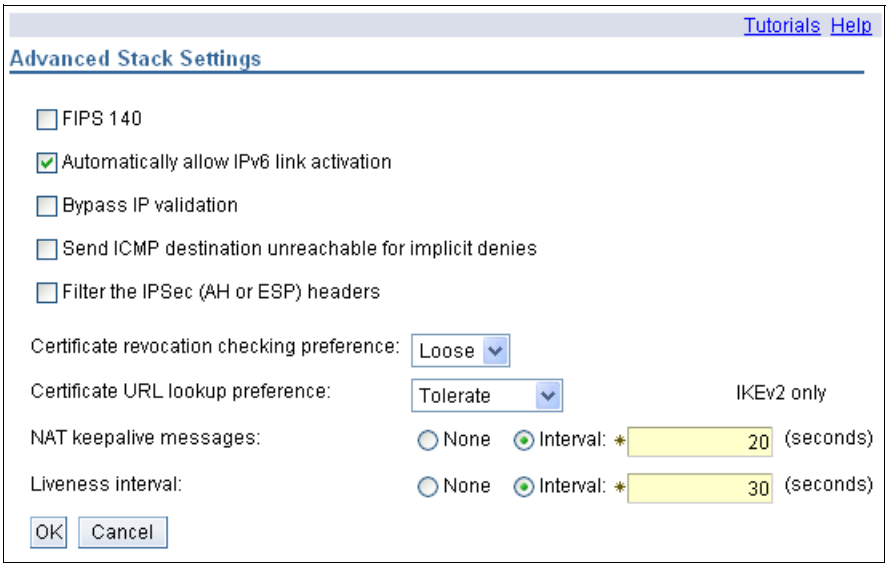


Figure 11-6 NAT keepalive settings

4. Click the Rules tab as shown in Figure 11-7. Select or highlight the rule for 10.10.1.241 to 192.168.1 253, and then click **Select Action** and click **Modify**.

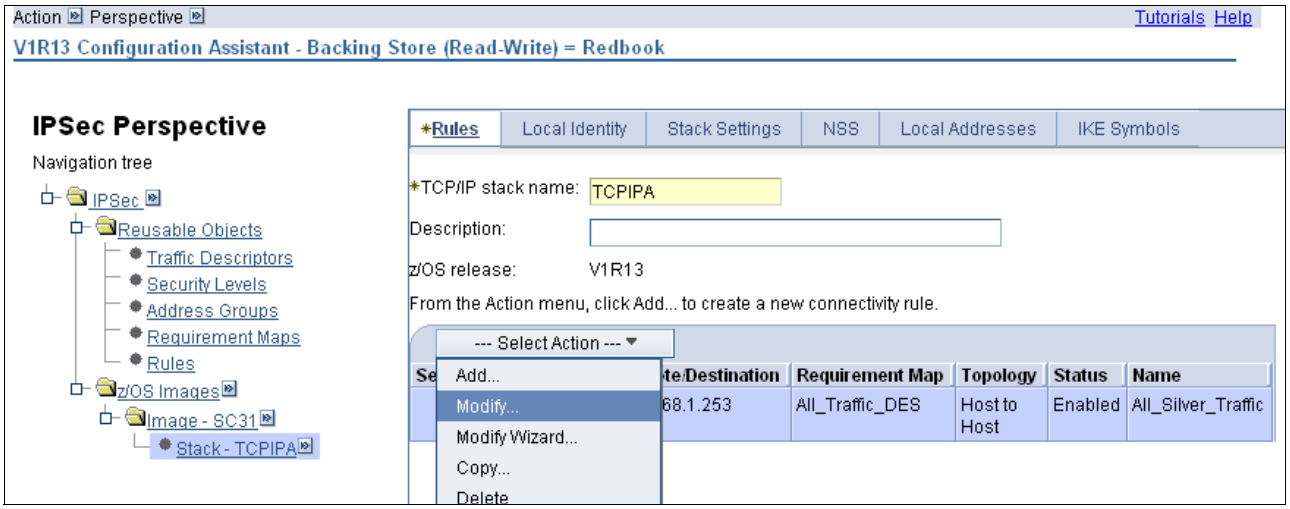


Figure 11-7 TCP/IP Connectivity Rules

5. In the Modify Connectivity Rule panel, click the Additional Settings tab. Click **Advanced** (Figure 11-8).

Figure 11-8 Connectivity Rule: Additional Settings

6. Click the Key Exchange tab. Next to **Network Address Translation (NAT) traversal** select **Allow** as shown in Figure 11-9.

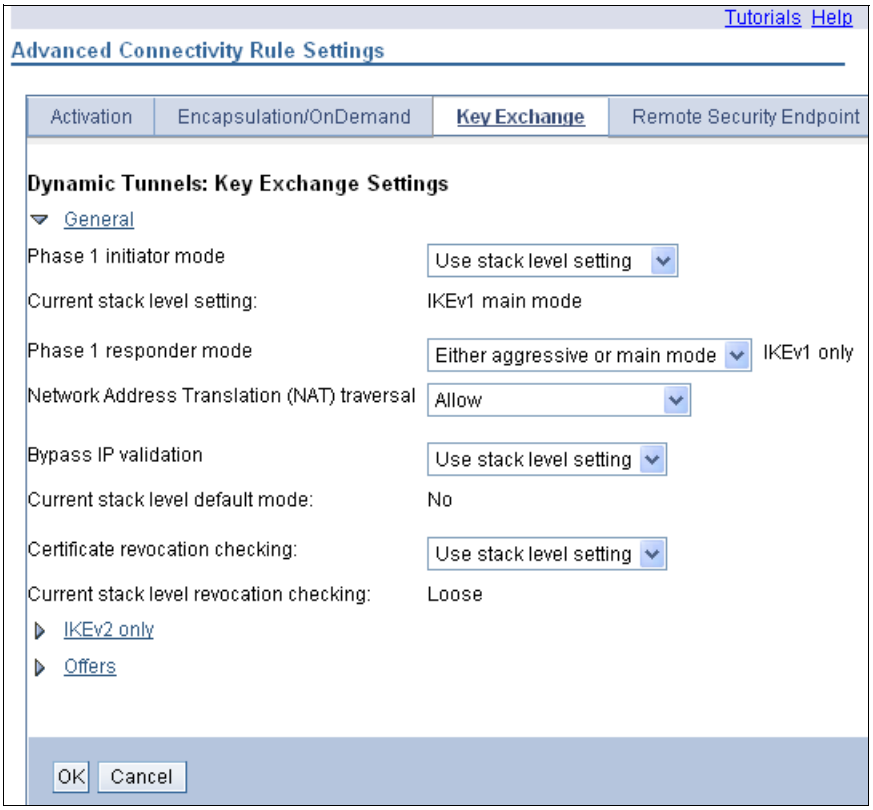


Figure 11-9 IPSec: Dynamic Tunnels: Key Exchange Settings

7. Click **OK** until the IPSec Navigation Tree is displayed. Click the small symbol to the right of **Stack - TCPIPA** and select **Install Configuration Files**, (Figure 11-10)

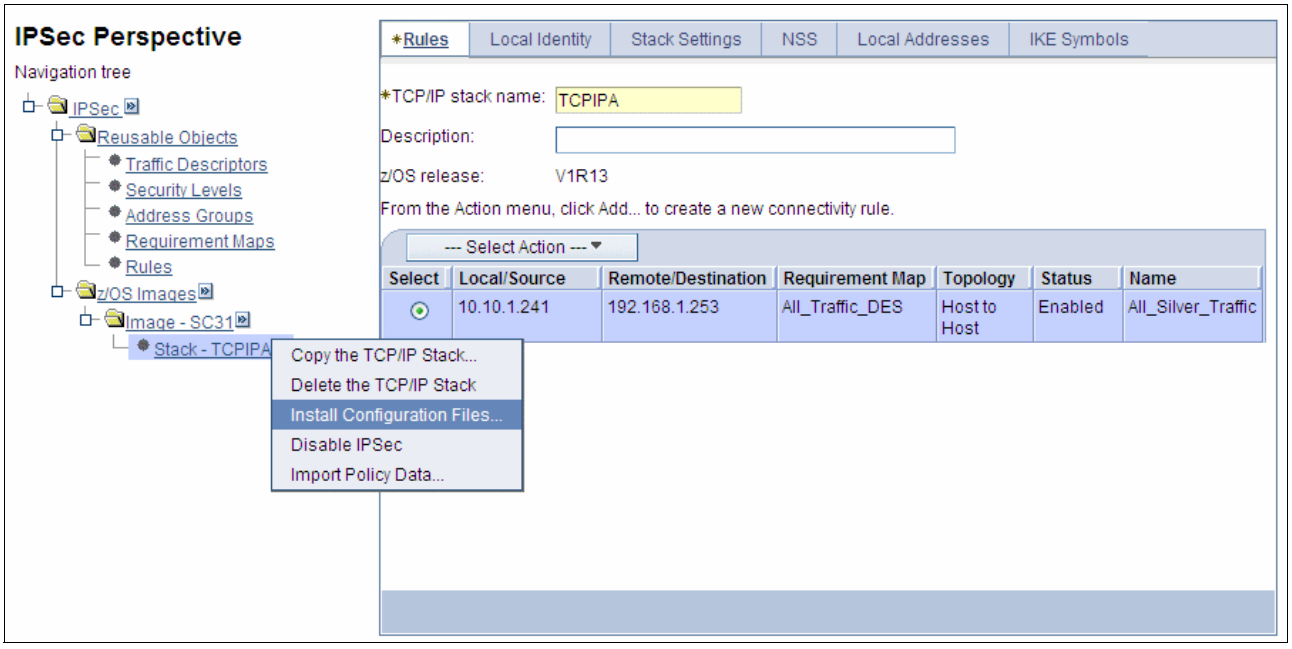


Figure 11-10 Install Configuration Files selection

8. Transfer the new IPSec rules to the PAGENT server on z/OS by selecting the IP Security Policy clicking the **Select** radio button, then click **Select Action** → **Install** (Figure 11-11).

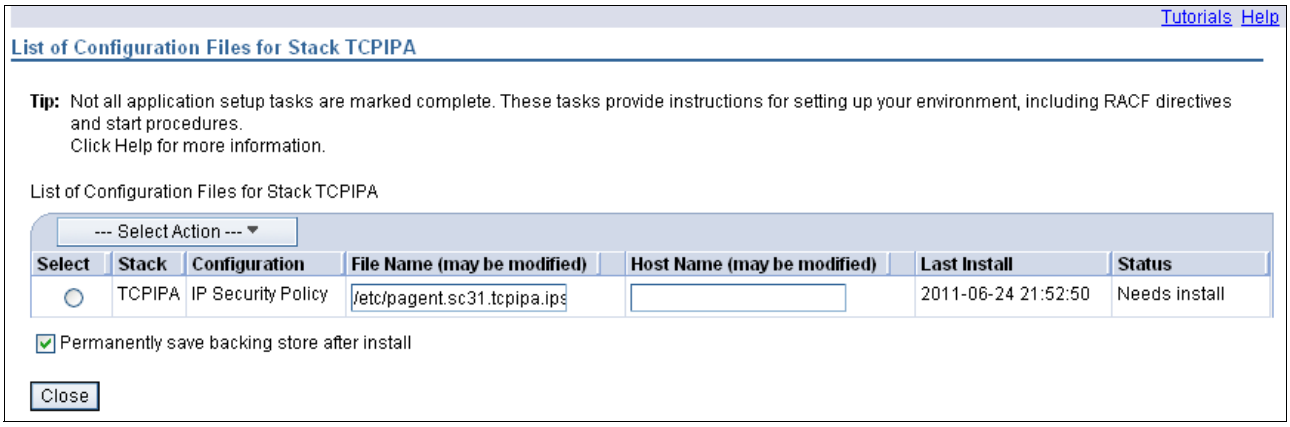


Figure 11-11 Configuration Files Installation

Router configuration

For our NAPT device, we used a 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 11-2.

Example 11-2 Router 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 859. However, because our IPSec tunnel traversed a NAT device, we also had to enable NAT traversal support for Windows. Therefore, we discuss it here.

Important: If these actions are not executed, the IPSec NAT traversal tunnel will not establish successfully.

We had Windows XP Service Pack 2 installed, and because a change in the default behavior for Windows IPSec NAT traversal, we had to make a modification to the Windows Registry. Search for “NAT Traversal” on the Microsoft support website to determine whether your system requires similar changes. The details and description for our change can be found at the following location:

http://support.microsoft.com/kb/885407/en-us

We also changed the Destination address in “Set up a Windows IPSec policy for pre-shared key mode” on page 868 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 router device shown in Example 11-2 on page 544.

11.3.2 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 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 11-12) was the first indication that we managed to establish the IPSec tunnel.

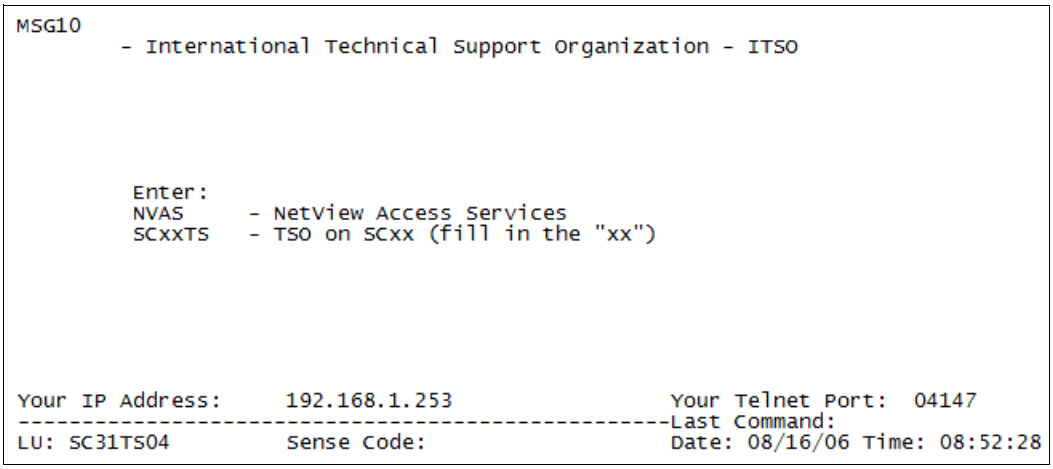


Figure 11-12 MSG10 on the Windows client

2. We logged on to the router and issued the **show ip nat translation** command. The output in 11.3, “NAPT traversal support for integrated IPSec/VPN” on page 539 confirmed two 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 11-3 Router 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#
```

c. 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 as shown in Example 11-4 on page 547.

Example 11-4 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
 ***
```

3. 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 11-5 and Example 11-6 on page 548, respectively.

Example 11-5 Output from the ipsec -p tcpipa -k display

```
Primary:  IKE tunnel      Function: Display           Format:   Detail
Source:   IKED            Scope:    Current           TotAvail: n/a

TunnelID:                     K1
Generation: 3
IKEVersion: 1.0
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 11-6 shows the output from the **ipsec -p tcpipa -y display** command.

Example 11-6 Output from the ipsec -p tcpipa -y display

```
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 1

TunnelID:                     Y2
Generation: 3
IKEVersion: 1.0
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
```

These examples might change with each release. See the most recent version of *z/OS CS: IP System Administrator's Commands*, GC31-8782.

Debugging tools

We used the following commands and tools to gather debugging information, depending on the type of problem on which we worked:

- TCP/IP Netstat, Ping, Tracerte, and displays
- Omproute (OSPF) **`d tcpip,tcpipa,omp,ospf,`*`xxxxx`** displays
- Log files, MVS console log, and Syslogd
- 2600 Router, **`show ip nat translations`**, **`debug ip nat detailed`**

12

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.

We discuss the following topics in this chapter.

Section	Topic
12.1, “Conceptual overview of AT-TLS” on page 552	The role of AT-TLS in securing socket-based applications
12.2, “AT-TLS Implementation Example: REXX socket API” on page 557	How to implement AT-TLS for a general application A REXX API is used as an example
12.3, “Problem determination for AT-TLS” on page 580	AT-TLS problem determination techniques
12.4, “Additional information sources for AT-TLS” on page 581	AT-TLS application restrictions

12.1 Conceptual overview of AT-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.

12.1.1 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 composed 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 key rings and the associated advantages (user ID mapping, SITE certificates, and more).

Tip: 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 as discussed in Chapter 8, “IP Security” on page 245.

12.1.2 How AT-TLS works

AT-TLS support is policy-driven and is managed by a policy agent (PAGENT), as discussed in Chapter 4, “Policy agent” on page 103. 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 12-1 is followed.

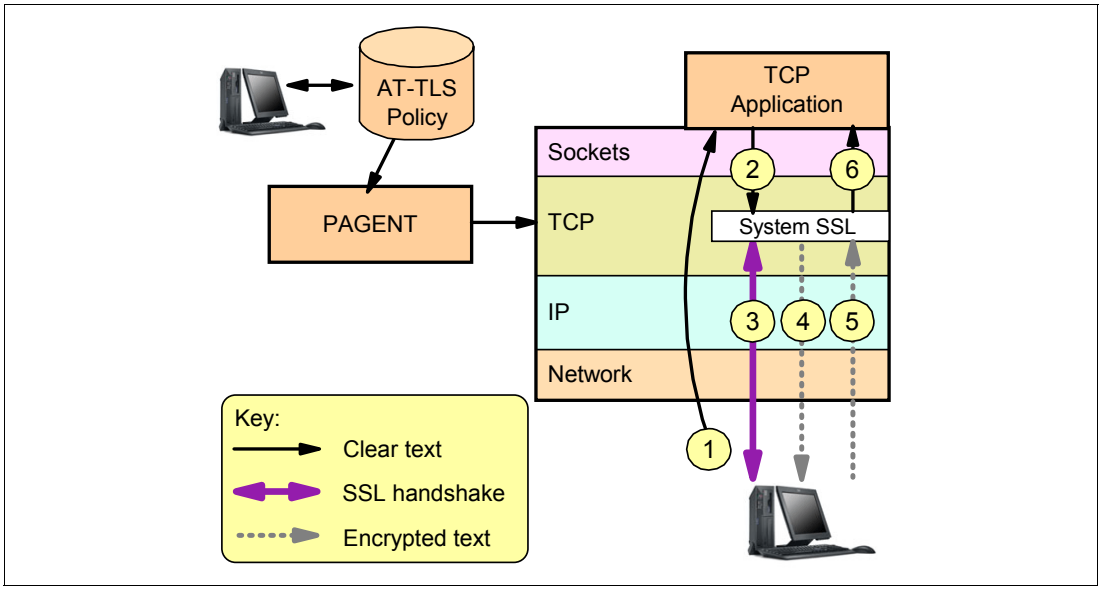


Figure 12-1 AT-TLS basic flow

This diagram illustrates the following flow:

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 through 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 554. 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.

12.1.3 How AT-TLS can be applied

Although AT-TLS can be used to provide security for the majority of applications transparently, certain 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.

Tip: Be careful to coordinate your use of AT-TLS with other application-specific encryption implementations. Otherwise, you could be 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. Certain applications need to be aware of when a secure connection is being used. Others might 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.

Tip: The AT-TLS capabilities for TN3270 and FTP server are 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 can negotiate the use of TLS in clear text 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 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.

See Chapter 17, “Secure File Transfer Protocol” on page 719 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 AT-TLS to manage secure connections. TLS managed by AT-TLS (TTLSPORT) supports more security functions than TLS managed by the TN3270 (SECUREPORT).

Tip: 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, 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

See Chapter 16, “Telnet security” on page 677 for complete details about implementing TLS and AT-TLS security for the TN3270 server.

Tip: 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 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.

Tip: Use of application-negotiated SSL/TLS (AUTH command) is preferred 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, avoid 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, 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 operates at a separate level from IP filtering and VPN policies. AT-TLS and VPN rules can function together, effectively independent of each other. However, their filtering should be coordinated.

12.2 AT-TLS Implementation Example: REXX socket API

This section shows how to implement general AT-TLS support using a REXX socket server application running on SC30 using stack TCPIPA, connecting to a REXX socket client application running on SC31 using stack TCPIPB. Showing the source of the client and server REXX programs and their JCL is beyond the scope of this book. 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 4, “Policy agent” on page 103.

Figure 12-2 shows the relationship of the policy agent configuration files to one another.

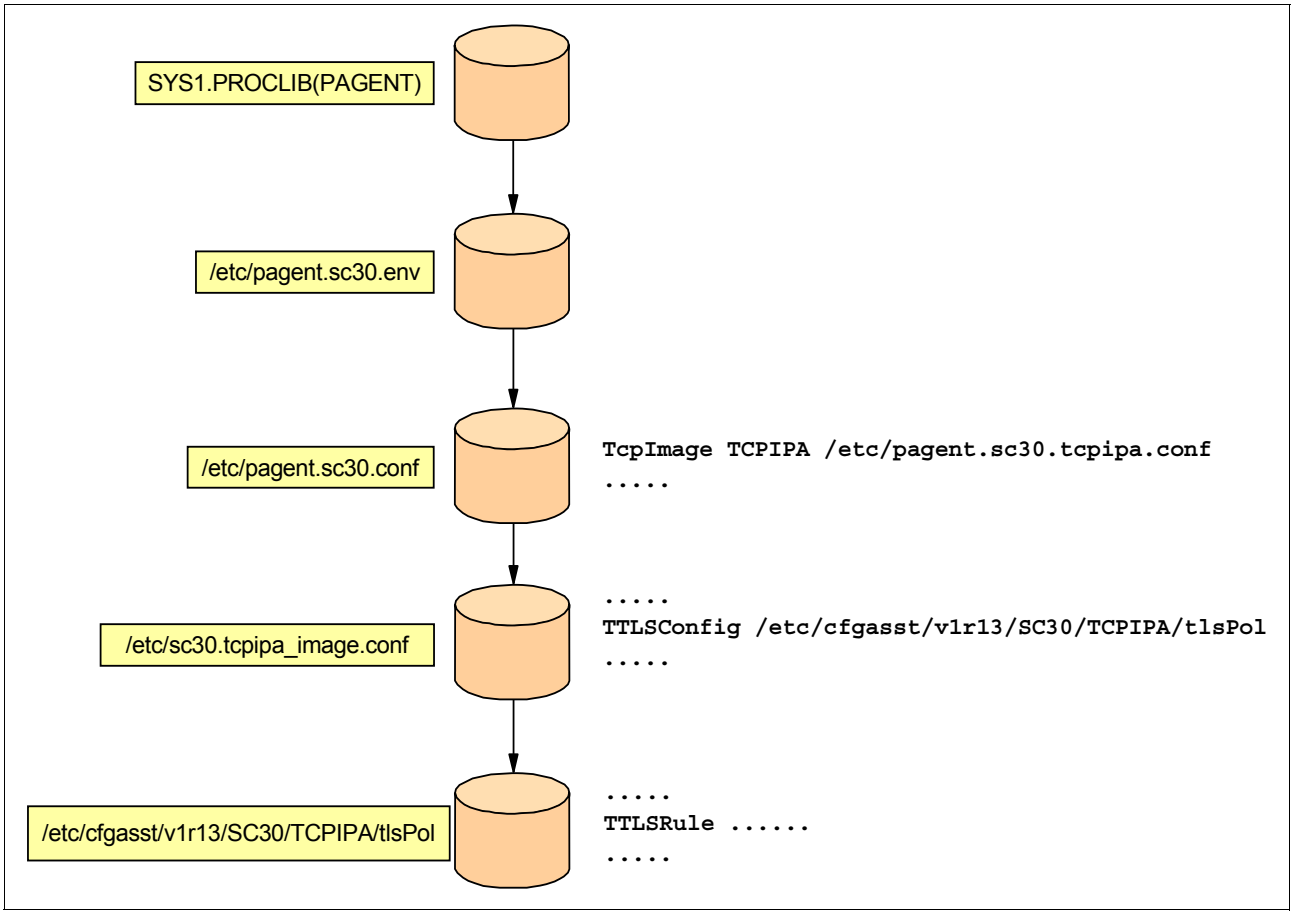


Figure 12-2 AT-TLS PAGENT configuration file relationship

As Figure 12-2 on page 557 shows, we set up our AT-TLS related policy agent configurations files by coding TcpImage statements pointing to a `/etc/sc30.tcpipa_image.conf` file for each TCP/IP stack. Each one of these stack TTLSConfig files contains a TTLSConfig statement that points to `/etc/cfgasst/v1r13/SC30/TCPIPA/tlsPol`. 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

12.2.1 Description of REXX AT-TLS support

We set up a REXX socket client and server application on our two z/OS systems, SC30 and SC31, as shown in Figure 12-3, 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.

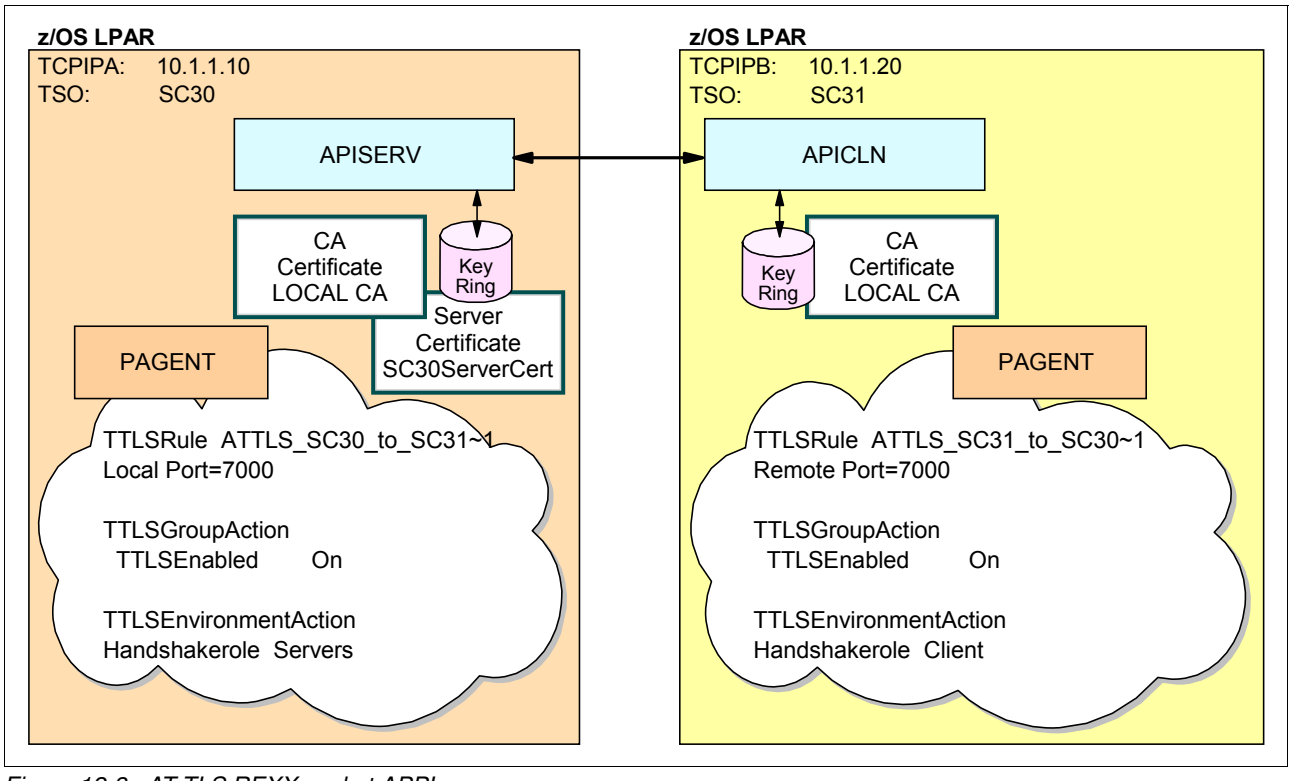


Figure 12-3 AT-TLS REXX socket APPLs

12.2.2 Configuration of REXX AT-TLS support

Setting up AT-TLS for our REXX client/server socket applications involved the following activities:

Step 1: Configuring the server policies
Step 2: Configuring the client policies
Step 3: Save the policy to z/OS
Step 4: Modify the policy agent configuration file
Step 5: Defining the digital certificates and key rings
Step 6: Enabling CSFSERV resources
Step 7: Controlling access during the window period
Step 8: Enabling AT-TLS in the TCP/IP profile

Step 1: Configuring the server policies

We use the z/OSMF Configuration Assistant 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 and create a backing store
2. Add a z/OS image, TCP/IP stack, and enable AT-TLS for SC30
3. Add a traffic descriptor object for REXX server
4. Add a requirement map object for REXX server
5. Add a connectivity rule for SC30

Open the Configuration Assistant and create a backing store

If you start working on the existing backing store, skip this section.

Complete the following steps:

1. Open the z/OSMF Configuration Assistant. Click **Action** → **Open** → **Create a New Backing Store**.
2. Type in a file name for the backing store file, as shown in Figure 12-4 and click **Open**. This will enable you to use an **Action** → **Save** operation.

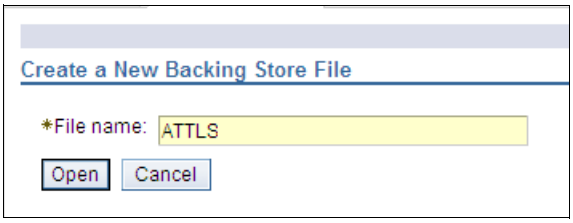


Figure 12-4 Create a New Backing Store

Add a z/OS image, TCP/IP stack, and enable AT-TLS for SC30

To add a z/OS image for the REXX server, complete the following steps:

1. In the Main Perspective panel, click **Add a New z/OS Image**.
2. In the z/OS image information panel, type a name and description for the z/OS image. In this example, we use `SC30`. Click **OK**.
3. When you are prompted to configure the TCP/IP stack, click **Yes**. We use `TCPIPA` as the stack name for this example. Enter the required information, then click **OK**.

4. In the Main Perspective panel (Figure 12-5), click the z/OS image that you have created. Select **AT-TLS** and click **Select Action** →**Enable**. The AT-TLS status changes from `Disabled` to `Incomplete`. Click **Select Action** →**Configure** to proceed.

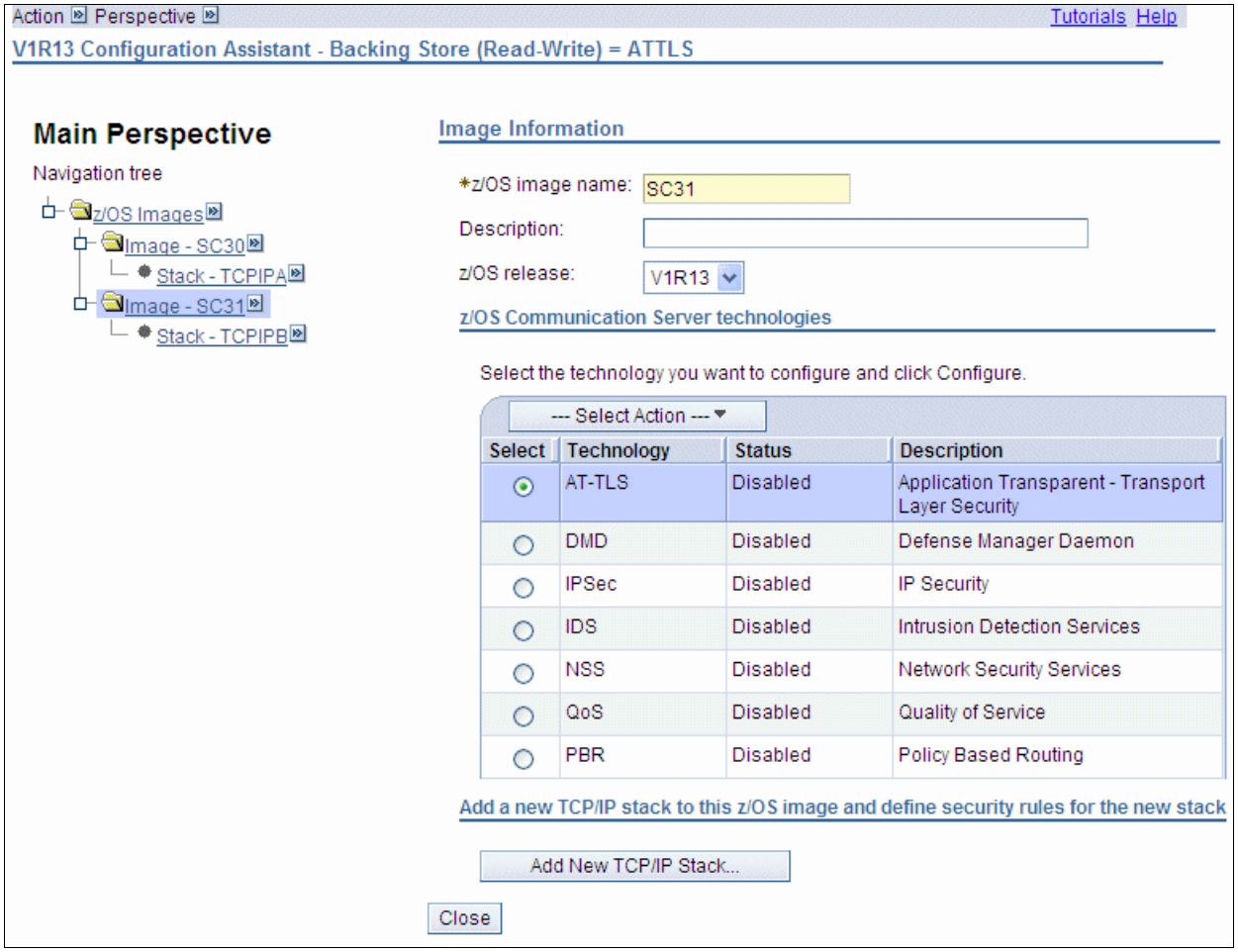


Figure 12-5 Enabling AT-TLS on z/OS panel

Add a traffic descriptor object for REXX server

Complete the following steps:

1. In the AT-TLS Perspective panel, click **Traffic Descriptors** under the “Work with reusable objects” section. In the next panel, click **Select Action** →**Add**.
2. In the New Traffic Descriptor panel (Figure 12-6 on page 561), enter a name (ours is `REXXServer`) and description for the new traffic descriptor, and click **Select Action** →**Add**.

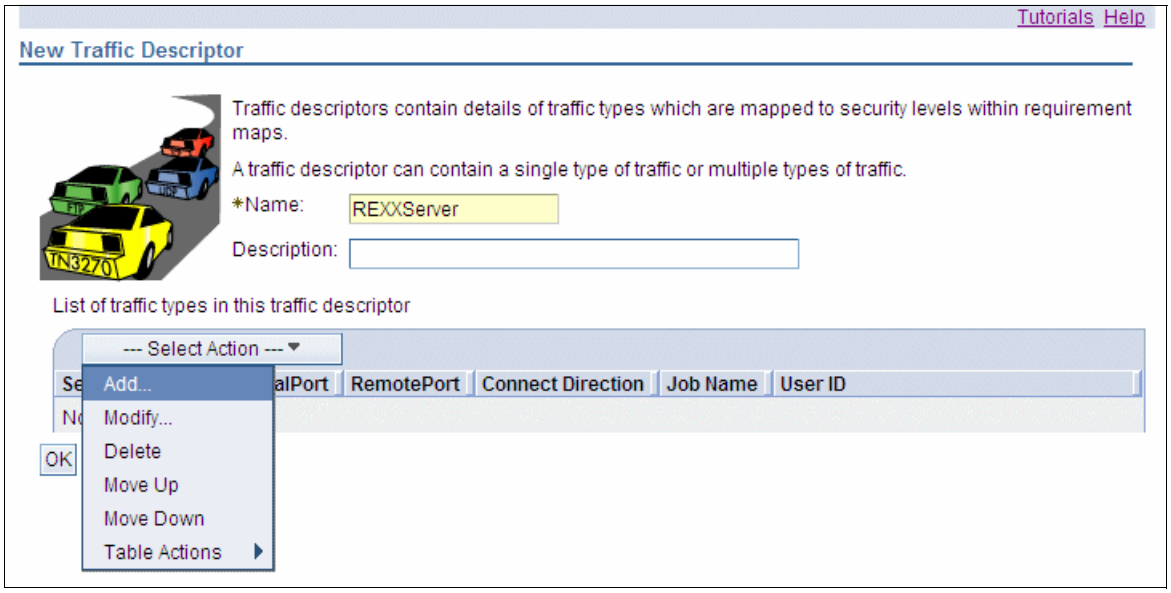


Figure 12-6 New traffic descriptor panel

3. In the New Traffic Type TCP panel (Figure 12-7) Details tab, enter configuration parameters for the traffic descriptor. In this example, we use the parameters to match REXX API server settings, which are local port set to 7000, TCP connect direction set to **Inbound only,** and AT-TLS handshake role set as **Server**.

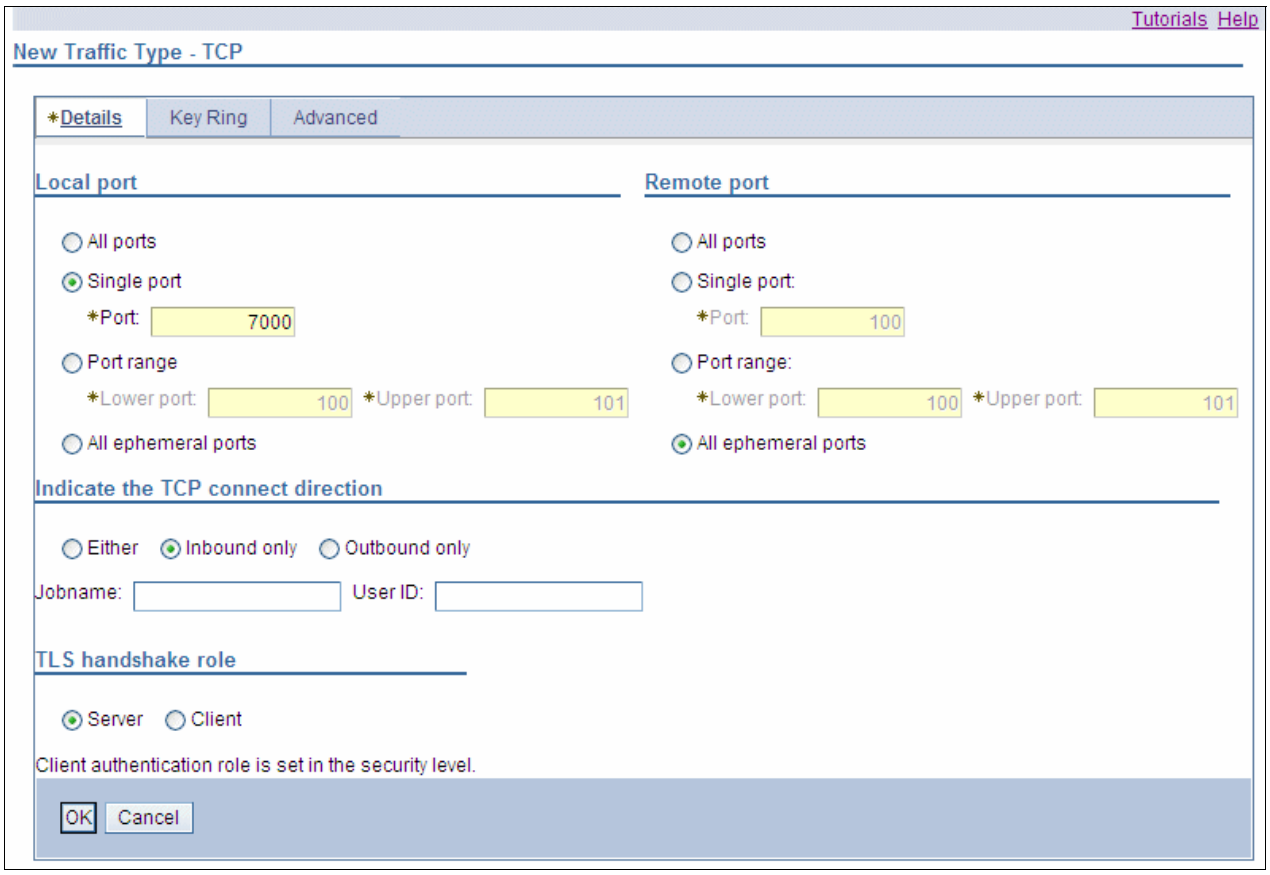


Figure 12-7 New Traffic Type - TCP (Details tab)

4. Click the Key Ring tab. For our scenario, we select the key ring database defined for z/OS image. Click **OK**.
5. In the New Traffic Descriptor panel that opens, click **OK**.

Add a requirement map object for REXX server

Complete the following steps:

1. In the AT-TLS Perspective, click **Requirement Maps**.
2. In the list of all defined requirement map objects, click **Select Action** →**Add**.
3. In the New Requirement Map panel (Figure 12-8), enter a name (ours is `SvrReqMap`) and a description for the requirement map. Select the traffic descriptor that you want to configure in this requirement map object (ours is `REXXServer`). Select the correct AT-TLS - Security Level for the traffic descriptor that you added by clicking the list. In this example, we defined our server traffic descriptor with the supplied `AT-TLS GOLD` security level. Click **OK** to continue.

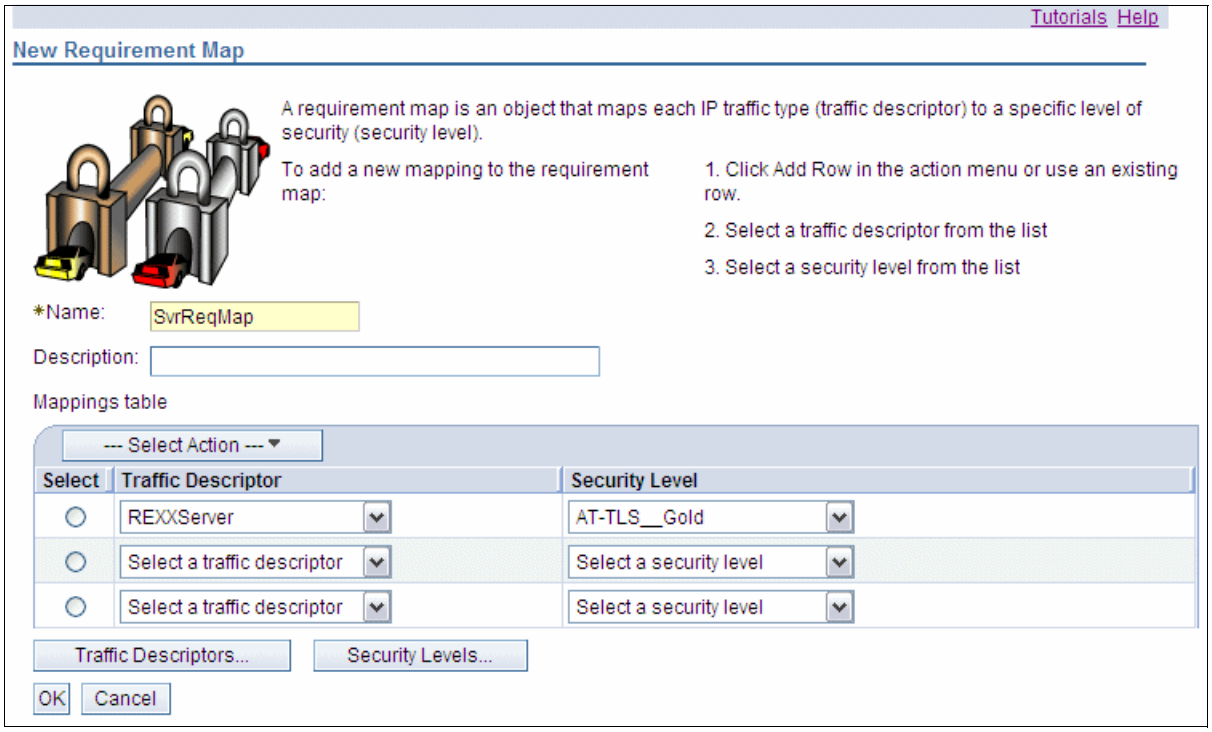


Figure 12-8 New Requirement Map panel

4. In the AT-TLS Perspective, select the image name (ours is `SC30`). Click the Settings tab. Under Default AT-TLS key ring database, we selected **Simple name** and typed the name of the key ring that we created in RACF, which is `ATTLS_keyring`, as shown in Figure 12-9 on page 563. We also selected all the trace levels. Click **OK**.

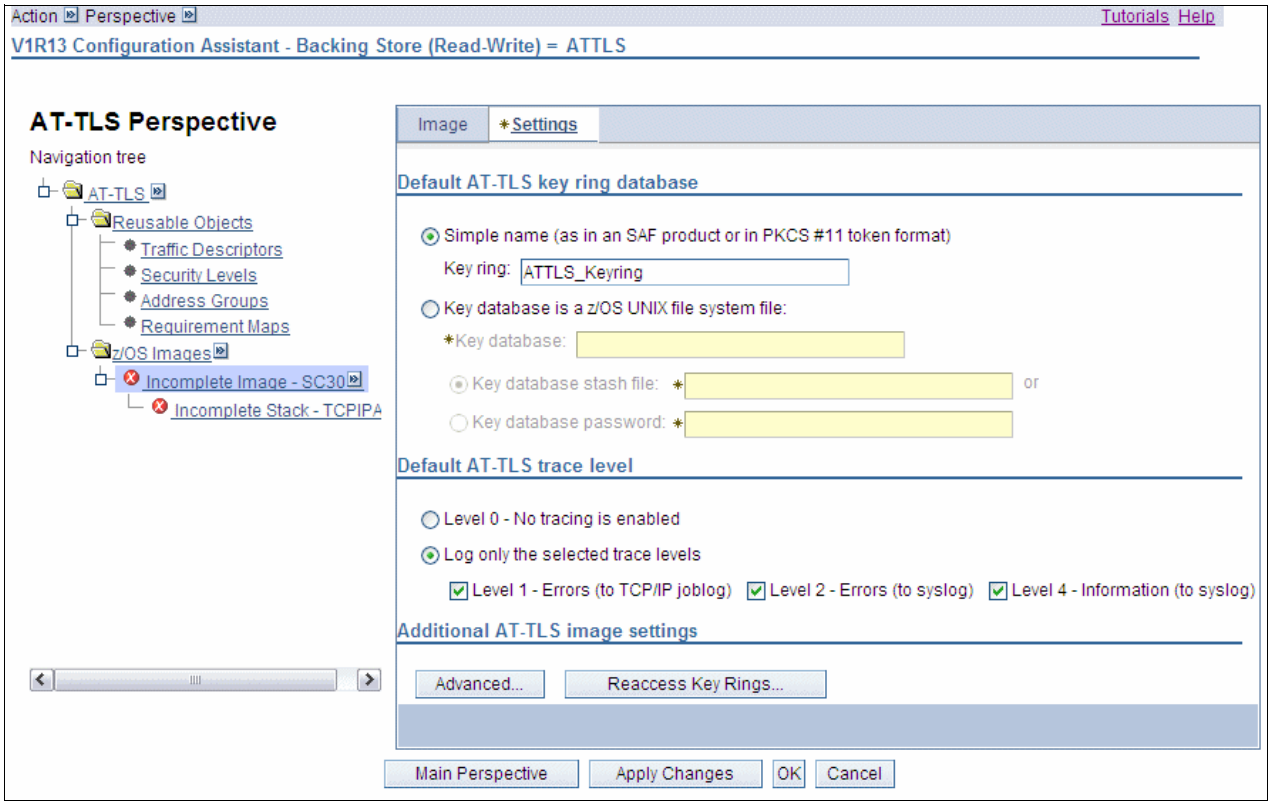


Figure 12-9 z/OS Image: AT-TLS Image Level Settings

Add a connectivity rule for SC30

Complete the following steps:

1. In the AT-TLS Perspective panel (Figure 12-10 on page 564), click the TCP/IP stack name that you have created (ours is `TCPIPA`), and then click **Enable AT-TLS** to set up a new connectivity rule. Click **Select Action** →**Add**. In the New Connectivity Rule: Welcome panel, click **OK**.

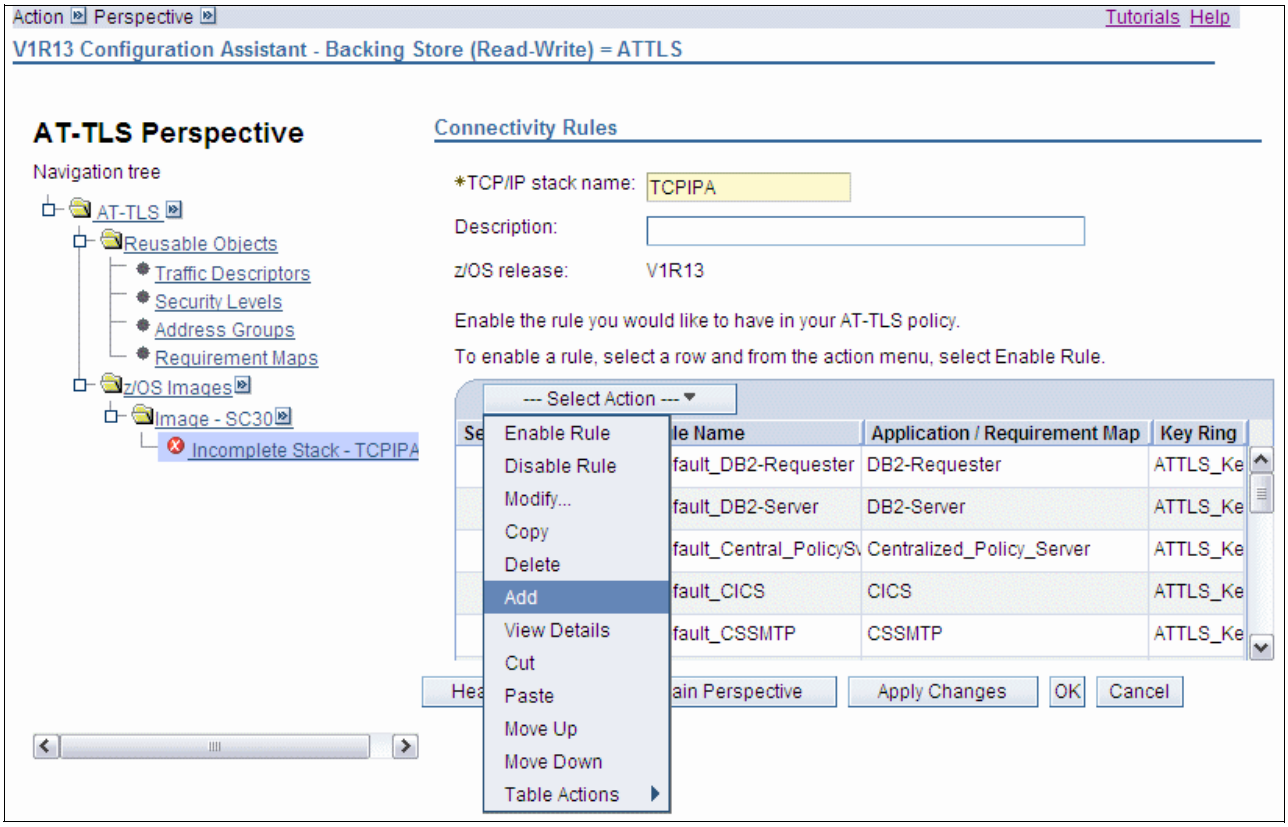


Figure 12-10 AT-TLS Perspective: adding new connectivity rule

2. In the new connectivity rule wizard, click **Next**. The first step is to specify Data Endpoints. In the New Connectivity Rule: Data Endpoints panel, type the endpoints data and a rule name as shown in Figure 12-11 on page 565. Click **Next**.

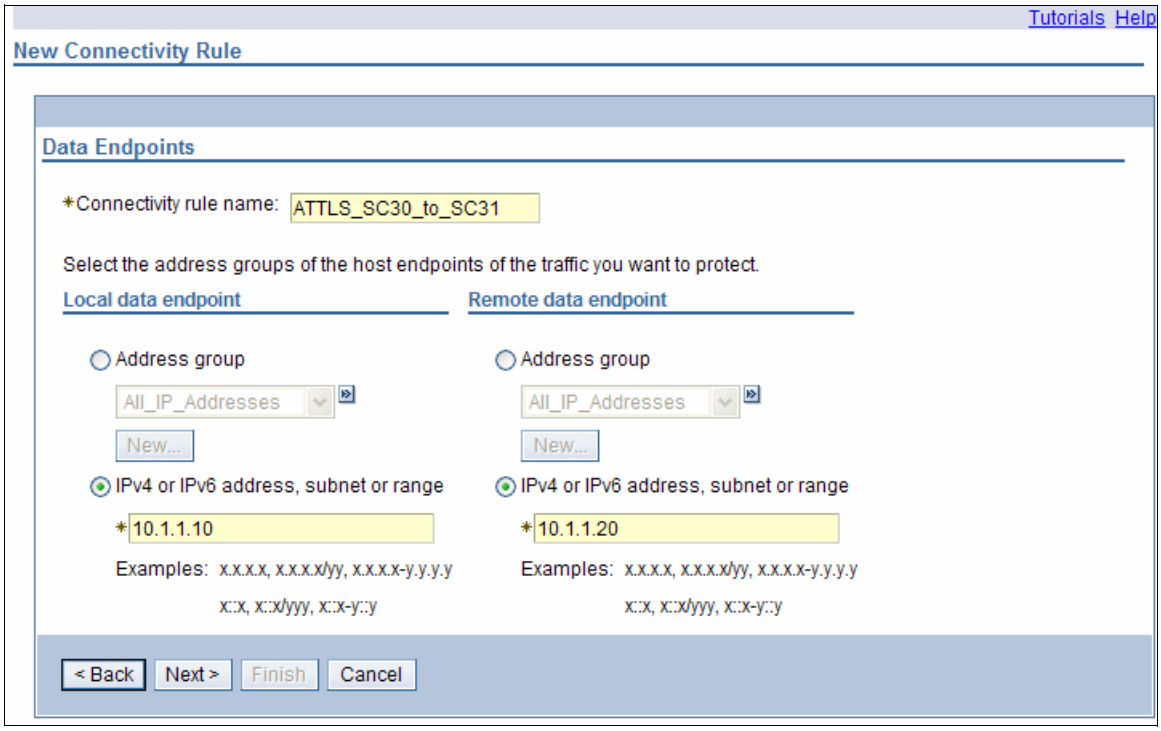


Figure 12-11 Connectivity Rule: Data Endpoints panel

3. In the New Connectivity Rule: Select Requirement Map panel, select existing requirement map and use the server requirement map (ours is `SvrReqMap`) that you created in the previous step. Click **Next** when you are done, and then click **Finish** on the last panel.
4. In the AT-TLS Perspective, click **Apply Changes**.

The resultant policy file for SC30 is listed in Example 12-1. An explanation of these policies, along with the changes, follows.

Example 12-1 Server AT-TLS policy for TCPIPA on SC30

```
##
## AT-TLS Policy Agent Configuration file for:
## Image: SC30
## Stack: TCPIPA
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 13
## Backing Store = ATTLS
## FTP History:
##
## End of Configuration Assistant information
TTLSRule ATTLS_SC30_to_SC31~1 1
{
LocalAddrRef addr1
RemoteAddrRef addr2
LocalPortRangeRef portR1
RemotePortRangeRef portR2
Direction Inbound
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~SC30
}
TTLSConnectionAction cAct1~REXXServer
{
HandshakeRole Server
TTLSCipherParmsRef cipher1~AT-TLS__Gold
TTLSConnectionAdvancedParmsRef cAdv1~REXXServer
CtraceClearText Off
Trace 7
}
TTLSConnectionAdvancedParms cAdv1~REXXServer
{
SecondaryMap Off
}
TTLSKeyringParms keyR~SC30
{
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 1024-65535 9
}
```

In this example, the numbers correspond to the following information:

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 the SC31 VIPA address.
8. The destination port used for testing was `7000`.
9. The source port used for testing was all ephemeral ports.

Step 2: Configuring the client policies

We used the z/OSMF Configuration Assistant to create our AT-TLS policy file for our client on SC31.

We performed the following steps to define the client policies:

1. Add a z/OS image, TCP/IP stack, and enable AT-TLS for SC31
2. Add a traffic descriptor object for REXX client
3. Add a requirement map object for the REXX client
4. Add a connectivity rule for SC31

As you can see, the steps are similar to the steps for configuring the server policies. Here we repeat them with the details.

Add a z/OS image, TCP/IP stack, and enable AT-TLS for SC31

Complete the following steps:

1. In the AT-TLS Perspective panel, select **z/OS Images** in the navigation tree. Click **Add a New z/OS Image**.
2. In the z/OS image information panel, enter a name and description for the z/OS image. For this example we use `SC31`. Click **OK**.
3. When prompted to configure the TCP/IP stack, click **Yes**. We use `TCPIPB` as the stack name for this example. Enter the required information, and then click **OK**.

Add a traffic descriptor object for REXX client

Complete the following steps:

1. In the AT-TLS Perspective panel under the “Work with reusable objects” section, click **Traffic Descriptors**, and then click **Select Action** →**Add**.
2. In the New Traffic Descriptor panel (Figure 12-12 on page 568), type a name and description for the new traffic descriptor. This time, we name it `REXXClient`. Click **Select Action** →**Add**.

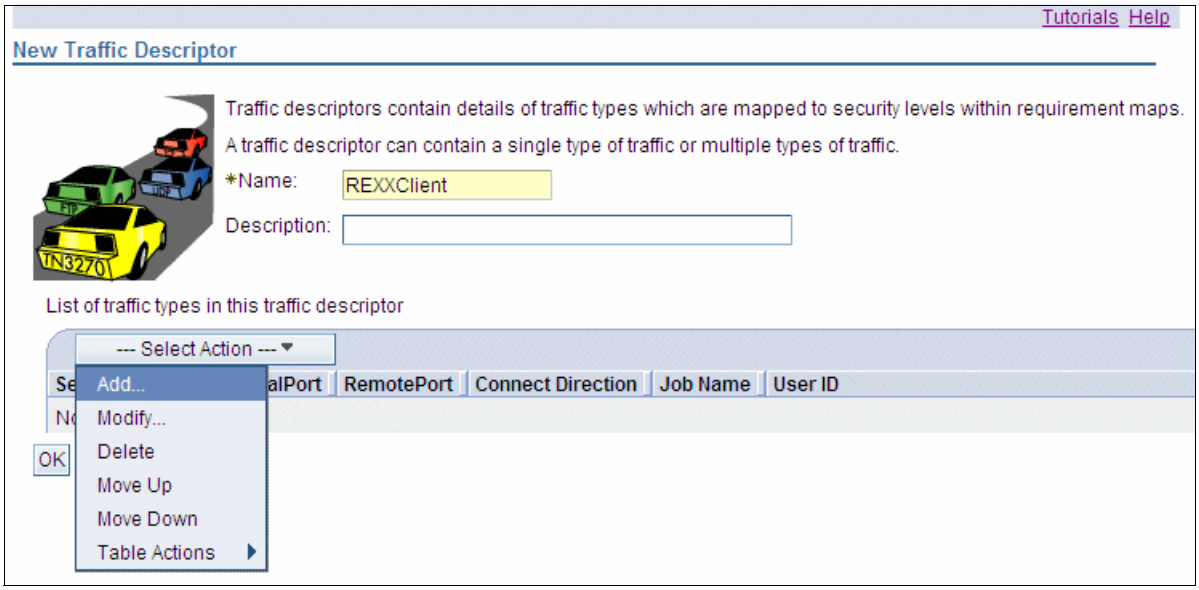


Figure 12-12 New traffic descriptor panel

3. In the New Traffic Type TCP panel (Figure 12-13), click the Details tab, and then type configuration parameters for the traffic descriptor. In this example, we use the parameters to match REXX API client settings: remote port set to 7000, TCP connect direction set to **Outbound only**, and AT-TLS handshake role set as a **Client**.

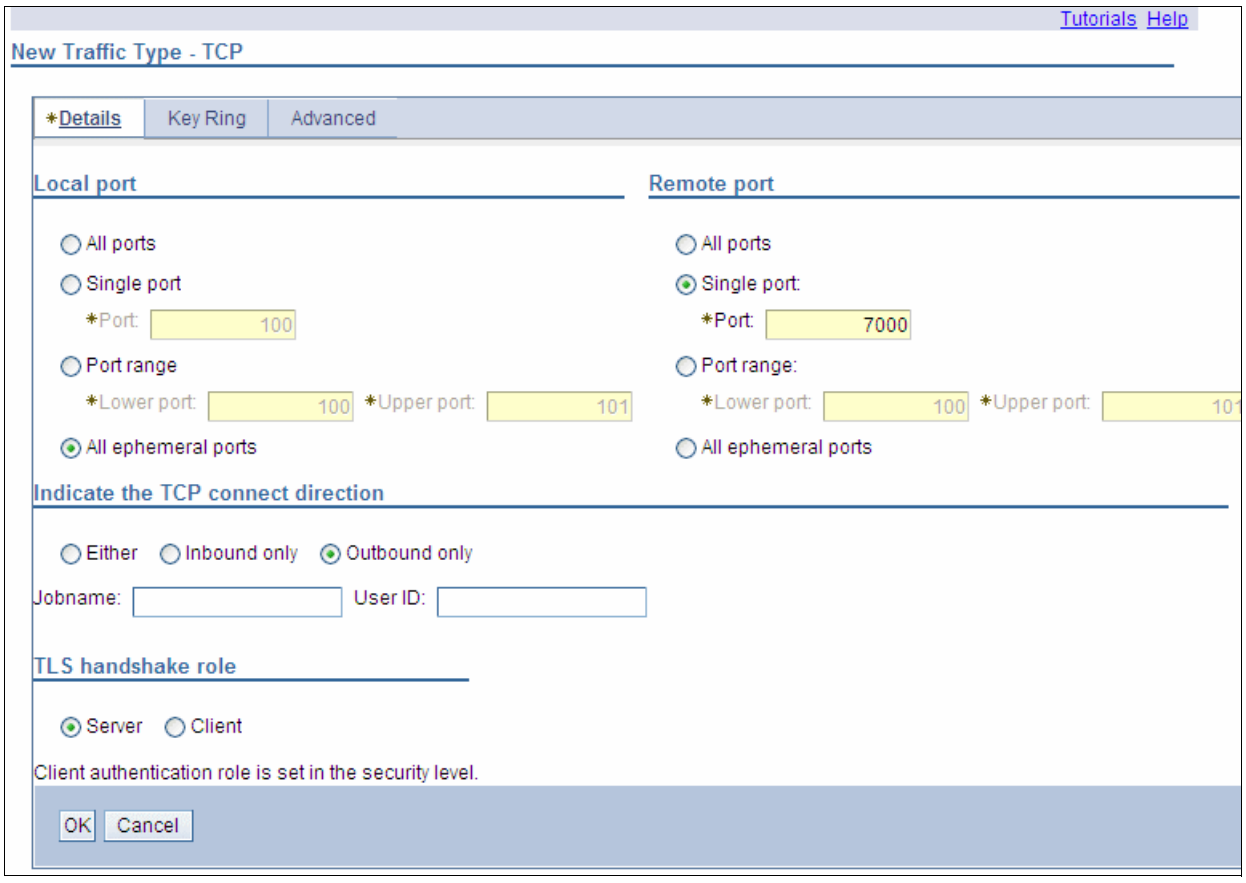


Figure 12-13 New Traffic Type - TCP, Details tab

4. Click the Key Ring tab. For our scenario, we select **Use the key ring database defined for the z/OS image.** Click **OK**.
5. In the New Traffic Descriptor panel, click **OK** to return to the AT-TLS Perspective.

Add a requirement map object for the REXX client

Complete the following steps:

1. In the AT-TLS Perspective, click **Requirement Maps**.
2. In the list of all defined requirement map objects, click **Select Action** →**Add**.
3. In the New Requirement Map panel (Figure 12-14), enter a name (ours is `CliReqMap`) and a description for the requirement map. Select the traffic descriptor that you want to configure in this requirement map object (ours is `REXXClient`). Select the correct AT-TLS - Security Level for the traffic descriptor that you added by clicking the list. In this example, we defined our client traffic descriptor with the supplied `AT-TLS GOLD` security level. Click **OK** to continue.

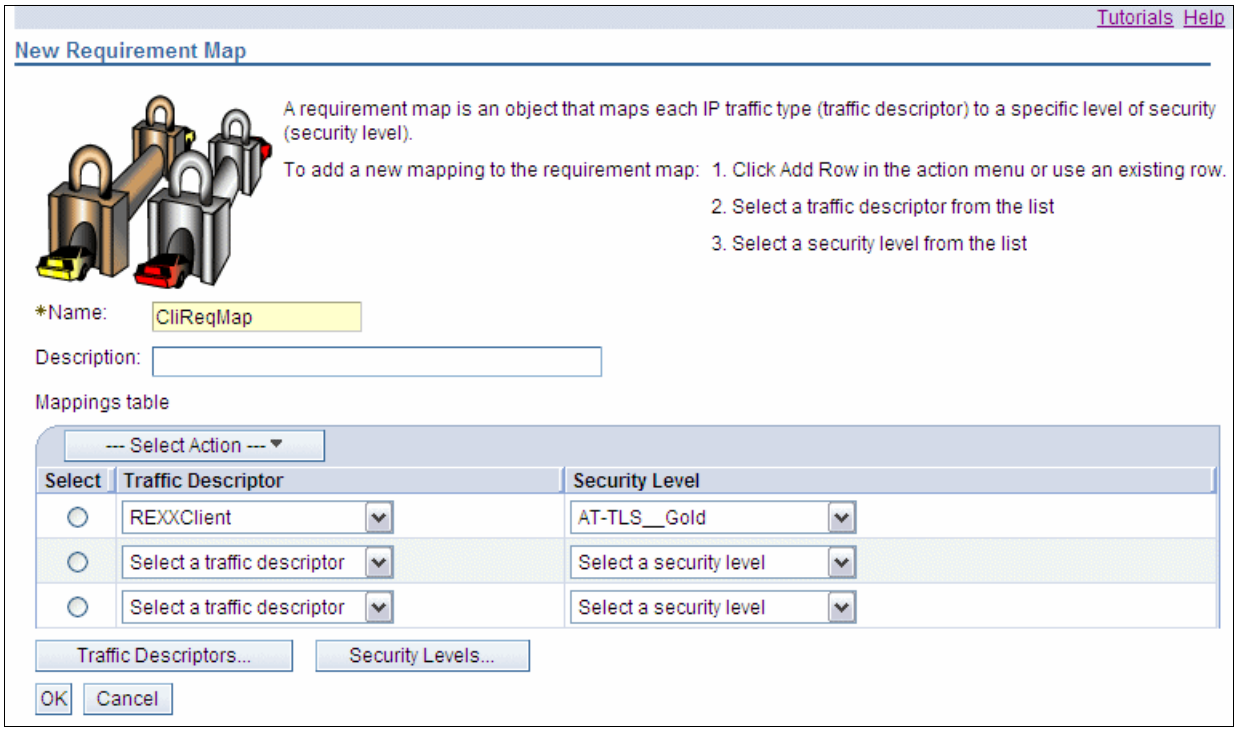


Figure 12-14 New Requirement Map panel

4. In the AT-TLS Perspective, select the z/OS image name that you created (ours is `SC31`). For Default AT-TLS key ring database on the Settings tab, we select **Simple name** and enter the name of the key ring that we created in RACF, which is `ATTLS_keyring`. Under the Default AT-TLS trace level, we select **Level 1 - Errors (to TCP/IP Joblog)**, **Level 2 - Errors (to Syslog)**, and **Level 4 - Information (to Syslog)** as shown in Figure 12-15. Click **OK**.

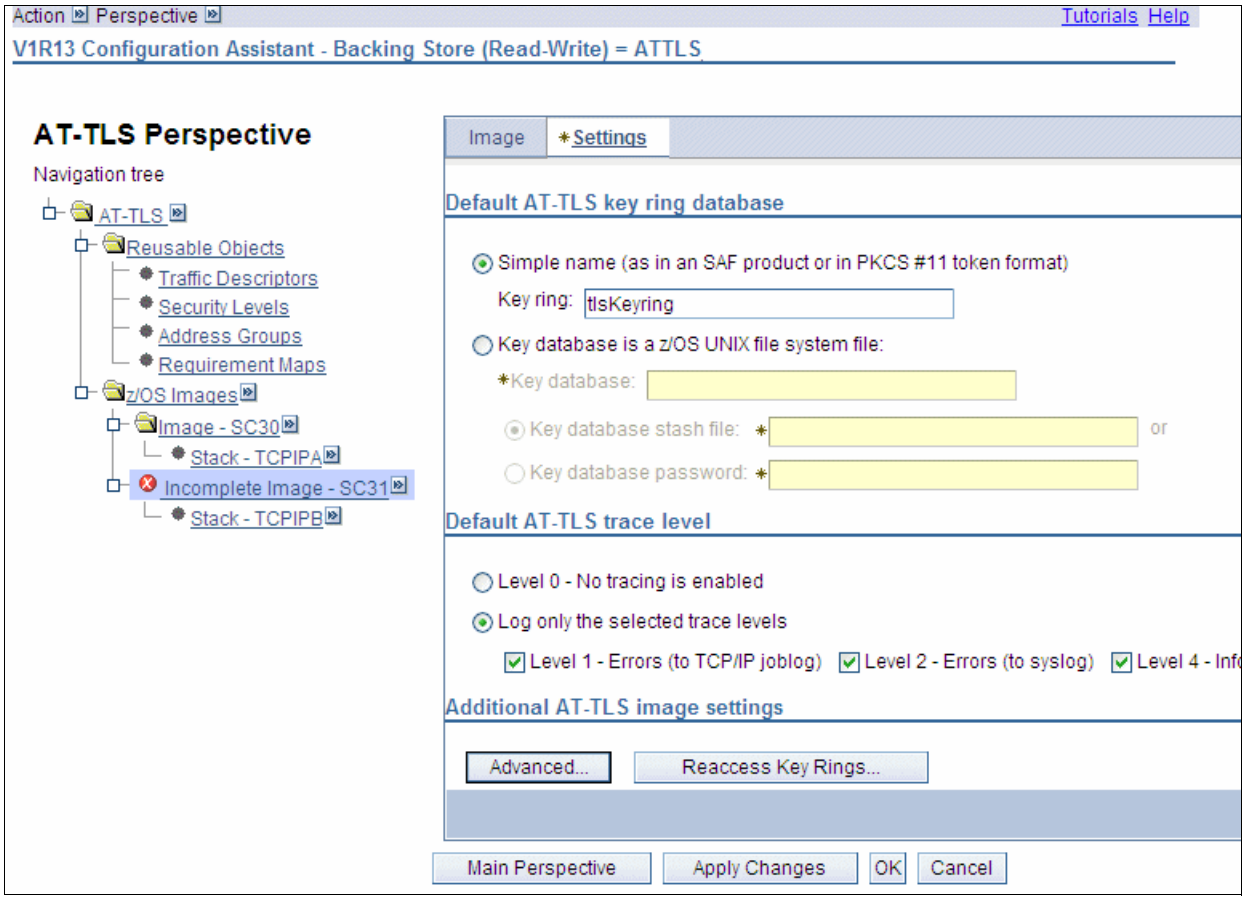


Figure 12-15 z/OS Image: AT-TLS image level settings

Add a connectivity rule for SC31

Complete the following steps:

1. On the AT-TLS Perspective panel (Figure 12-16), click the TCP/IP stack that you have created (which is `TCPIPB`), and then click **Select Action** →**Add** to set up a new connectivity rule. When the New Connectivity Rule: Welcome panel opens, click **Next**.

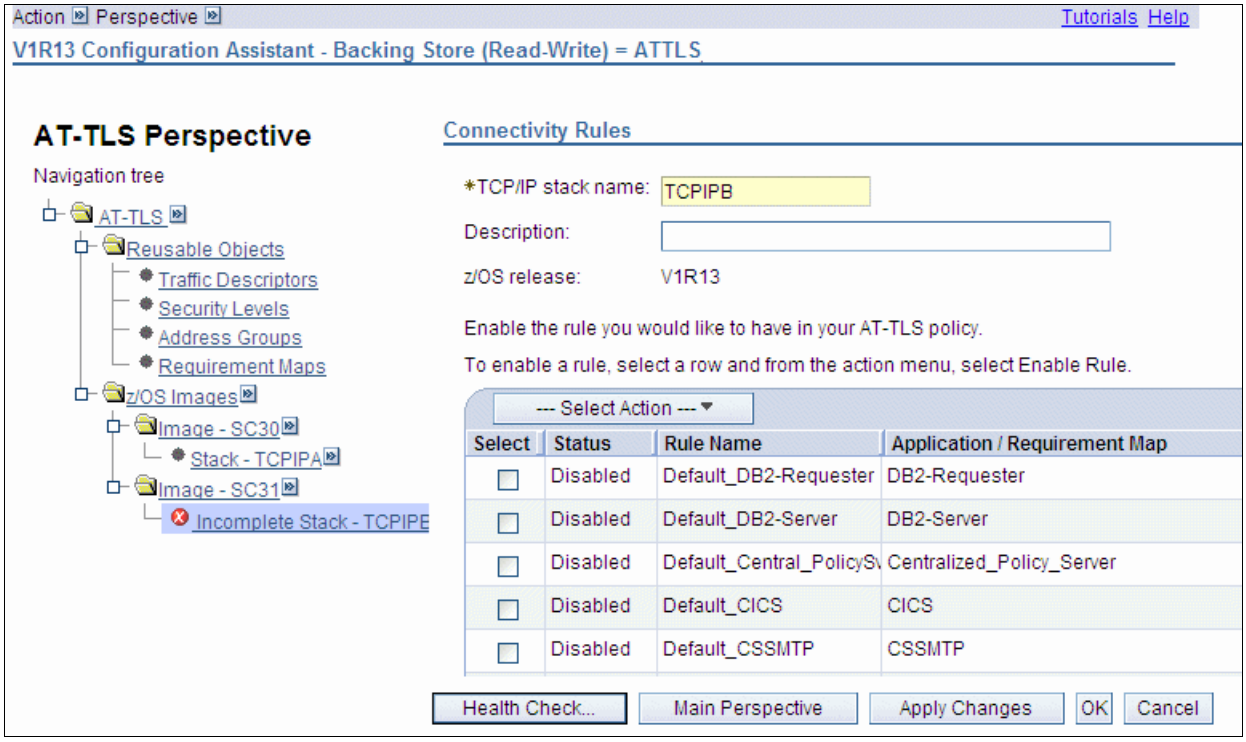


Figure 12-16 AT-TLS Perspective: adding new connectivity rule

2. The first step is to specify Data Endpoints. In the New Connectivity Rule: Data Endpoints panel, type the endpoints data and a rule name as shown in Figure 12-17. Click **Next**.

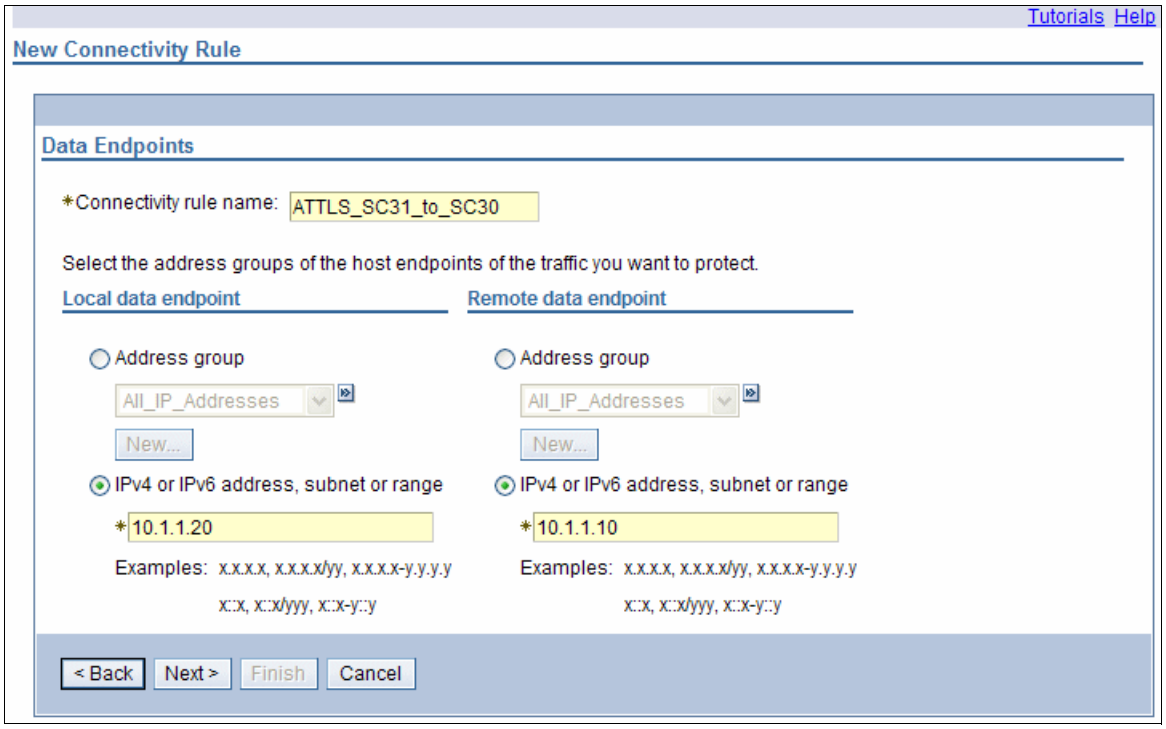


Figure 12-17 Connectivity Rule: Data Endpoints panel

3. In the New Connectivity Rule: Select Requirement Map panel, select an existing requirement map and use the client requirement map (ours is `CliReqMap`) that you created in a previous step. Click **Next** and then click **Finish**. When you return to the AT-TLS Perspective panel, click **Apply Changes**.

The resulting policy file for SC31 is displayed in Example 12-2. The outbound connection direction for this client is reflected in this policy.

Example 12-2 Client AT-TLS policy for TCPIPB on SC31

```
##
## AT-TLS Policy Agent Configuration file for:
## Image: SC31
## Stack: TCPIPB
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 13
## Backing Store = ATTLS
## FTP History:
##
## End of Configuration Assistant information
TTLSRule ATTLS_SC31_to_SC30~1
{
LocalAddrRef addr1
RemoteAddrRef addr2
LocalPortRangeRef portR1 1
RemotePortRangeRef portR2 2
Direction Outbound
```

```
Priority 255
TTLSGroupActionRef gAct1~REXEC-Client
TTLSEnvironmentActionRef eAct1~REXEC-Client
TTLSConnectionActionRef cAct1~REXEC-Client
}
TTLSGroupAction gAct1~REXEC-Client
{
TTLSEnabled On
}
TTLSEnvironmentAction eAct1~REXEC-Client
{
HandshakeRole Client
EnvironmentUserInstance 0
TTLSKeyringParmsRef keyR~SC31
}
TTLSConnectionAction cAct1~REXEC-Client
{
HandshakeRole Client
TTLSCipherParmsRef cipher1~AT-TLS__Gold
TTLSConnectionAdvancedParmsRef cAdv1~REXEC-Client
CtraceClearText Off
Trace 2
}
TTLSConnectionAdvancedParms cAdv1~REXEC-Client
{
SecondaryMap On
}
TTLSKeyringParms keyR~SC31
{
Keyring ATTLS_keyring
}
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
{
Port 1024-65535 1
}
PortRange portR2
{
Port 7000 2
}
```

Note that the local **1** and remote **2** port ranges are reversed: from the client end, the target or remote port is 7000. Locally, the client will use any ephemeral port.

Step 3: Save the policy to z/OS

Complete the following steps:

1. In the Configuration Assistant Navigation Tree, click the small symbol to the right of TCP/IP stack name (`Stack - TCPIPA` in our example), and click **Install Configuration Files**.
2. In the List of Configuration Files for Stack TCPIPA panel as shown in Figure 12-18, select the policy in the list which we have just created and click **Select Action** →**Install**.

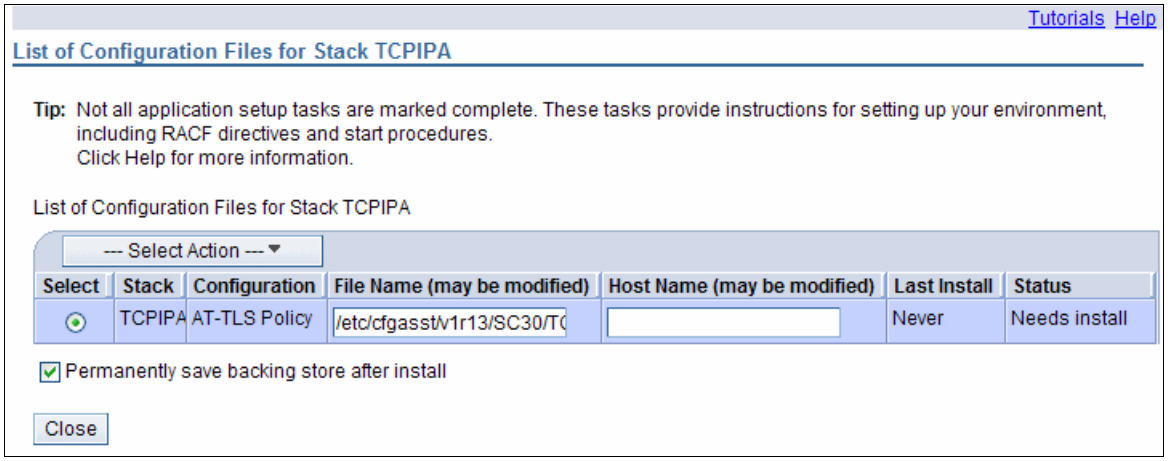


Figure 12-18 Installation panel: Installing Configuration File

3. We specify the file name to be saved as, and select **Save to disk** to save the policy to z/OS local file system, as shown in Figure 12-19. Another option is to use FTP to the other z/OS image. In that case, enter the FTP server IP address, port number, FTP user ID, and password. Click **Go**.

   When the save or transfer is complete, the result is displayed in the bottom of the same panel. Click **Close**. (Optional: Leave a comment, and then click **OK**.) Back in the List of Configuration Files for Stack TCPIPA panel, click **Close**. In our environment, because z/OSMF Configuration Assistant runs on different system, we chose to use FTP.

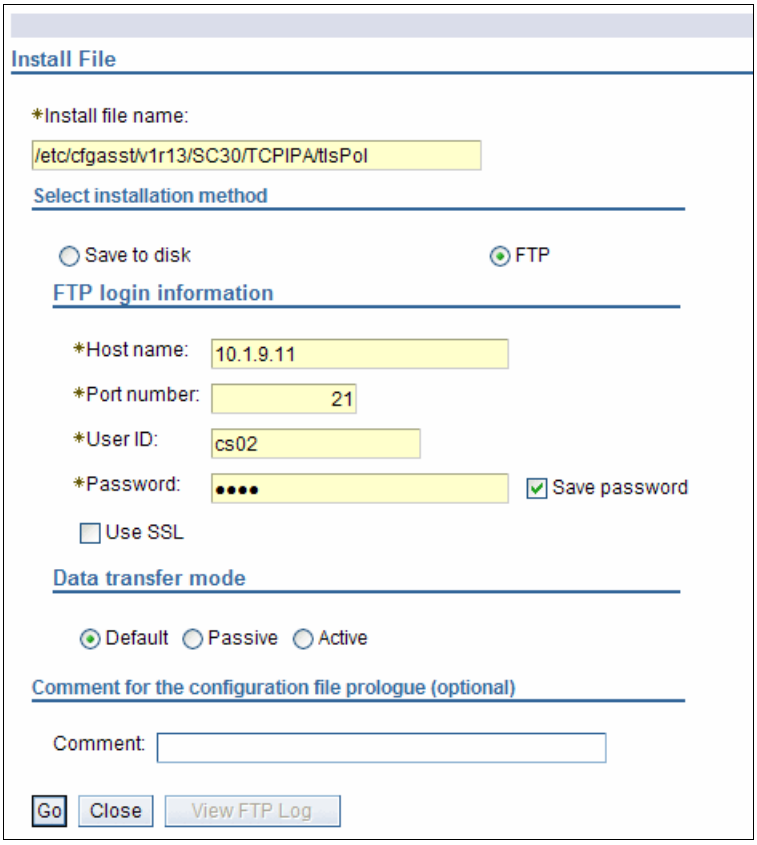


Figure 12-19 Installing files to remote host

Step 4: Modify the policy agent configuration file

Example 12-3 shows the main configuration file of the policy agent. It defines the stack-specific configuration file name for the TCPIPA stack.

Example 12-3 Main configuration file of policy agent

```
#  ******************************************************************
# /SC30/etc/pagent.sc30.conf
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent.sc30.tcpipa.conf  FLUSH PURGE 600
```

Example 12-4 shows the stack-specific configuration file for the TCPIPA stack. Add the TTLSConfig statement that points to the policy configuration file we created.

Example 12-4 Stack-specific configuration file of policy agent

```
# This file /etc/pagent.sc30.tcpipa.conf contains                  #
# image statements for TCP/IP stack tcpipd in z/OS image sc30
####################################################################
# A policy for AT-TLS support                                      #
####################################################################
TTLSConfig /etc/cfgasst/v1r13/SC30/TCPIPA/tlsPol
```

Do the same steps for SC31 TCPIPB (REXXClient) policy.

Step 5: 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 12-5. 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 12-5 Defining our digital certificates and key ring

```
SETROPTS CLASSACT(DIGTCERT,DIGTNMAP)

RACDCERT ID(CSO9) addring(ATTLS_keyring)

RACDCERT ID(csO9) CERTAUTH GENCERT                      -
         SUBJECTSDN( O('I.B.M Corporation')                      -
            CN('itso.ibm.com')                          -
            C('US')) TRUST                              -
            WITHLABEL('LOCALCA')                        -
            KEYUSAGE(certsign)

RACDCERT ID(CSO9) GENCERT                               -
         SUBJECTSDN  (CN('SC3OServerCert')              -
                      OU('ITSO')                        -
                      C('US'))                          -
                      WITHLABEL('SC3OServerCert')       -
                      SIGNWITH(CERTAUTH                 -
                      label('LOCALCA'))

RACDCERT ID(CSO9) CONNECT(ID(CSO9)                      -
                     LABEL('SC3OServerCert')            -
                     RING(ATTLS_keyring)                -
                     USAGE(personal))
RACDCERT ID(CSO9) CONNECT(ID(CSO9) CERTAUTH             -
                     LABEL('LOCALCA')                   -
                     RING(ATTLS_keyring)                -
                     USAGE(certauth))

SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
```

Step 6: 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 12-6.

Example 12-6 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
/*
```

Step 7: 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 can be permitted to this resource. Users who are not permitted to this resource cannot open sockets on this stack until the TTLS policy is installed. You might receive one of the following messages when you issue a NETSTAT or FTP command:

- `EZZ2382I Unable to open UDP socket to TCPIPA: TCPIPA is not active.`
- `EZA2590E socket error from initIPv4Connection - EDC5112I Resource temporarily unavailable. (errno2=0x12CA00B6)`

However, after the policy agent has established its policies, you will 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 12-7.

Example 12-7 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.TCPIPB UACC(NONE)
PERMIT  EZB.INITSTACK.SC31.TCPIPB CLASS(SERVAUTH) ID(*) ACCESS(READ) -
          WHEN(PROGRAM(PAGENT,EZAPAGEN))
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
SETROPTS WHEN(PROGRAM) REFRESH
```

Step 8: 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 12-8.

Example 12-8 Profile statement to enable AT-TLS

```
TCPCONFIG TTLS
```

12.2.3 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 12-9, started our APISERV application on SC30, and started our APICLN application on SC31.

Example 12-9 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 12-10, to determine whether the stack mapped a connection to our client AT-TLS policy and, if so, which policy it was mapped to.

Example 12-10 Display result of NETSTAT TTLS on client

```
D TCPIP,TCPIPB,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPB 122
TTLSGRPACTION                          GROUP ID           CONNS
GACT1~REXEC-CLIENT                     00000001               0
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

We issued a NETSTAT TTLS display on the server side as shown in Example 12-11 to determine whether the stack mapped a connection to our server AT-TLS policy and, if so, which policy it was mapped to.

Example 12-11 Display result of NETSTAT TTLS on server

```
D TCPIP,TCPIPA,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPA 599
TTLSGRPACTION                          GROUP ID           CONNS
GACT1~REXXSERVER                       00000002               0
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

Our NETSTAT ALLCONN command on SC30 showed that the APISERV application was listening on port 7000 (Example 12-12).

Example 12-12 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 12-13 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 12-13 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:            00000000000000000031
 BytesOut:           00000000000000000031
 SegmentsIn:         00000000000000000011
 SegmentsOut:        00000000000000000013
 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 as shown in Example 12-14.

Example 12-14 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
```

12.3 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 12-1. Add 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 12-1 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, see *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782, or to the appropriate IP message's manual.

For information about codes below 5000, see *z/OS Cryptographic Services System SSL Programming*, SC24-5901.

12.4 Additional information sources for AT-TLS

For additional information, see the following resources:

- *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.

13

Intrusion detection services

Intrusion is a term describing undesirable activities. The objective of an intrusion might be to acquire information that a person is not authorized to have. It might be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. It might 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) guards against these intrusions, providing protection against potential hackers.

We discuss the following topics in this chapter.

Section	Topic
13.1, “What is intrusion detection services” on page 584	The types of intrusions, and how policies are used to fend them off
13.2, “Basic concepts” on page 585	Scan detection, attack detection, and traffic regulation
13.3, “How IDS is implemented” on page 596	Using z/OSMF Configuration Assistant to create IDS policies

13.1 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 might indicate a network or system attack from someone attempting to break into or compromise a system. IDS is integrated into the TCP/IP stack's processing and can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. IDS provides a reactive system whereby it responds to the suspicious activity by taking policy-based actions that include the ability to drop attack packets. It can also limit TCP and UDP traffic by dropping packets that exceed the configured connection limit or receive queue length.

Reporting is a key piece of the TCP/IP stack's IDS support. Several IDS reporting options are available. You can choose to have events logged to the system console, to syslogd, or to both. You can choose to have packets traced to the IDS packet trace. You also may request that statistics be kept for various IDS events that occur.

The IDS messages can be retrieved by an external security information and event manager, and correlated with messages and events from other parts of the network. The external manager can analyze information from across the network and take the appropriate action.

Figure 13-1 shows the architecture and the components involved in z/OS Communications Server's IDS support.

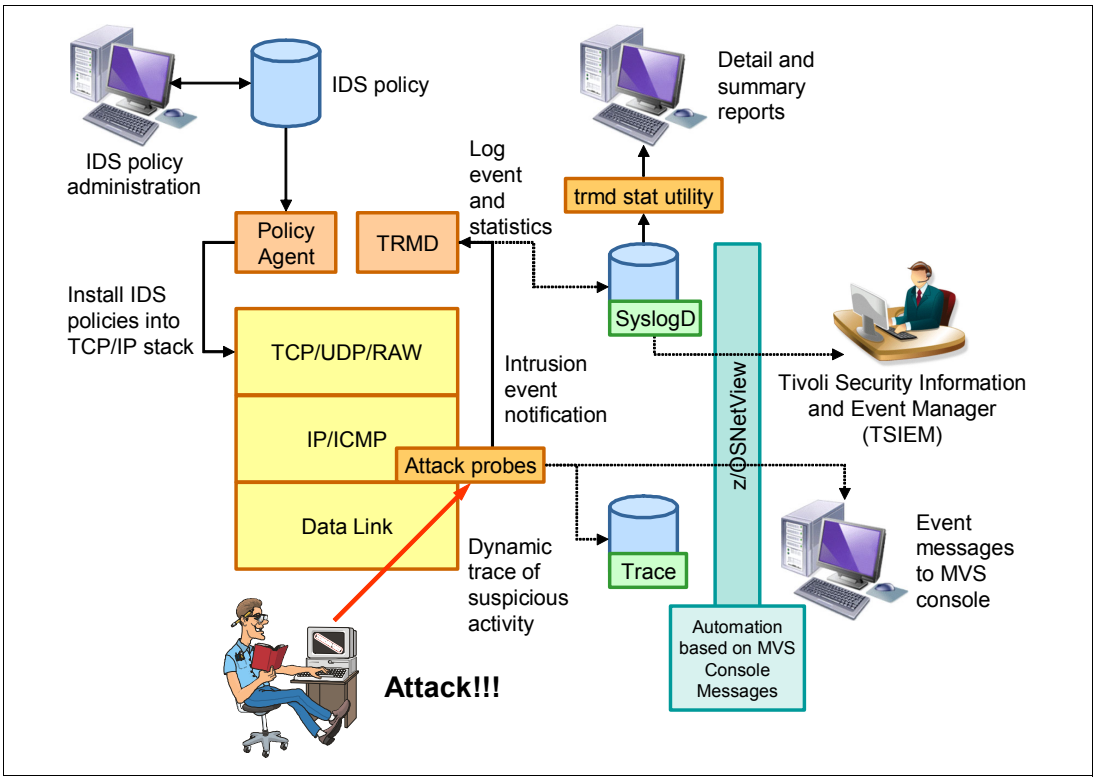


Figure 13-1 Intrusion detection services architecture

The IDS policy administrator creates an IDS policy file by using the z/OSMF Configuration Assistant. The Policy Agent reads the IDS policy and installs it in the TCP/IP stack.

The TCP/IP stack continuously monitors all packets. Based on the configured policy rules, it detects an attack and then takes the appropriate actions in the following list:

1. Discards the packet.
2. Provides TRMD with data to generate a message to be logged to syslogd.
3. Generates a message for display on the MVS console.
4. Creates a trace entry that is written to the IDS Trace data set (SYSTCPIS).
5. Counts the attack in statistics data. The statistics data is provided to TRMD on a configured time interval; TRMD generates a statistics message that is logged to syslogd.

The z/OS UNIX **trmdstat** command can be used to generate reports from the syslogd messages, providing a consolidated view of the messages.

External event managers, such as Tivoli Security Information and Event Manager, can monitor the syslogd log file. z/OS NetView automation can monitor either the MVS console messages or the syslogd messages.

In addition, the defense manager daemon (DMD) can be set up to do defensive filtering to block packets in response to an attack.

Note: Policy agent supports configuring IPv6 addresses for IDS policies in a configuration file. IPv6 traffic is monitored for scan events, attacks, and traffic regulation.
IPv6 addresses are not supported in an LDAP policy.

13.2 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?

IDS policies are supported in a policy agent configuration file and 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.

The policy agent configuration file can be created with the stand-alone version of the z/OSMF Configuration Assistant or the one running under z/OSMF. In this chapter, we cover only the policy agent configuration file that is created with the z/OSMF Configuration Assistant running under z/OSMF.

The policy information is loaded into the policy agent application during PAGENT startup. All IDS policies allow the logging of 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.

Most 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 can involve multiple packets. The correlator value helps to identify which messages 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.

13.2.1 Scan policies

Scans are used to gather information about a system. Scans are not harmful and can be part of normal operation, but many serious attacks, are preceded by information-gathering scans. For example, a port scan can determine which ports are open on a system and are potentially available for attack.

z/OS Communications Server detects a scan when multiple unique information gathering events from a single source IP address occur within a defined period of time. Because 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.

Certain network monitoring tools perform legitimate scans, so the scan exclusion list can be used to exclude these devices from scan detection. The scan exclusion list is a configured list of IP addresses and optionally ports to be excluded from scan detection

Note: The IDS scan and TR support is available for IPv6. Policy agent supports configuring IPv6 addresses for IDS policies in a configuration file. IPv6 addresses are not supported in an LDAP policy.

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 using policy. Two categories of scans are supported:

- Fast scans
- Slow scans

Fast scan

During a *fast scan*, many resources are rapidly accessed in a short time period as shown in Figure 13-2. They are usually program-driven and take less than 5 minutes.

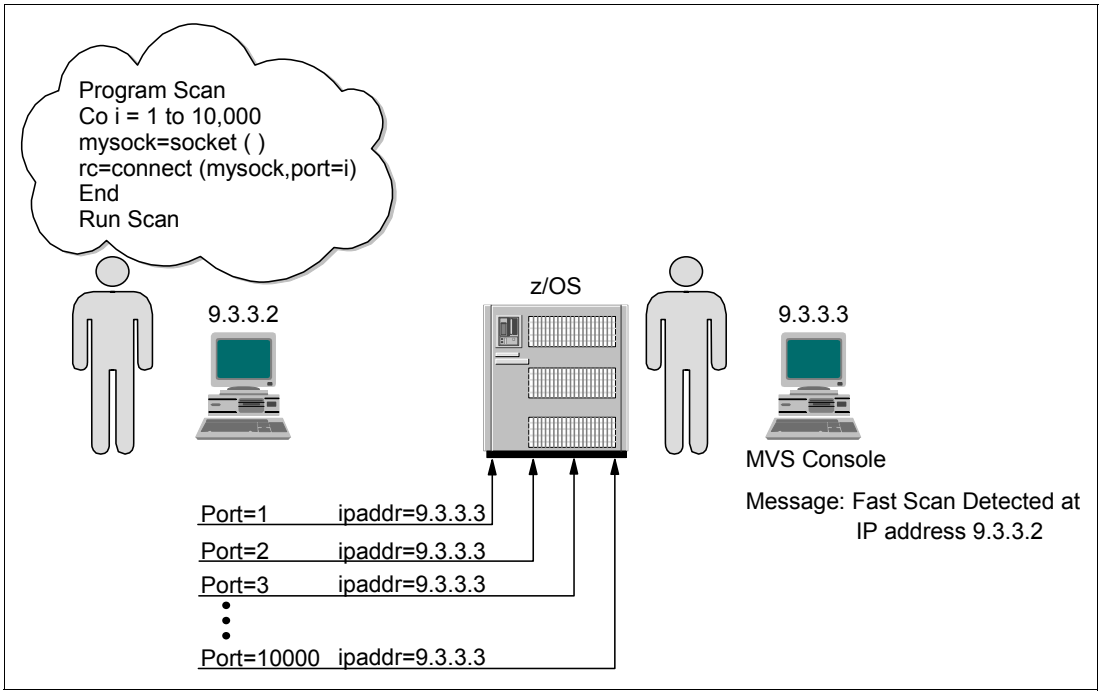


Figure 13-2 Fast scan

Slow scan

During a *slow scan*, resources are accessed intermittently over a longer period of time (many hours). A slow scan can occur when a scanner is trying to avoid detection as shown in Figure 13-3.

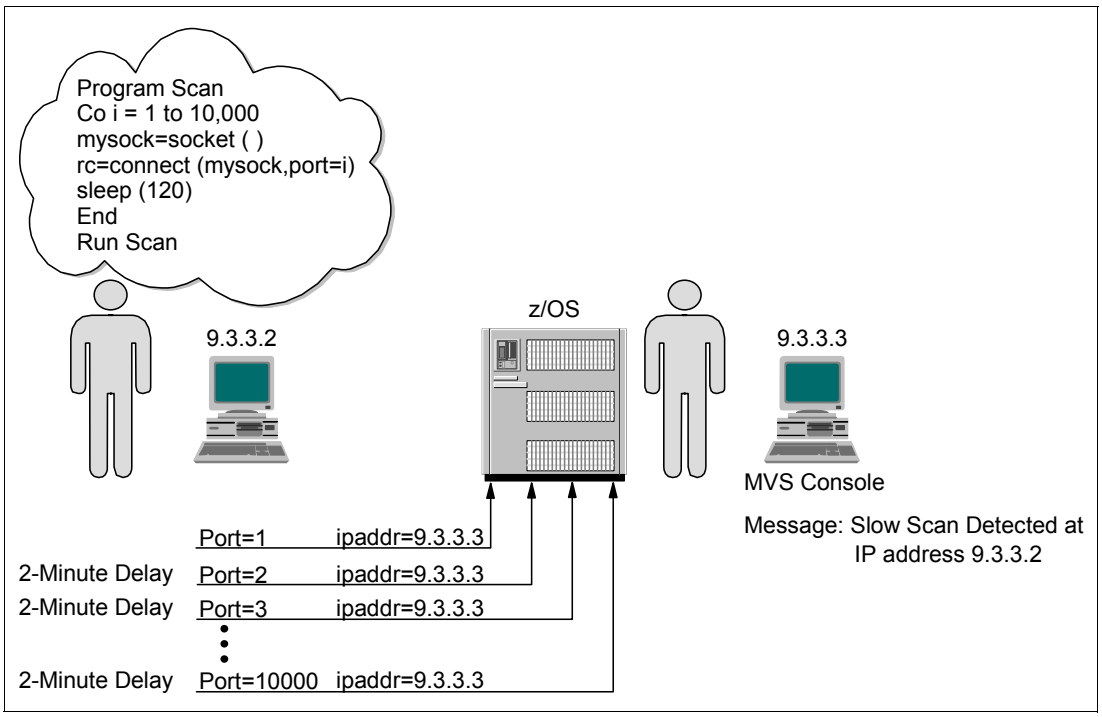


Figure 13-3 Slow scan

A fast scan scenario might 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 can be the basis for a future attack.

A slow attack is more deliberate. Occasional packets can 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.

Tip: 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 using an exclusion list

- Specify a sensitivity level by port or port range (to reduce performance impacts)
- Notify the installation of a detected scan through a 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.

- ICMPv6 scan

  ICMPv6 echo requests are used to obtain or map network information.

- TCP port scans

  TCP is a stateful protocol. There are many events that can 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 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 can be sent to the TRMD for logging to syslogd, a console message can be issued, and optionally a packet trace record can be issued. These events are 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 values (client IP address, client port, server IP address, and server port) and the IP protocol are unique 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.

13.2.2 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 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 can 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. Scanning can be considered 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 attack types that can be enabled on the z/OS Communications Server are described in Table 13-1.

For each enabled attack type, you configure the types of notification that are provided if an attack is detected. Also, you configure whether the packet should be discarded or not.

Table 13-1 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: Can provide notification.
Inbound IPv4 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 88 bytes of a packet.	► TCP/IP stack: No default action. ► IDS policy: Can 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 you defining those protocols as restricted. IP protocol restrictions are configured independently for IPv4 and IPv6 because the headers differ. For IPv4, there is an IP protocol restrictions attack type. For IPv6, there is a next header restrictions attack type.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.

Category	Attack description	Actions
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 IPv4 and IPv6 options is performed on all inbound packets, even those forwarded to another system. IP option restrictions are configured independently for IPv4 and IPv6 because IP options are implemented differently. For IPv4, there is a single IP option restrictions attack type. For IPv6, there is a hop-by-hop option restrictions attack type and a destination option restrictions attack type.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
UDP perpetual echo	Certain UDP applications unconditionally respond to every datagram received. In certain cases, such as Echo, CharGen, or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases, it might 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: Can 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: Can 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. Generally, you should restrict the TCP protocol on the outbound raw rule. Outbound raw restrictions are configured independently for IPv4 and IPv6	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
TCP SYN flood	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: Can provide notification.
Interface flood	An interface flood is detected when a large number of the incoming packets over an interface is being discarded. Enabling the flood attack allows you to receive notification about the attack	► TCP/IP stack: Discards the packets. ► IDS policy: Can provide notification.

Category	Attack description	Actions
Data hiding	Detects inbound IP packets that might contain hidden data. Checking is done for both IPv4 and IPv6 packets	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
TCP queue size	Detects attacks targeting TCP/IP send, receive, or out-of-order queues. Constraint triggered when data on queue for at least 60 seconds or a “fixed” limit of data on queue for at least 30 seconds	► TCP/IP stack: Storage marked page eligible. ► IDS policy: Can provide notification and cause the connection to be reset.
Global TCP stall	A global TCP stall condition is detected when at least 50% of the active TCP connections are stalled and at least 1000 TCP connections are active	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause all stalled connections to be reset.
EE malformed packet	Malformed EE packets are attempts to crash the EE protocol stack.	► TCP/IP stack: No default action ► IDS policy: Can provide notification and cause the packet to be discarded
EE LDLC check	EE LDLC control packets must flow over the EE signalling port. Control packets received on another port may represent an attacker's attempt to probe or crash a system.	► TCP/IP stack: No default action ► IDS policy: Can provide notification and cause the packet to be discarded
EE port check	EE packets are expected to use the same source and destination port numbers. Packets received with differing port numbers may represent an attacker's attempt to probe a system.	► TCP/IP stack: No default action ► IDS policy: Can provide notification and cause the packet to be discarded
EE XID flood	An attacker may attempt to consume system resources, or block or delay legitimate EE connections, by generating a flood of EE XIDs. This attack type monitors for EE XID timeouts that can result from an attacker's spurious XIDs.	► TCP/IP stack: No default action ► IDS policy: Can provide notification

13.2.3 IPv6 Support

All the above attack types are supported for both IPv4 and IPv6, except the IP fragment restrictions attack type, which is supported only for IPv4. Certain classes of attacks (IP protocol restrictions, IP option restrictions and outbound RAW restrictions) have separate IPv4 and IPv6 configuration because of the ways in which the IPv4 and IPv6 headers differ.

13.2.4 IDS Reporting

Reporting is a key piece of the TCP/IP stack’s IDS support. z/OS Communication Server’s IDS support provides protection based on stack-level knowledge, that is, activity in that stack. The TCP/IP stack’s messages can be correlated with messages and events from other parts of the network to take the appropriate actions.

Several IDS reporting options are available so you can do the following tasks:

- Have events logged to the system console, to syslogd, or to both.
- Have packets traced to the IDS packet trace.
- Request that statistics be kept for various IDS events that occur.

Attack policy notification

With the IDS attack policy notification, attack events can be logged to syslogd and the system console. For all attack categories (except EE XID flood, TCP queue size, and global TCP stall), a single packet triggers an event. The three ways to configure IDS policies are LDAP, policy agent flat file, and Configuration Assistant.

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. With the LDAP method, there is no default; therefore you should code one. 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.

Tip: 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, specify exception statistics. With exception statistics, a statistics record will be generated only for the category of attack if the count of attacks is non-zero. If normal statistics are requested, a record will be generated each statistics interval regardless of whether an attack has been detected during that interval.

Attack policy tracing

IDS attack policy tracing uses the component trace facility SYSTCPIS. For all attack categories except EE XID flood, TCP queue size, and global TCP stall, 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. No IDS tracing is provided for EE XID flood, TCP queue size and global TCP stall.

Tip: To use the attack policy tracing through 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.

13.2.5 Traffic regulation policies

The IDS traffic regulation (TR) policies are used to limit memory and time:

- Memory usage
- Queue delay time

The two types of traffic regulation policies are as follows:

- TCP traffic regulation policies
- 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 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

  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 13-1.

Example 13-1 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
```

Tip: If a host does not currently have any connections open on the port and connections are available, that host will always be allowed at least one connection.

Quality of service policy

Multi-user source IP addresses can 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. Generally, UDPQUEUELIMIT should 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.

Tip: 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 13-2 for the internal values.

Table 13-2 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 might 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 bursty traffic patterns might 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 might have given up or retransmitted the datagram before it is processed.

Slow applications with consistently high traffic rates can 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 can reduce the risk to system buffer storage under random port flood attacks, with little impact on percentage of datagrams lost.

13.3 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 4, “Policy agent” on page 103, for discussion and implementation examples for PAGENT.

The z/OSMF Configuration Assistant graphical user interface (GUI) tool can be used to create the IDS policies. The policies are flat files; you can also change the IDS policy configuration file manually.

13.3.1 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 4, “Policy agent” on page 103.

Tip: 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 13-2 shows the IDS statements in the TCP/IP stack configuration file used by policy agent.

Example 13-2 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)'
```

13.3.2 The z/OSMF Configuration Assistant

The z/OSMF Configuration Assistant enables centralized configuration of intrusion detection policies for z/OS. 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.

Note: The following attack types are supported only in a policy agent configuration file, not in LDAP: Data hiding, TCP Queue Size, Global TCP Stall, IPv6 next header restrictions, IPv6 hop-by-hop option restrictions, IPv6 destination option restrictions, EE malformed packet, EE LDLC check, EE port check, EE XID flood, and IPv6 addresses.

The z/OSMF Configuration Assistant is a tool designed to allow a network administrator to produce a configuration file for the policy agent (PAGENT).

The z/OSMF Configuration Assistant 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 through FTP, or moved to and placed in the PAGENT configuration file manually.

The z/OSMF Configuration Assistant 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.

13.3.3 Configuring IDS policy using the z/OSMF Configuration Assistant

This section can 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 as follows:

- 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
- Creating IDS objects and rules
- Creating reusable objects

Upon completion of this chapter, you will have created the following items:

- Reusable objects
- A scan global policy
- An attack, scan event, and TR TCP (condition, action, and policy)

Creating a new IDS policy

Complete the following steps:

1. From z/OSMF Configuration Assistant, click **Action** → **Open** → **Create a New Backing Store** (Figure 13-4).

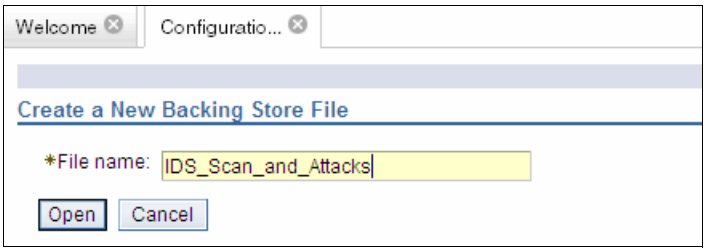


Figure 13-4 Creating a new backing store

2. Enter a file name for the new configuration, and click **Open**.
3. In the welcome window, select **Intrusion Detection Services** as shown in Figure 13-5.

> **Tip:** z/OSMF Configuration Assistant help is available through the Help menu option.

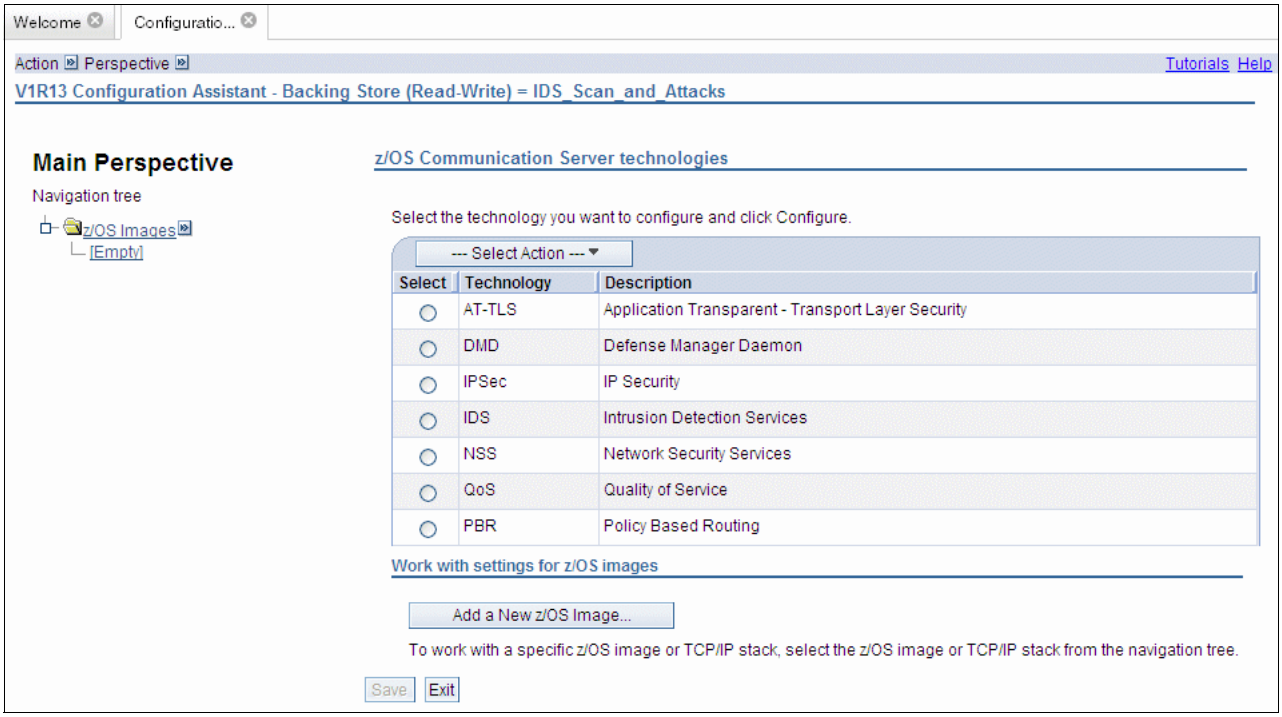


Figure 13-5 Welcome window Choosing to create an IDS configuration file

4. Click **Select Action** → **Configure**. The basic window for configuring IDS policy opens (Figure 13-6).

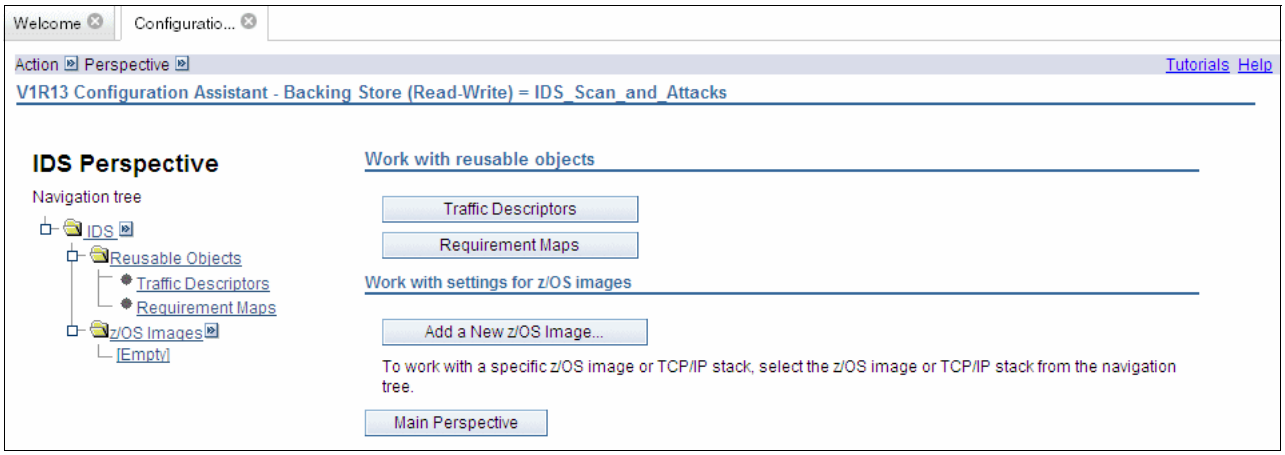


Figure 13-6 IBM Configuration Assistant first window when choosing IDS

Adding a z/OS image to the policy

Complete the following steps:

1. Click **Add a New z/OS Image**.
2. In the New z/OS Image: Information window, type the name of the z/OS image and a description (Figure 13-7), and then click **OK**.

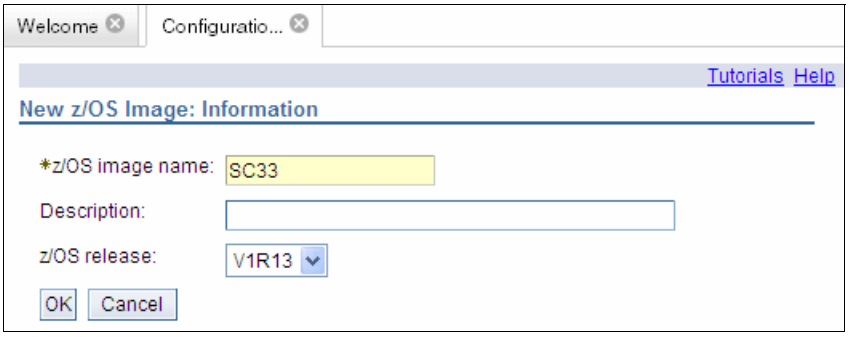


Figure 13-7 Adding new zOS Image

3. At the confirmation panel to configure the TCP/IP Stack (Figure 13-8), click **Yes**.

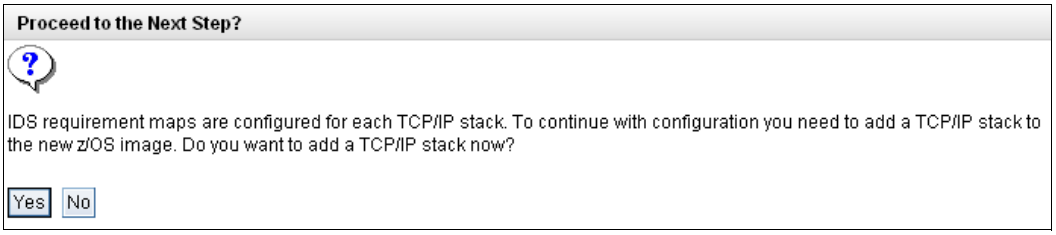


Figure 13-8 Confirmation to configure TCPIP Stack

Adding a TCP/IP stack under the z/OS image

In the New TCP/IP Stack: Information window, type the TCP/IP stack name and definition as shown in Figure 13-9, and click **OK**.

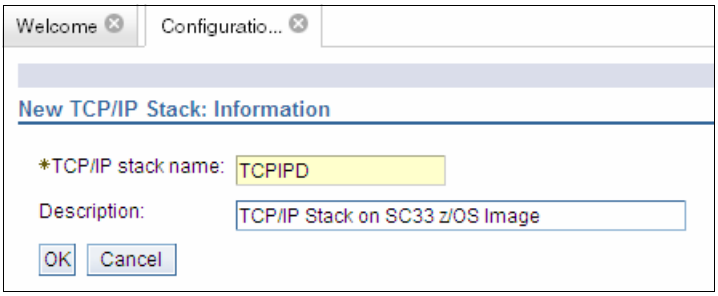


Figure 13-9 AddingTCPIP stack to the zOS image

The default IDS requirement map settings

The Proceed to the Next Step window offers four options:

- Select View Details to examine the default settings.
- Select View Tutorial to view a brief tutorial about requirement maps
- Accept the default requirement map settings.
- Have a wizard walk me through the creation of a new requirement map.

As a start, we use the default requirement map settings:

1. To see the default settings, click **View Details**.
2. Select **Accept the default Requirement Map settings** (Figure 13-10) and click **OK**.

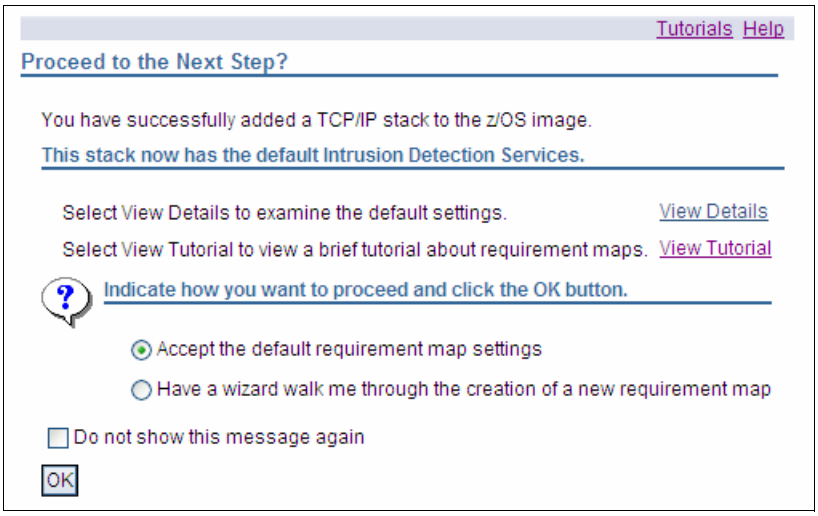


Figure 13-10 Choosing the default settings

The next window (Figure 13-11) displays the requirement map selected for stack TCPIPD.

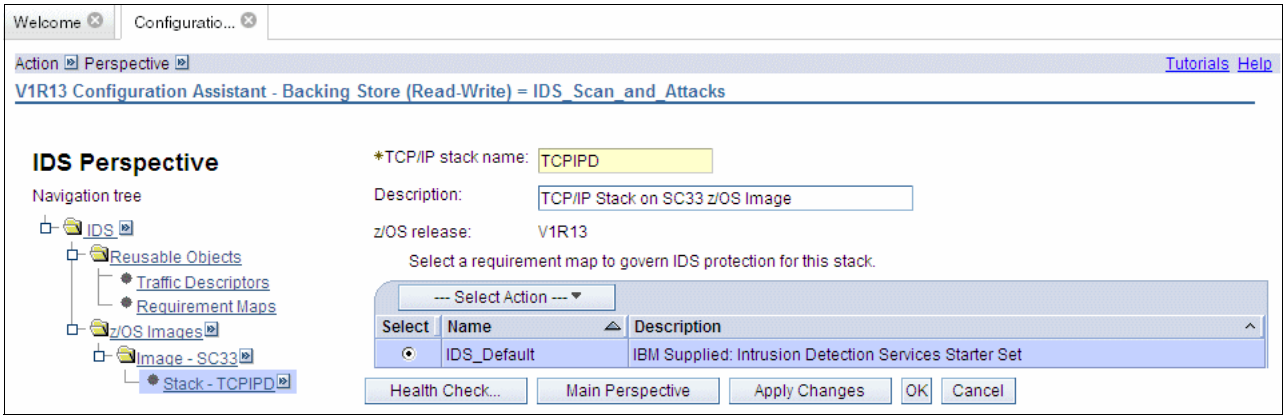


Figure 13-11 z/OS Image Settings window: After defining the default settings for IDS policy

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.

- Required: Establish one condition set and one action set in at least one policy rule.
- Optional: Specify that those rules apply only during validity periods.

In this section, you specify IDS policy rules, which can include condition sets, actions, policy keyword sets, and validity periods. Only one policy rule and associated actions can be applied to a particular packet.

Use this section to create an IDS policy rule:

- Create reusable objects that will be used in your action and condition sets.
- Create the actions and conditions based on those reuseable objects.
- Build your policy rule from the available condition and action sets.

Creating reusable objects

The reuseable objects are designed to be incorporated into multiple policies, conditions, or actions, depending on the type of object.

The two types of reusable objects are as follows:

- Traffic descriptors

  Reusable objects that describe properties of network traffic such as protocols and ports.

- Requirement maps

  Reusable objects that are entire sets of intrusion detection requirements. They contain, by default, only the default requirement map settings object.

Click **Traffic Descriptors** from the navigation tree (Figure 13-12) to see the reusable traffic descriptors that are available.

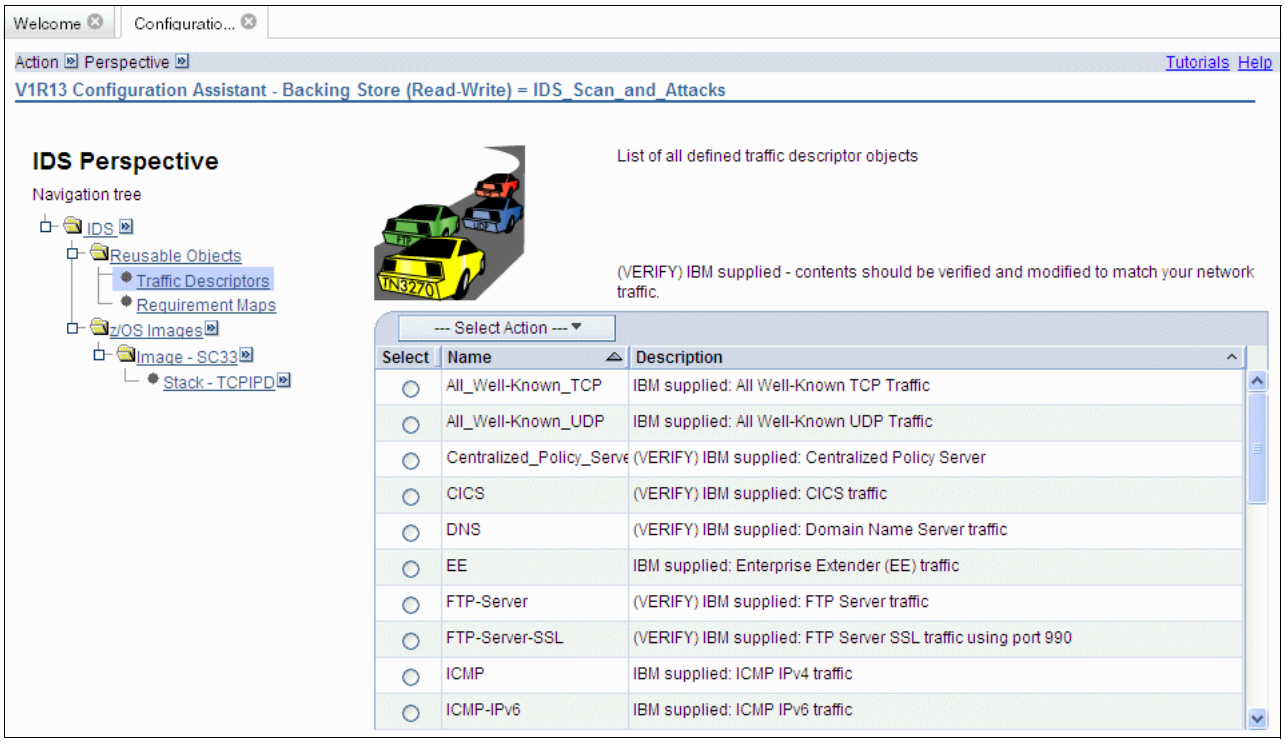


Figure 13-12 Traffic descriptors for IDS policy

Creating a new IDS policy on attack protection

The next steps configures an attack protection rule:

1. Select **Requirement Maps** from the IBM Configuration Assistant navigation tree.
2. Click **IDS** → **Select Action** →**Add**. The New Requirements Map window opens (Figure 13-13 on page 602).

3. Type a name and description as shown in Figure 13-13, and click **Next**.

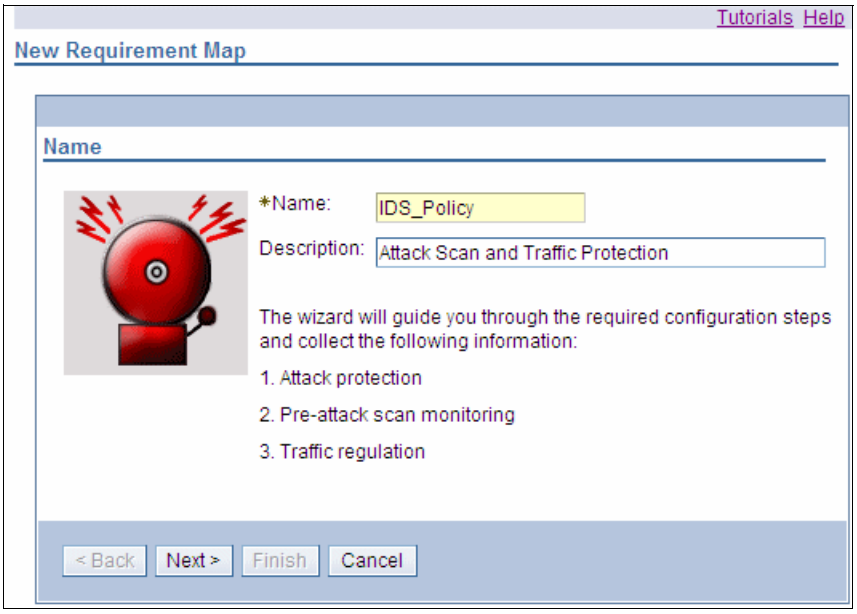


Figure 13-13 Add Requirement Map: Name - Adding an attack protection rule

Enabling protection from attacks

An *attack* can be a single packet designed to crash or hang a system, or multiple packets designed to consume a limited resource, which causes a network, host system, or application to be unavailable to its intended users (a *denial of service attack*). The IDS attack policy lets you turn on attack detection for one or more categories of attacks independently of each other.

Figure 13-14 on page 603 shows the *Attacks* panel, which indicates the attack types that you can enable protection from:

- Data Hiding Attack
- IPv6 Outbound Raw Attack
- IPv6 Destination Options Attack
- IPv6 Hop-by-Hop Options Attack
- IPv6 Next Header Attack
- TCP Queue Size Attack
- Global TCP Stall Attack
- Flood Attack
- Perpetual Echo Attack
- IPv4 Protocols Attack
- IPv4 Options Attack
- ICMP Redirect Attack
- Malformed Packet Attack
- IPv4 Outbound Raw Attack
- IPv4 Fragment Attack
- EE Malformed Packet Attack
- EE LDLC Check Attack
- EE Port Check Attack
- EE XID Flood Attack

You can modify and change the specific values of some of these attack types such as flood, perpetual echo, IP protocols, IP options, and outbound raw.

You must specify the actions to be taken when an attack is detected. You can select one or more of the following actions, depending on the attack type.

- Event logging
- Statistics gathering
- Packet tracing
- Discarding the attack packets

Complete the following steps

1. In the New Requirement Map - Attacks window (Figure 13-14), verify that the **Enable attack protection** check box is selected.
2. We chose to retain the default configuration that is supplied with the z/OSMF Configuration Assistant as shown in the figure. Click **Next**.

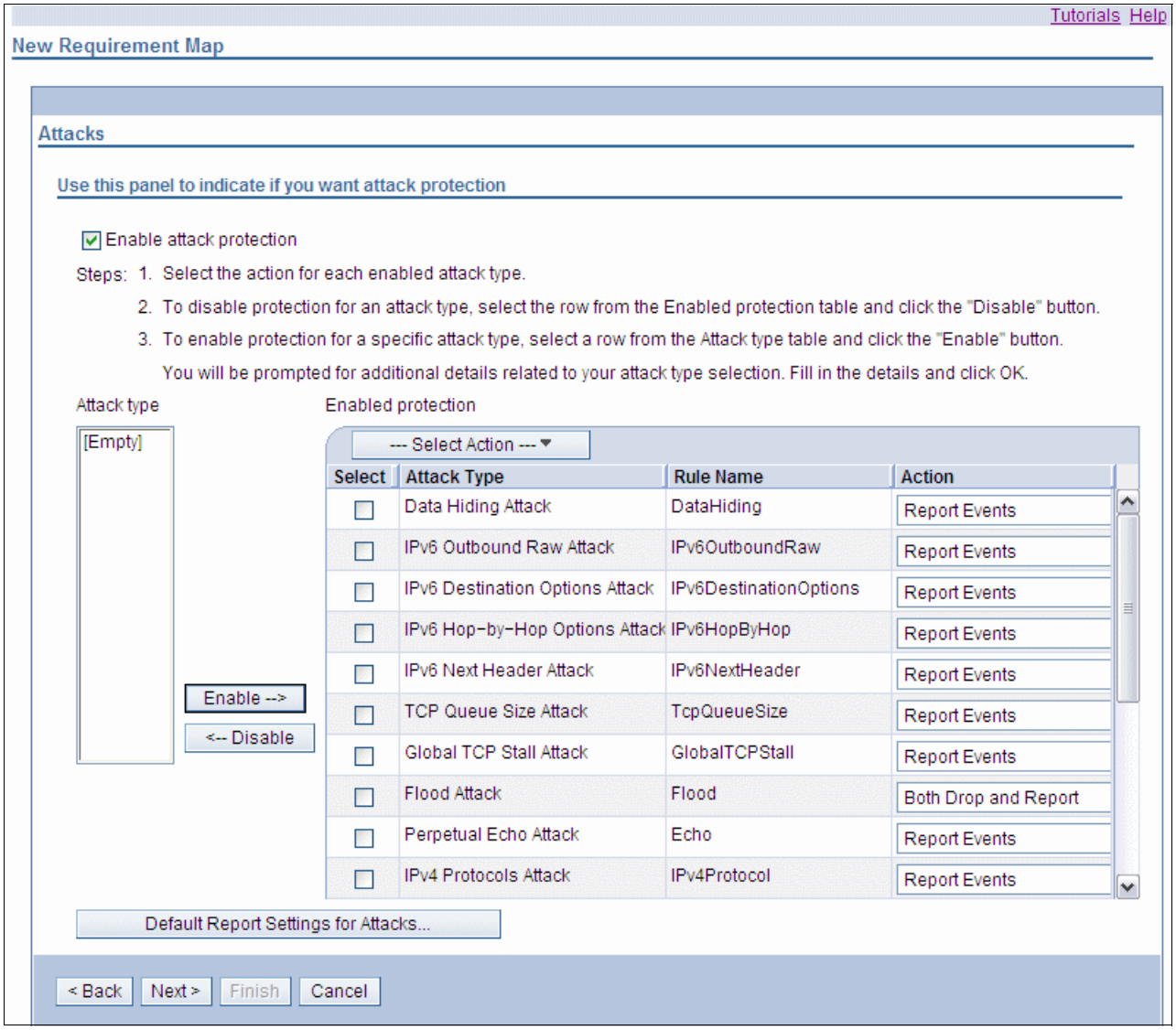


Figure 13-14 Add Requirement Map: Attacks - Configuring attack protection

Tip: For information about the attack types, select **IDS** → **Select Action** → **View Details**.

A new Requirements Map window opens (Figure 13-15).

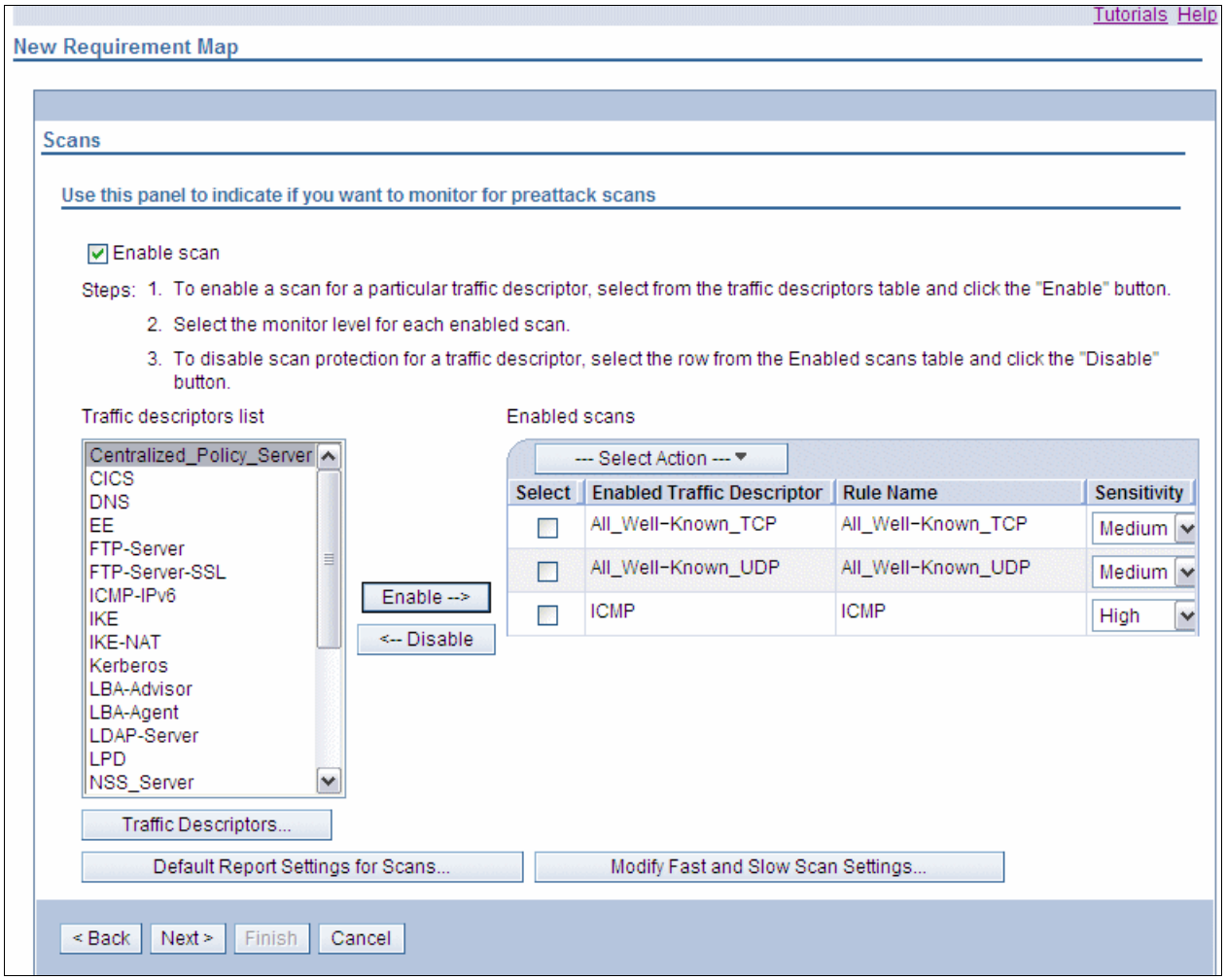


Figure 13-15 Add Requirement Map: Scans window for selecting the default-enabled scans

Specifying sets of global scan detection parameters

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.

Complete the following steps:

1. In the Add Requirement Map: Scans window (Figure 13-15), select **Enable scan**.

   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 **Modify Fast and Slow Scan Settings** and then clicking **OK**. We accepted the default values.

Scan events are 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 events can be classified as normal, suspicious, or highly suspicious. For more details, see “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. You can set one of the following sensitivity levels:

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.

We accept the default configuration of enabled scans as shown in Figure 13-15 on page 604.

Order is important: 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.

2. Click **Default Report Settings for Scans**. See Figure 13-15 on page 604.
3. In the Report Types window (Figure 13-16), indicate where to report IDS events. We selected **System console** and **SYSLOGD**. Click **Modify Details**.

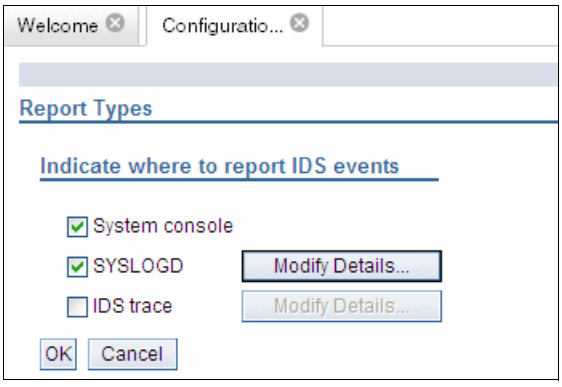


Figure 13-16 Report Types window to indicate where to report IDS scan events

4. When you are configuring your IDS policy, select a high-value log level. We selected **6 -** information, as shown in Figure 13-17. In the Report Events Details window, click **OK**.

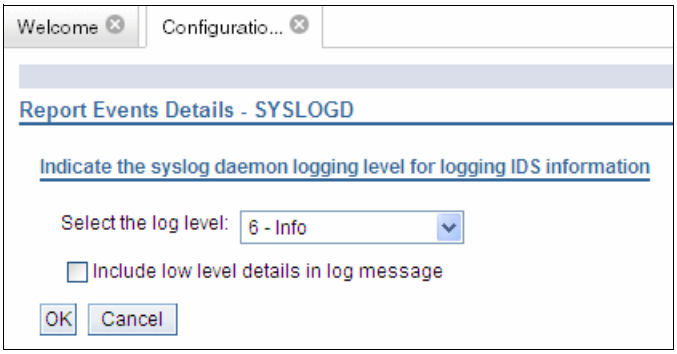


Figure 13-17 Report Events Details - SYSLOGD window - Selecting a log level

5. In the Report Types window, click **OK**.
6. In the New Requirement Map, Scans window, click **Next**.

Setting traffic regulations (TRs)

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.

Complete the following steps:

1. In the Add Requirement Maps: Traffic Regulation window (Figure 13-18 on page 607), select the **Enable traffic regulation** check box.
2. In the Traffic descriptors list, select **All_Well-Known_TCP**, and click **Enable**.

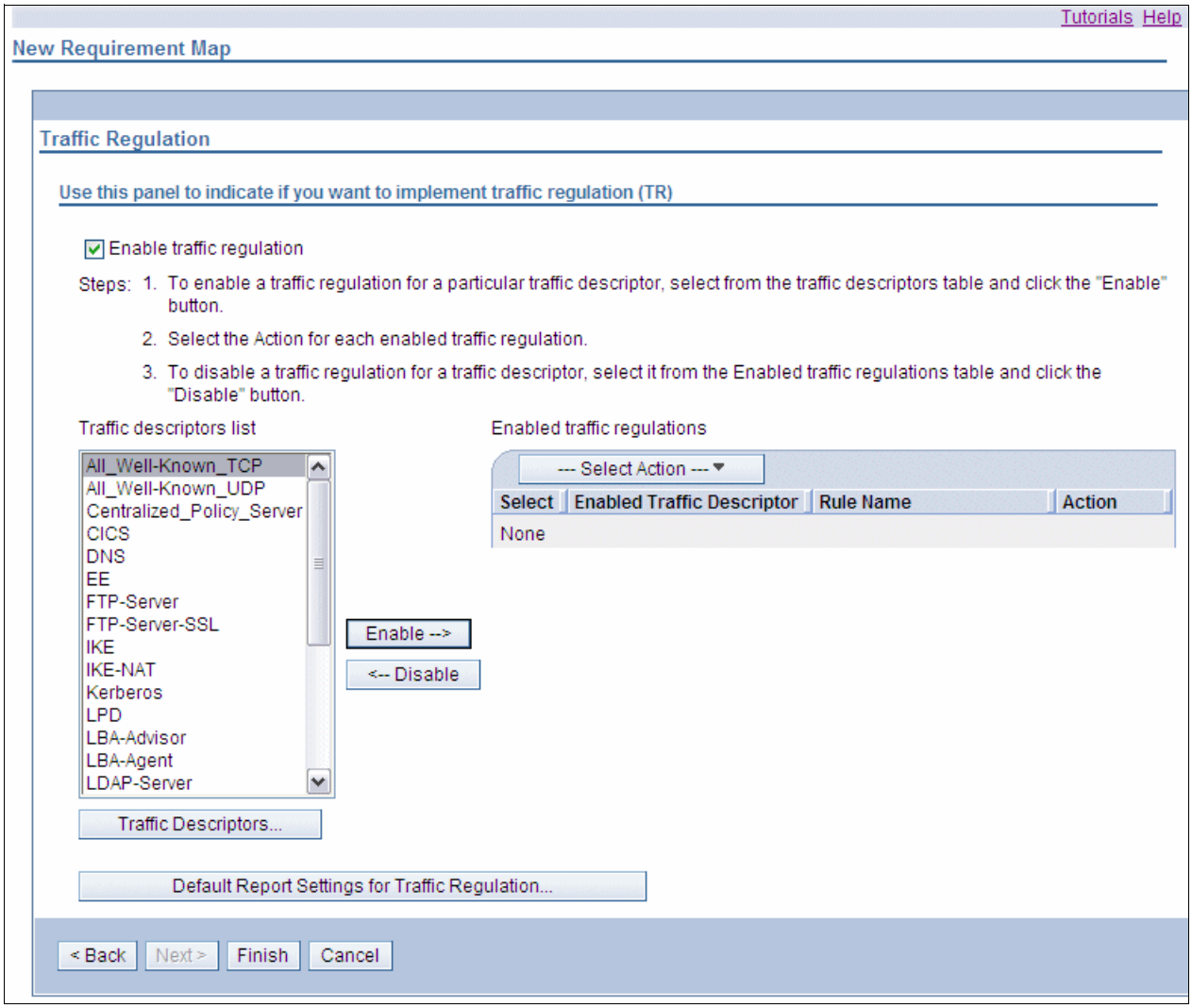


Figure 13-18 Adding All_Well-Known_TCP

3. In the Traffic Regulation details window, we accepted the default values, as shown in Figure 13-19 on page 608. However, consider the following information 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, consider the following limits:
     - 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.

4. Click **OK** and then click **Finish**.

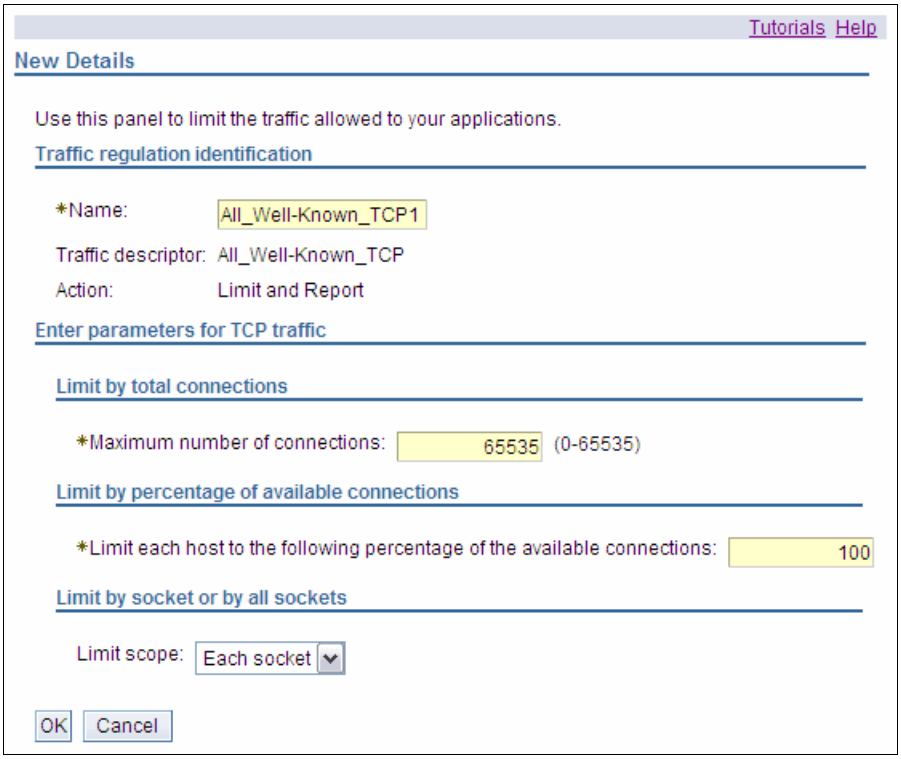


Figure 13-19 Traffic Regulation details window

Setting 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, complete the following steps:

1. Select **IDS** → **z/OS Images** → **Image -** ***image_name*** → **Stack -** ***stack_name*** from the IBM Configuration Assistant navigation tree (Figure 13-20 on page 609).
2. Select **IDS_Policy** from the requirement map list. Click **Select Action** → **Set Addresses.**

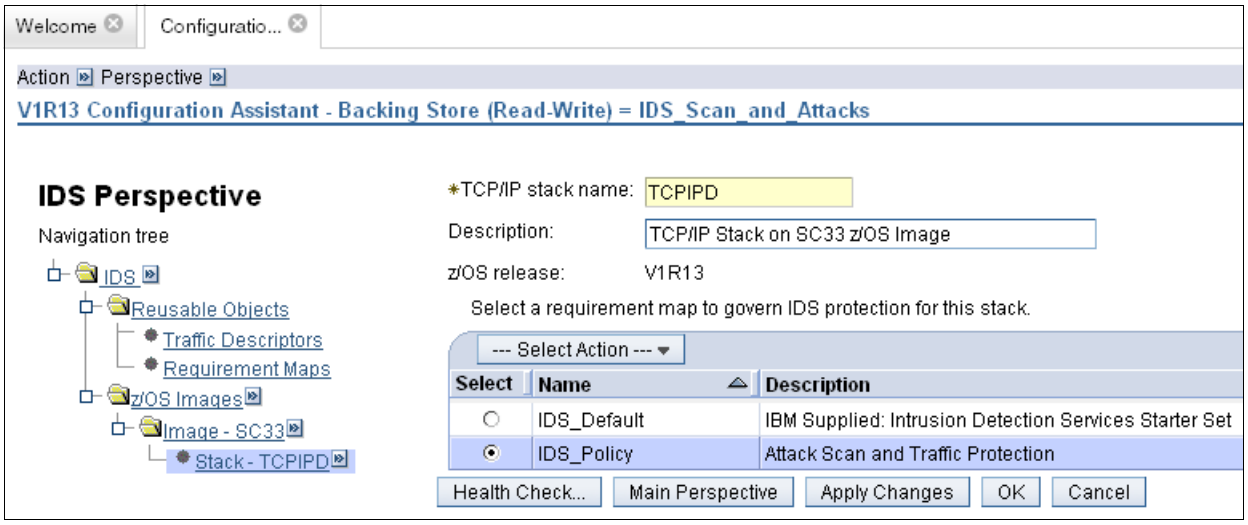


Figure 13-20 IBM Configuration Assistant - Requirement Maps for stack TCPIPD

3. In the Advanced Stack Settings window, select the traffic descriptor for which you want to set remote exclusions and local addresses, and then click **Set Remote Exclusions and Local Addresses**.
4. In the Stack Level Scan Settings window, select **Exclude the following remote addresses and ports** and click **Add**.
5. In the Scan Exclusion Remote Details window, type the remote exclusion address and port, and click **OK**.

   We did not add any exclusions, and therefore clicked **Cancel**.
6. If you want to set local addresses for which the rule will apply, click the **Local Addresses** tab in the Stack Level Scan Settings window, and then select **Rule applies to only the following specified local addresses**. Click **Add** and type 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 window, click **OK**.
8. In Advanced Stack Settings window, click **OK**.

Tip: Click **Health Check** in the IDS Perspective panel (Figure 13-20 on page 609) with the stack selected before you install the configuration file in the z/OS image. The Health Check reports any errors in your policy, which we found helpful.

13.3.4 Installing the IDS policy

After you finish configuring the IDS policy, complete the following steps:

1. Click **Main Perspective** → **IDS**, and select **Image SC33** from the navigation tree options list, as shown in Figure 13-21.

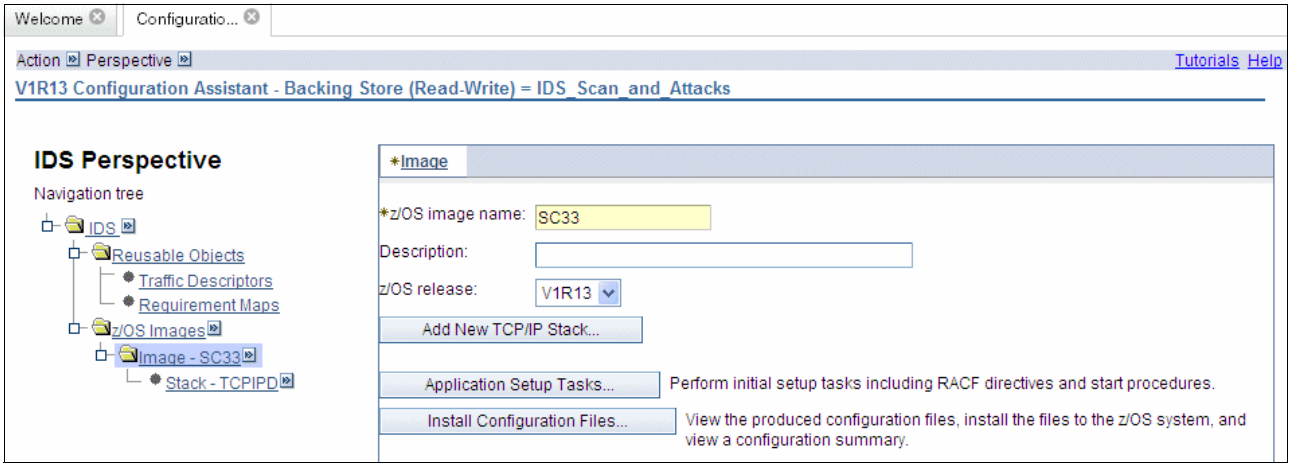


Figure 13-21 Installing IDS Configuration (Part 1)

2. From the panel, click **Install Configuration FIles** and select **IDS Policy**.
3. Specify the filename on the z/OS system where you want to install the IDS configuration files as shown in Figure 13-22, and click **Select Action** → **Install**.

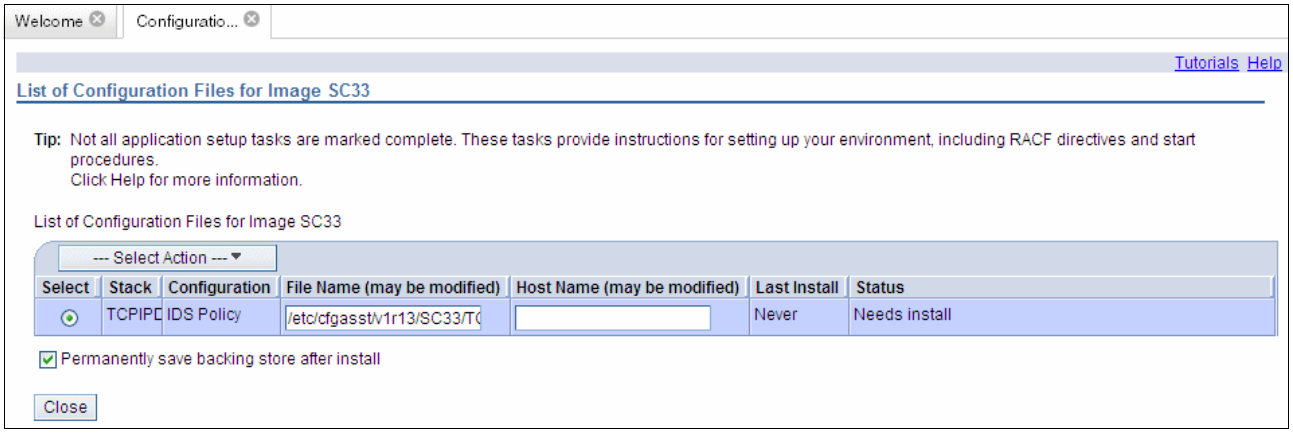


Figure 13-22 Installing IDS Configuration (Part 2)

4. In the FTP Configuration File window, enter the host name, the FTP port (usually 21), and a user ID and its password. In addition, enter the file name and location of the new IDS policy. This file name and location should be the same file mentioned in the TCP/IP stack configuration file that is used by policy agent. Click **Go**.

Figure 13-23 shows the outcome.

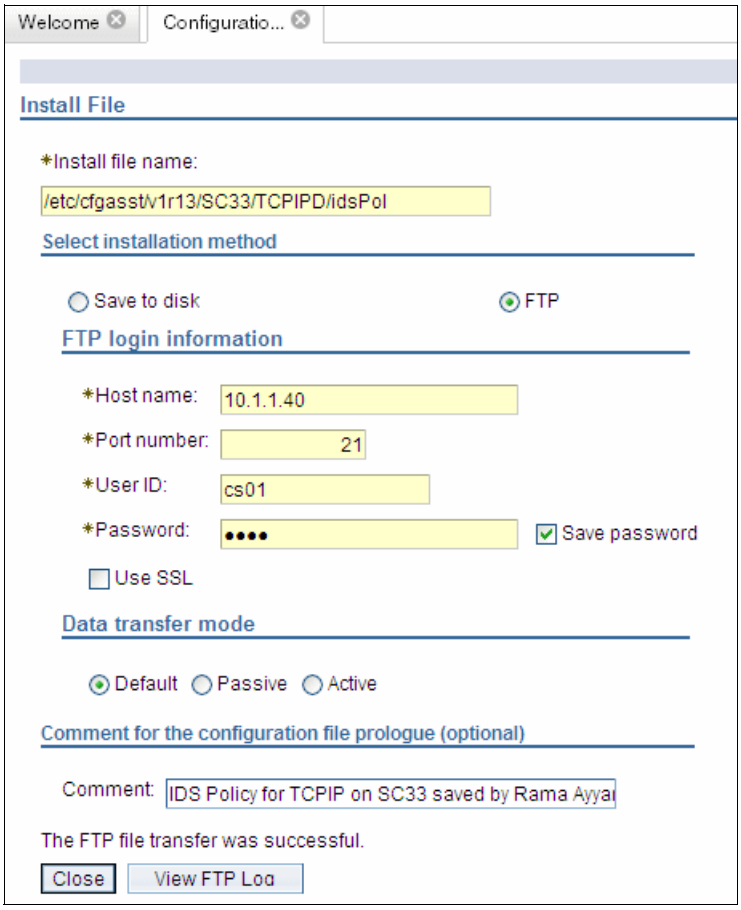


Figure 13-23 Installing IDS Configuration (Part 3)

5. After you receive the following message, refresh PAGENT.

   ```
   The FTP transfer was successful
   ```

6. Issue the following console command:

   ```
   MODIFY PAGENT,REFRESH
   ```

   The following log messages are generated:

   ```
   EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
   EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPA : IDS
   ```

7. Issue the **pasearch -n** command to make sure the policy is loaded. The results are shown in Example 13-3.

Example 13-3 Verify that the IDS policy is loaded into PAGENT

```
CS01 @ SC33:/u/cs01>pasearch -n

TCP/IP pasearch CS V1R13                Image Name: TCPIPD
  Date:                 07/19/2011      Time:  14:38:09
IDS Instance Id:      1311100301

policyRule:             Echo
  Ids Action:             Echo
```

```
policyRule:             IPv4Option
  Ids Action:             IPv4Option

policyRule:             IPv4OutboundRaw
  Ids Action:             IPv4OutboundRaw

policyRule:             IPv4Protocol
  Ids Action:             IPv4Protocol

policyRule:             IPv6DestinationOptions
  Ids Action:             IPv6DestinationOptions

policyRule:             IPv6HopByHop
  Ids Action:             IPv6HopByHop

policyRule:             IPv6NextHeader
policyRule:             IPv6NextHeader
  Ids Action:             IPv6NextHeader

policyRule:             IPv6OutboundRaw
  Ids Action:             IPv6OutboundRaw

policyRule:             DataHiding
  Ids Action:             DataHiding

................................................
................................................

policyRule:             Flood
  Ids Action:             Flood

policyRule:             GlobalTCPStall
  Ids Action:             GlobalTCPStall

policyRule:             ICMPRedirect
policyRule:             ICMPRedirect
  Ids Action:             ICMPRedirect

policyRule:             IPv4Fragmentation
  Ids Action:             IPv4Fragmentation

policyRule:             MalformedPacket
  Ids Action:             MalformedPacket

policyRule:             TcpQueueSize
  Ids Action:             TcpQueueSize
```

Figure 13-24 on page 613 shows part of the output of the Display IDS command. Note that the scan protection is on.

```
D TCPIP,TCPIPD,N,IDS
EZD0101I NETSTAT CS V1R13 TCPIPD 994
INTRUSION DETECTION SERVICES SUMMARY:
SCAN DETECTION:
  GLOBRULENAME:   SCANGLOBAL
  ICMPRULENAME:   ICMP~1
  ICMPV6RULENAME: *NONE*
  TOTDETECTED:  0           DETCURRPLC: 0
  DETCURRINT:   0           INTERVAL:   30
  SRCIPSTRKD:   1           STRGLEV:    00000M
ATTACK DETECTION:
  MALFORMED PACKETS
    PLCRULENAME: MALFORMEDPACKET
    TOTDETECTED: 0           DETCURRPLC: 0
    DETCURRINT:  0           INTERVAL:   60
  OUTBOUND IPV4 RAW RESTRICTIONS
    PLCRULENAME: IPV4OUTBOUNDRAW
    TOTDETECTED: 0           DETCURRPLC: 0
    DETCURRINT:  0           INTERVAL:   60
  RESTRICTED IPV4 PROTOCOLS
    PLCRULENAME: IPV4PROTOCOL
    TOTDETECTED: 0           DETCURRPLC: 0
    DETCURRINT:  0           INTERVAL:   60
  RESTRICTED IPV4 OPTIONS
    PLCRULENAME: IPV4OPTION
    TOTDETECTED: 0           DETCURRPLC: 0
    DETCURRINT:  0           INTERVAL:   60
```

Figure 13-24 Display of D TCPIP,TCPIPD,N,IDS command

13.4 Sample displays

This section shows what the reports from the attack detection will look like.

13.4.1 IDS Support to detect IPv6 attacks

Use the **NETSTAT IDS -k** report to see some of the IPv6 attack types as shown in Figure 13-25.

```
Attack Detection:
. . .
   Data Hiding
     PlcRuleName: AttackDataHiding-rule
     TotDetected: 8           DetCurrPlc: 0
     DetCurrInt:  0           Interval:   0
. . .
   OutBound IPv6 RAW Restrictions
     PlcRuleName: AttackOutboundv6Raw-rule
     TotDetected: 0           DetCurrPlc: 0
     DetCurrInt:  0           Interval:   0
   Restricted IPv6 Next Headers
     PlcRuleName: AttackNextHdr-rule
     TotDetected: 30          DetCurrPlc: 4
     DetCurrInt:  0           Interval:   0
   Restricted IPv6 Destination Options
     PlcRuleName: AttackDestOpts-rule
     TotDetected: 15          DetCurrPlc: 2
     DetCurrInt:  0           Interval:   0
   Restricted IPv6 Hop-by-Hop Options
     PlcRuleName: AttackHopOpts-rule
     TotDetected: 3           DetCurrPlc: 1
     DetCurrInt:  0           Interval:   0
```

Figure 13-25 IPv6 attack types

Figure 13-26 shows another use of **NETSTAT IDS -k**, displaying both IPv4 and IPv6 addresses.

```
Intrusion Detection Services TCP Port List:
 . . .
 TcpListeningSocket: 2001:db8::9:67:115:66..21
   ScStat: C  ScRuleName: ids-rule7
   TrStat: C  TrRuleName: ids-rule1
   TrPortInst: Y  TrCorr: 0           MxApp: 1           MxHst: 2
   SynFlood:   N  ConnFlood: N
 Intrusion Detection Services UDP Port List:
 . . .
 UdpDestSocket: 2001:db8::9:67:115:78..911
   ScStat: C  ScRuleName: ids-rule7
   TrStat: C  TrRuleName: *NONE*
   TrCorr: 0           Discarded: 0
```

Figure 13-26 Display showing both IPv4 and IPv6 addresses

13.4.2 Port scan

Run a ports scan on the z/OS image IP address. then run the NETSTAT IDS command to get an IDS summary of scan, attack, and traffic regulation detected. See Example 13-4.

Tip: 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, see *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Example 13-4 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
```

The Netstat IDS/-k report provides information about IDS policy and activity. The Scan Detection section of the report displays the name of the active scan global rule and the active ICMP scan rule as shown in Figure 13-27.

```
NETSTAT IDS
 MVS TCP/IP NETSTAT CS V1R13       TCPIP Name: TCPCS           11:51:44
 Intrusion Detection Services Summary:
 Scan Detection:
   GlobRuleName:   ScanGlobal-rule

   IcmpRuleName:   ScanEventIcmp-rule
   Icmpv6RuleName: ScanEventIcmpv6-rule
   TotDetected:  0           DetCurrPlc: 0
   DetCurrInt:   0           Interval:   60
   SrcIPsTrkd:   0           StrgLev:    00000
```

Figure 13-27 Netstat IDS/-k report showing IDS Summary

13.4.3 Additional information about NetView and z/OS IDS

NetView z/OS provides management support for z/OS Communications Server IDS. With it, the following tasks are possible:

- 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 email notifications to security administrators (including running trmdstat and attaching the output to the email).
- Issue predefined commands.

IDSAUTO is a set of NetView REXX CLISTs and automation table entries that automates IDS messages and performs notifications and reporting through emails. You can download these CLISTs from the following website:

http://www.ibm.com/support/docview.wss?uid=swg24001743

14

IP defensive filtering

The IP defensive filtering function of z/OS Communications Server provides a mechanism to introduce IP filters quickly into a TCP/IP stack to thwart an attack that is in progress.

IP security filtering policies require a GUI or manual update of the policy agent policy files if a new attack is detected. However, IP defensive filtering can be implemented quickly with a command on the running system. The defensive filtering command introduces a temporary protective action to counteract any new attacks on a TCP/IP stack in contrast with IP security filters, which are intended for permanent protective actions in a TCP/IP stack.

We discuss the following topics in this chapter.

Section	Topic
14.1, “Overview of defensive filtering” on page 618	Comparing IP Filtering and defensive filtering
14.2, “Basic concepts” on page 620	Basic architecture for defensive filtering
14.3, “Implementing defensive filtering” on page 626	Steps to implement defensive filtering
14.4, “Additional information” on page 640	Further references to understand the defensive filtering mechanism

14.1 Overview of defensive filtering

For a better understanding of defensive filtering, we first contrast IP defensive filtering with IP security filtering.

Filters are rules that are 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, and time. Thus, IP filtering enables a z/OS system to classify any IP packet from a network interface and to take specific action according to a predefined set of rules. An administrator can configure IP filtering to permit or deny any given network packet into or out of a z/OS system with an IP filtering rule.

Two types of IP filtering rules exist:

- IP *security* filtering, where an administrator defines a long-term and usually permanent policy that is then loaded into a TCP/IP stack by the policy agent procedure
- IP *defensive* filtering, where an administrator executes an **`ipsec`** command to install a temporary defensive filter into a TCP/IP stack

Both types of IP filtering provide packet filtering and logging.

Figure 14-1 illustrates the basics of IP security filtering architecture that we discuss in Chapter 7, “IP filtering” on page 217 and Chapter 8, “IP Security” on page 245.

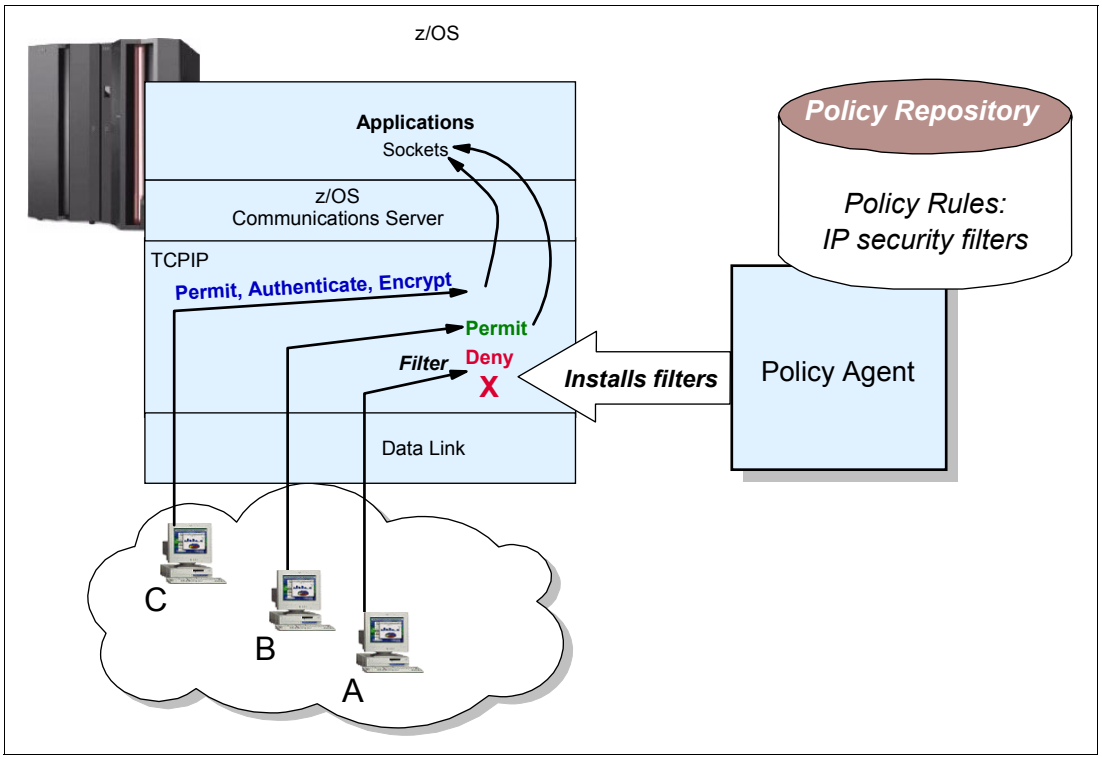


Figure 14-1 IP security filtering architecture

Figure 14-1 shows that the z/OS Communications Server policy agent reads IP security filter rules from the security policy repository and installs the security filter rules into the TCP/IP stack. The policies themselves are built with the help of a GUI or with a text editor. Also, as shown in the figure, the rules affect the traffic from terminal A, B, and C differently. The traffic

from terminal A is denied. The traffic from Terminal B is permitted. The traffic from Terminal C is permitted, and is also authenticated and encrypted using IPSec technology. Therefore, with IP security filters, traffic can be permitted, denied, or permitted with IPSec. With TRMD running, messages about IP filter activity are read and then logged into the SYSLOG daemon log.

You can use the z/OS UNIX **ipsec** command to manage and monitor the IP security filtering and IPSec VPN policies.

Figure 14-2 shows how IP defensive filtering relates to IP security filtering.

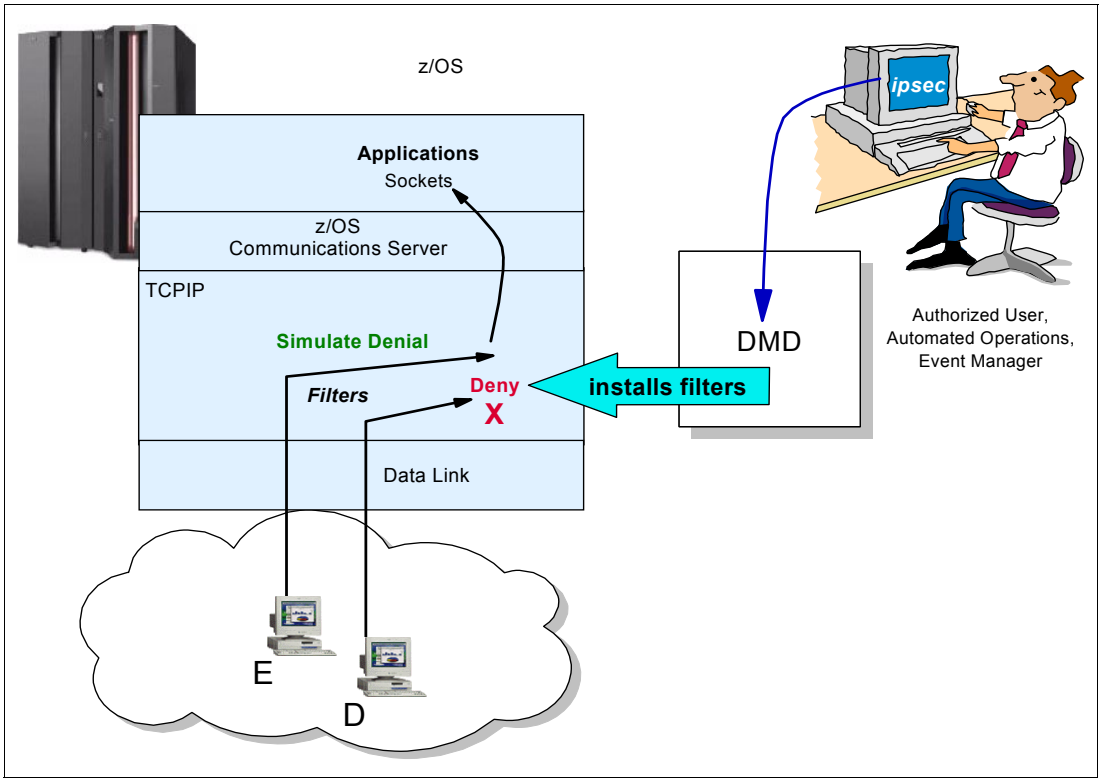


Figure 14-2 IP defensive filtering architecture

Figure 14-2 illustrates the IP defensive filtering architecture. When required, an authorized user or an authorized automated operations procedure can issue the z/OS UNIX **ipsec** command to a daemon named the *Defense Manager Daemon* (DMD). This daemon installs defensive filter coded on the **ipsec** command into the TCP/IP stack. The **ipsec** command can even be issued from a remote security and event management console. As the name implies, IP defensive filters are intended to defend the TCP/IP stack. Thus, blocking traffic is really the only purpose of these filters, unlike the IP security filters that can deny, permit, or permit with IPSec.

Despite the defensive nature of the IP defensive filters, these filters can operate in two modes. Note in Figure 14-2 how the traffic from Terminal D is denied. With TRMD running the denial is logged in SYSLOG daemon. Alternatively, the IP defensive filter action for the traffic from Terminal E only simulates a denial of traffic. The denial simulation sends messages to the TCP/IP stack, which are logged in SYSLOG daemon if TRMD is running, but the traffic itself is not blocked. You can then examine the log to determine what effect the defensive filter might have had on traffic had the action been to explicitly deny traffic.

The simulate mode for an IP defensive filter serves several purposes. It is helpful during testing to permit packets without actually disrupting traffic that matches a filter pattern. For example, you might want to verify in the logged message that the traffic affected by the IP defensive filter is actually that which you intended in the **ipsec** command syntax. The simulate mode is also helpful if you suspect that certain traffic flows on your system are in fact intrusive and you want to document through the SYSLOG daemon messages the suspicious traffic pattern while you conduct further investigation.

14.2 Basic concepts

Time is of the essence when an attack is sensed either through automation on the z/OS system itself, through alert operator observation of system messages, or through reports from external security information and event managers. Although a reactive response to the threats is possible through the generation of a new IP security filtering policy for deployment by z/OS Communications Server policy agent, this response can be less effective and less timely because of the IP filtering policy creation mechanism.

In such a scenario, a quick and easy way to introduce filtering policies into the system is needed. Best is a method to react to automation messages and, in turn, use automation to introduce policies into the system without the need for operator intervention. This automation is where defensive filtering plays an important role in securing the z/OS system from attack.

A second defensive filtering role involves providing a long-term response to a long-term or frequently recurring threat. A short-term event needs a short-term response. Policies built for policy agent installation into a stack are permanent unless a policy is deleted from the configuration files.

Although you can address a short-term event through an IP security filter policy, eliminating that policy when it is no longer needed is somewhat cumbersome in that you need to revise or delete the policy from the policy configuration file and then UPDATE or REFRESH the policy agent. In contrast, IP defensive filtering is a short-term response to an immediate threat or attack. It is implemented through the z/OS UNIX **ipsec** command with an expiration time and does not become a permanent part of the policy agent IP security policy. If the installed defensive filter needs to survive past its expiration time, you can extend the filter's lifetime through a command.

With a command-based approach to installing filters, either an operator can sign on to the system under attack and install the defensive filter, or a remote event manager can execute such a command. This approach allows an external security information and event manager with a wider view than a single TCP/IP stack or z/OS image to issue a command to this stack as part of a coordinated effort to protect the network.

Tip: If you determine that a particular event has become a recurring event for which protection requires a permanent policy, you can update the policy agent files for IP security filtering with the conditions and actions that are part of defensive filtering.

Defensive filtering includes the following major components:

- The Defense Manager Daemon (DMD)
- The stack's IP security filtering policies
- The z/OS UNIX **ipsec** command

Figure 14-2 on page 619 provided a high-level overview of IP defensive filtering. Figure 14-3 provides more detail.

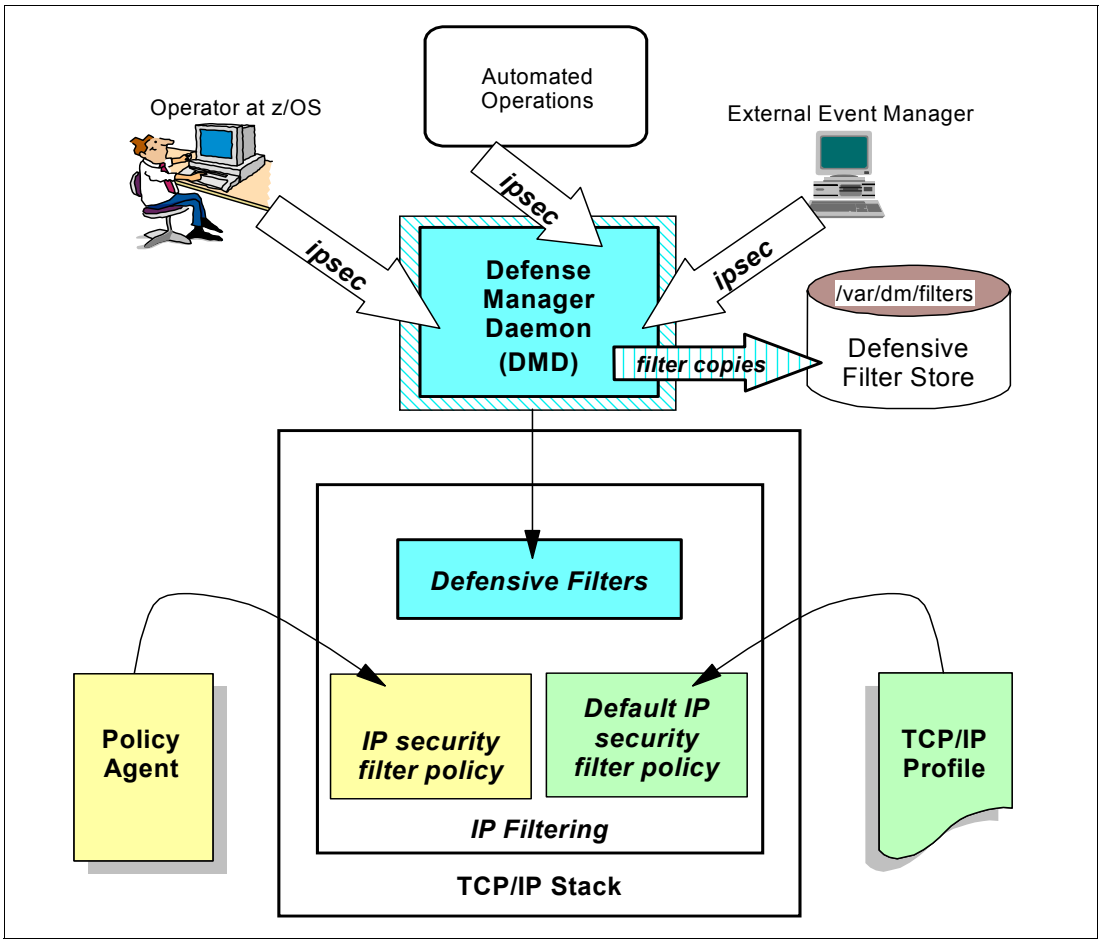


Figure 14-3 Role of defensive filtering and the Defense Manager Daemon in IP Filtering

Figure 14-3 illustrates that defensive filters are processed by the stack's existing IP filtering component. IP filters can be installed using any of the following methods:

- Policy agent installs IP security filters that are configured in the policy agent repository.
- Default IP security filters are configured in the TCP/IP profile and installed when the stack is started or when a VARY OBEY command is processed. When IP security is enabled by coding `IPSECURITY` on the `IPCONFIG` statement, one of these policies will be in effect.
- You issue an **`ipsec`** command, which is intercepted and acted upon by the DMD.

The DMD installs the defensive filters into the stack by using **`ioctl`** and writes a copy of them in binary format to its persistent defensive filtering storage. With this binary copy of the filters in a persistent store (`/var/dm/filters`), the defined filters can be applied again against the TCP/IP stacks, even if the stacks themselves or the DMD have been shut down and reinitialized.

When defensive filters are installed, the defensive filters are prioritized ahead of the installed IP security filters and checked first before proceeding to the filters installed by the policy agent or through processing of the TCP/IP profile. As each defensive filter is installed, it is prioritized ahead of the previously defined defensive filters. If a series of defensive filters are installed in the sequence Filter1, Filter2, Filter3, they are searched in the order Filter3, Filter2, Filter1.

14.2.1 Filter types

Two types of defensive filters are available:

- Global filters

  Global filters can be implemented to install filters into all TCP/IP stacks of a Common INET (CINET) environment. The invocation of **ipsec** for a global filter specifies the **-G** option and does not use a TCP/IP stack name. Without the **-G** option, the request is for a stack-specific filter.

- Stack-specific filters

  Stack-specific filters are installed in the default TCP/IP stack, unless the **-p <*stackname*>** option is specified on the **ipsec** command. For example, **-p TCPIPE** causes the defensive filter to be installed in the stack named `TCPIPE`.

If neither the **-G** option nor the **-p <*stackname*>** option accompanies the **ipsec** command, the request is sent to the default stack.

If multiple stacks in a CINET configuration all require the same filters, then one **ipsec** command causes DMD to install each filter into all the running stacks. If yet another stack is initialized in the same z/OS image, the global filter defined in the earlier **ipsec** command is retrieved automatically by the DMD from the persistent store and applied to the newly started stack.

Global filters can be installed in an INET system, but they then behave as stack-specific filters. Because global filters do not require the TCP/IP stack name in their invocation, an environment with automated operations can choose to request global filters to avoid the need for specifying a stack name. Alternatively, if individual stacks—even in a CINET environment—require separate filter rules, the **ipsec** command requests that the DMD install stack-specific filters.

Tip: The DMD can support a maximum of 10 concurrent **ipsec** command connections. Remote management of defensive filters using a Network Security Services (NSS) server is not supported.

14.2.2 Format of the ipsec command

You can invoke several parameters and options with the **ipsec** command. The administrator that executes this command requires RACF authorization for it and special RACF authorizations if it is used with defensive filtering. We describe these RACF authorizations in 8.5.4, “Restricting the use of the ipsec command” on page 267 and in 14.3.2, “Defining SAF (RACF) authorizations for defensive filtering” on page 627.

Only three of the main options apply specifically to the defensive filtering mechanism:

F Add, update, delete, or display a defensive filter
N Specifies a defensive filter’s name
G Indicates that the filter is a global filter

If you add a defensive filter (**ipsec add -F**), you can specify many combinations of subordinate options to define the traffic characteristics for the new defensive filter. These options are similar to those used to define IP security filter rules:

- **srcip**: source IP address
- **destip**: destination IP address
- **prot**: protocol
- **srcport**: source port
- **destport**: destination port
- **type**: ICMP type
- **code**: ICMP code
- **dir**: inbound or outbound
- **routing**: local, routed, or either

Together with the traffic characteristics, you can specify the following *operational characteristics* when defining a defensive filter:

- Lifetime (in minutes)
- Log (yes or no)
- Mode block (discard the packet)
- Mode simulate (test or simulate the effect of a block filter by issuing message EZD1722I to SYSLOGD)

Finally, you can also use the following additional options with the **ipsec -F** command:

N	To provide a defensive filter *name*
G	To designate this filter as a *global filter*
p <*stackname*>	To designate this filter as a *stack-specific filter*

You can read more about the various **ipsec** options in *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Prior to illustrating how to implement defensive filtering, we describe various examples of the **ipsec -F** command to show how to use the many options.

The ipsec -F add command

Example 14-1 shows the **ipsec -F add** command.

Example 14-1 Examples of ipsec -F add executions to implement defensive filtering

```
ipsec -F add srcip 192.168.1.0/24 dir inbound lifetime 30 mode block -p TCPIPE
-N Block_malformed    1

ipsec -F add srcip 192.168.1.0/24 prot tcp destport 21 dir inbound lifetime 5
-G -N G_Block_local_FTP    2
```

In this example, the numbers correspond to the following information:

1. Shows how to add a stack-specific defensive filter to block all inbound traffic from the network 192.168.1.0/24 for a period of 30 minutes. Note the name of the filter is `Block_malformed`. The name is a unique character string that is used to refer to the filter in a later command. When you create the filter, you can give it a meaningful name or a random unique string of characters. The name has no bearing on the type of traffic that is blocked.
2. Shows how to add a defensive filter to block only inbound traffic to use FTP to copy (control port 21) from the network 192.168.1.0/24. This is a global filter, and lasts for only

five minutes. You might introduce this filter to test whether it truly is the FTP traffic from that network that might be causing over-utilization of the system resources. After five minutes the filter is deleted, and you can reassess the conditions in the network. Notice the name for this filter is `G_Block_local_FTP`. In our testing, we employed a convention of prefacing the name with the letter *G* to indicate a global filter. The defensive filter is a global filter not because of the name assigned, but because the **-G** option was used with the **ipsec** command. Mode is not specified on this variation of the command, which means that the command will use the default mode of `block`.

Naming conventions: Global filters and stack-specific filters share the same defensive filter namespace. A filter name cannot be used for both a global filter and a stack-specific filter. Consider using a unique naming convention for global filters.

The ipsec -F update command

Example 14-2 shows an example of the **ipsec -F update** command and option. With the **update** option, you can change the *operational characteristics* of a defensive filter.

Example 14-2 Examples of ipsec -F update executions to implement defensive filtering

```
ipsec –F update log no –N G_filter1 –G    1

ipsec –F update lifetime 20 –N filter2 –p TCPIPE    2
```

In this example, the numbers correspond to the following information:

1. Shows how the **update** option disables logging of this global filter.
2. Shows how the **update** option resets the lifetime of the filter installed in the stack TCPIPE to 20 minutes. Note that this filter might have been created as a stack-specific filter or as a global filter. In the latter case, the command execution causes only the copy for the named stack to be updated.

The ipsec -F delete command

Example 14-3 shows an example of the **ipsec -F delete** command and option. If neither **–G** nor **–p** is specified, the update request is directed to the default stack.

Example 14-3 Examples of ipsec -F delete executions to implement defensive filtering

```
ipsec –F delete –N all –G    1

ipsec –F delete –N all    2

ipsec –F delete –N G_filter1, filter2 –p TCPIPE    3

ipsec -F delete -N all -p TCPIPE   4
```

In this example, the numbers correspond to the following information:

1. Shows how all copies of all global defensive filters in this z/OS image are to be deleted. If you have a combination of stack-specific and global filters installed, the stack-specific filters are not deleted.
2. Shows how all defensive filters for the *default stack* are to be deleted. Note that **-p <*stackname*>** is not specified.
3. Illustrates the deletion of multiple filters on the TCPIPE stack only. Note that if a copy of a global filter is deleted from a stack, it is not added back to that stack, even if the stack goes down and is brought back up.
4. Shows how to delete all defensive filters from a specified stack. Both stack-specific and global filters installed in stack TCPIPE will be deleted.

The ipsec -F display command

We can **display** the defensive filters by showing the defensive filters on a stack-by-stack basis or by showing the global filters alone as shown in Example 14-4.

Example 14-4 Examples of ipsec -F display executions for displaying defensive filtering

```
ipsec -F display -p TCPIPE -N G_Block_local_FTP    1

ipsec -F display -G -N G_Block_local_FTP    2

ipsec -f display -c current   3
```

In this example, the numbers correspond to the following information:

1. Illustrates a display from the TCPIPE stack of what we presume, based on our naming convention, to be a global filter.
2. Requests a display of a global filter from the DMD.
3. Requests a display of all IP filters—both defensive filters and IP security filters—that are currently in effect for the default stack.

The ipsec -t command

Finally, we show how to use a traffic test command to display the installed IP filters—both defensive filters and IP security filters—that match a traffic pattern. In Example 14-5, the traffic test command displays the IP filters that match an inbound packet with source IP address 10.1.1.1 and destination IP address 10.2.2.2.

Example 14-5 Examples of ipsec -t execution to filter on two defensive filtering IP addresses

```
ipsec –t 10.1.1.1 10.2.2.2 0 in 0    1
```

1. Takes an input traffic pattern and returns the IP filters that match the traffic pattern. Both matching defensive filters and IP security filters are displayed. The first `0` indicates any protocol, and the final `0` specifies any security class.

14.3 Implementing defensive filtering

Using defensive filtering requires the following prerequisites:

- SYSLOG daemon must be active as described in *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*.
- TRMD must be active for each stack in the z/OS image for which defensive filtering is to be enabled, as described in Chapter 4, “Policy agent” on page 103.
- IPSec filtering must be enabled in the TCP/IP stack as described in 14.3.1, “Enabling IPSec filtering in the TCP/IP stack” on page 627 and also in the defensive filtering chapter of *z/OS Communications Server: IP Configuration Guide*, SC31-8775.
- SAF authorizations must be in place in the z/OS image as described in 14.3.2, “Defining SAF (RACF) authorizations for defensive filtering” on page 627.
- The DMD must be running as explained in 14.3.3, “Implementing the DMD procedure” on page 629.

Figure 14-4 shows the scenario with which we tested defensive filtering.

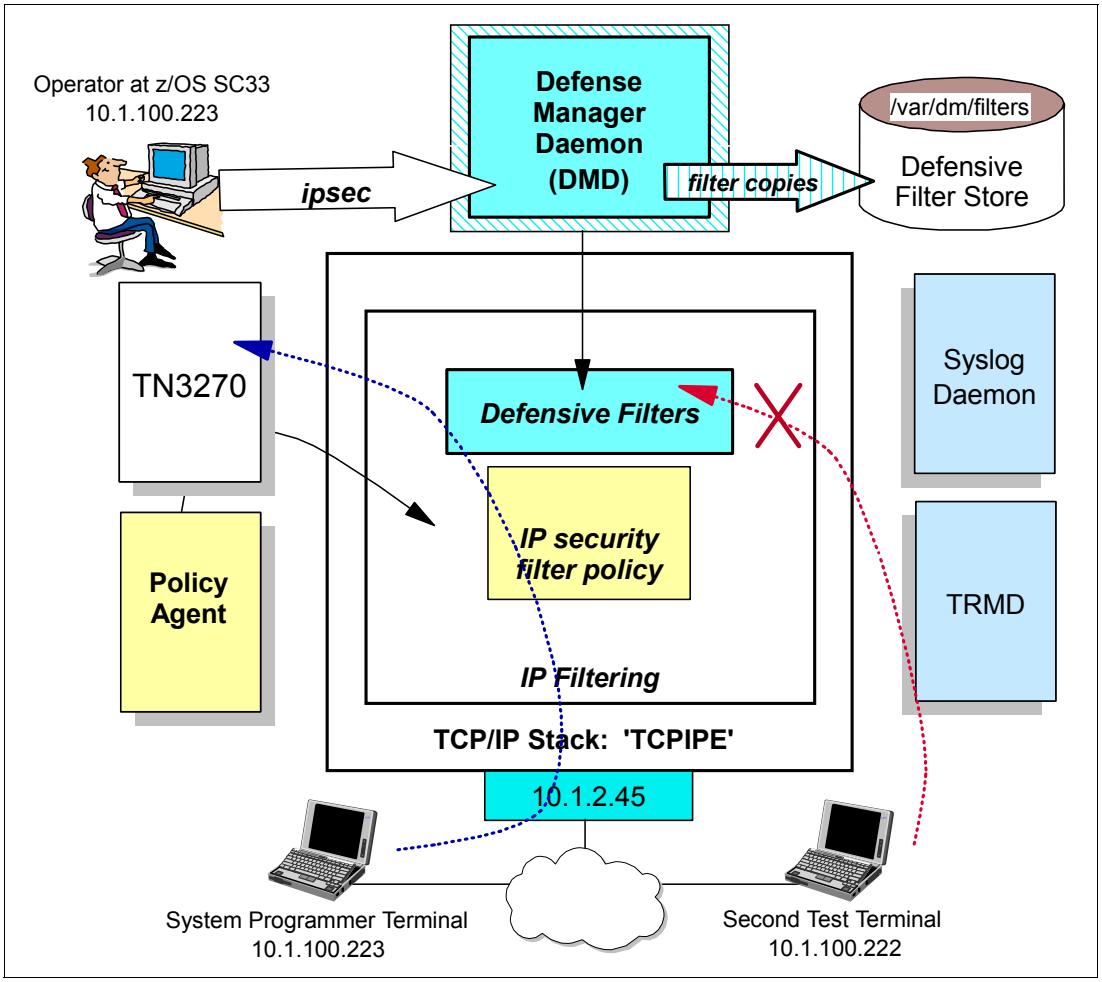


Figure 14-4 Test Scenario: Blocking inbound traffic to TN3270 with defensive filtering

In our testing, we wanted to enable defensive filtering in TCPIPE on z/OS image SC33. A stack-specific defensive filter was installed to block traffic between 10.1.100.222 and TN3270 at 10.1.2.45. Then a global filter was installed. We deleted the filters at the end of the testing.

14.3.1 Enabling IPSec filtering in the TCP/IP stack

Defensive filtering for a stack requires that IPSec filtering be enabled in that TCP/IP stack, even if it is enabled only minimally.

If you do not have an IP security policy implemented in PAGENT and still want to use defensive filtering, then you must define IPSECURITY on the IPCONFIG (or IPCONFIG6) statement of the TCP/IP profile. In addition, you must define one or more IPSECRULE (or IPSECRULE6) statements in the TCP/IP profile. With minimal enablement, as shown in Example 14-6, you might have only a single rule that allows all traffic.

Example 14-6 Minimal TCP/IP profile changes for the implementation of defensive filtering

```
...........
IPCONFIG IPSECURITY
...........
IPSEC
; Rule SourceIp DestIp Logging Prot SrcPort DestPort Routing Secclass ;
; Permit all local and routed IPv4 traffic, no logging.
IPSECR * * NOLOG PROTO * ROUTING EITHER
ENDIPSEC
...........
```

If you have already enabled an IP security policy, you do not need to change that policy to use defensive filters.

In our scenario, we had already implemented IP filtering and IPSec with a policy agent policy following the examples described in Chapter 8, “IP Security” on page 245. Therefore, we did not need to make the changes in Example 14-6.

14.3.2 Defining SAF (RACF) authorizations for defensive filtering

Defensive filtering requires authorization in the following areas:

- A user ID must be built under which the DMD authoritatively runs.
- The user IDs that are associated with administrators, with automated operations, with an external security information and event manager, and with operators need to be authorized to issue the **ipsec** command to the DMD.

You can find examples of the necessary RACF definitions in `hlq.SEZAINST(EZARACF)`. In our system, the name of the data set and member was `TCPIP.SEZAINST(EZARACF)`.

Ensuring authorization for DMD

The DMD procedure requires an OMVS segment that is associated with a user ID for initialization. A UID of 0 is not required for the OMVS segment that is associated with the DMD. However, our installation had no restrictions for running a procedure such as the DMD under a user ID with a UID of 0. Therefore, we added the user ID of DMD to the RACF database with a UID of 0, by using the commands shown in Example 14-7. We then defined the procedure name to the `STARTED` class and associated the new user ID with it.

Example 14-7 Adding a user ID with UID 0

```
ADDUSER DMD DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
RDEFINE STARTED DMD.* STDATA(USER(DMD))
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED) REFRESH
```

Non-zero UID? If your installation has restrictions that prevent you from defining the DMD user ID with a UID of 0, be aware that there are implications for the DMD directory and file definitions that we describe later in 14.3.3, “Implementing the DMD procedure” on page 629.

We then authorized the user that was associated with the DMD started task to access the SYS1.PARMLIB using the following command:

```
PERMIT SYS1.PARMLIB ID(DMD) ACCESS(READ)
```

Authorization for the ipsec command for defensive filtering

The z/OS UNIX **ipsec** command is used for several functions, including IPSec, NSS, and defensive filtering. If you have already implemented IPSec or NSS, the format of the RACF commands to create a SERVAUTH class and to permit authorization to it will look familiar to you.

You also know that the authorizations to execute the **ipsec** command can be made with wildcards or with granularity in each of the fields of the SERVAUTH class. We chose to establish granularity when we created the SERVAUTH class. Example 14-8 shows our definitions.

Example 14-8 SERVAUTH class definition and authorizations for defensive filtering

```
RDEFINE  SERVAUTH EZB.IPSECCMD.*.DMD_GLOBAL.* UACC(NONE)
PERMIT   EZB.IPSECCMD.*.DMD_GLOBAL.* CLASS(SERVAUTH) ID(SYS1) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

In Example 14-8, we use SERVAUTH profiles to control the users who are allowed to manage defensive filters. Subsequently, we permitted the RACF group to which the system administrators belong (SYS1) to this SERVAUTH class.

You must also authorize the user to work with stack-specific filters. If you have already implemented IPSec or NSS for this stack on this z/OS image, you might already have the necessary RACF authorizations in place. If not, you need to execute the commands that pertain to your configuration.

Example 14-9 shows the commands that we executed for the TCPIPE stack in system SC33. We also permitted the user group named SYS1 to perform display and control work surrounding stack-specific filters.

Example 14-9 Authorizing users in SYS1 group to display, add, update, delete stack-specific filters

```
RDEFINE  SERVAUTH EZB.IPSECCMD.SC33.TCPIPE.DISPLAY UACC(NONE)

RDEFINE  SERVAUTH EZB.IPSECCMD.SC33.TCPIPE.CONTROL UACC(NONE)

PERMIT   EZB.IPSECCMD.SC33.TCPIPE.DISPLAY CLASS(SERVAUTH) ID(SYS1) ACCESS(READ)
PERMIT   EZB.IPSECCMD.SC33.TCPIPE.CONTROL CLASS(SERVAUTH) ID(SYS1) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

14.3.3 Implementing the DMD procedure

The implementation that we describe in this section requires specific permissions to the directories and files that the DMD requires. Recall that we defined our DMD user ID as a superuser. If your installation has a non-zero user ID, see the following note.

Important: If your installation does not require superuser authority (`UID=0`) to start DMD, the DMD executable modules must be in an APF-authorized library.

In addition, the DMD user ID must be able to create, delete, read, and write files in the following directories.

- `/var/dm/`
- The directory specified on the `DefensiveFilterDirectory` statement (the default value is `/var/dm/filters`)

If the DMD user ID is not configured with `UID=0`, then make sure it has read/write/execute access to the directories.

If you have overridden the default location (`/var/dm/`) of the DMD pidfile with the use of the DMD_PIDFILE environment variable, then this non-zero UID user must be able to create and write files to that location. You might have coded this environment variable in the STDENV member identified in the DMD procedure.

The non-zero UID user ID requires execute access to the directory in which the DMD configuration file resides and read access to the configuration file itself. Because `/var/dm/` will already have the correct permissions for directory access, you might consider placing the DMD configuration file in this directory and then pointing to it with the environment variable named DMD_FILE, which you can have coded in the STDENV member identified in the DMD procedure.

You can find more information about this subject and these steps in the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Creating the DMD directories in UNIX

DMD requires access to several directories and files:

- `/var/dm/`
- `/var/dm/filters/`

We created the directories for DMD from the UNIX shell using the following commands:

```
su: switch into Superuser mode
cd /var: move to the /var directory
mkdir dm: create the dm directory
ls -al: to verify that the directory has been created
```

Example 14-10 shows the output of the **`ls -al`** command.

Example 14-10 Output of ls -al command in the /var/ directory

```
drwxr-xr-x   2 BPXROOT  SYS1         320 Feb  5  2009 .
drwxrwxrwt  20 BPXROOT  SYS1         864 Oct  6 09:38 ..
crw-rw-rw-   1 BPXROOT  SYS1       6,  0 Feb  5  2009 dmd.sock
```

However, because we defined DMD as a superuser, it already has the necessary access. If the user ID of DMD is not a superuser, follow the instructions in the earlier note about a non-zero UID or consult the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Next, move into the new `dm` directory to create the `filters` directory, which is the directory that represents the persistent store for installed defensive filters:

```
cd dm
mkdir filters
```

Again, the DMD user ID requires read/write/execute access to this directory. If the user ID of DMD is not a superuser, follow the instructions in the earlier note about a non-zero UID or consult the *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Capacity planning: Monitor the space for this directory or for the file system on which it resides. The directory needs sufficient space to support at least 1 MB of data for each TCP/IP stack, plus another 1 MB for the global filter definitions. Depending on your usage of defensive filtering, you might need to ensure that even more space is set aside.

Creating the DMD configuration file and procedure

The DMD configuration file resides by default in `/etc/security`. It establishes the logging level for the DMD, the TCP/IP stacks for which defensive filtering is enabled, the defensive filter mode for each stack, and any excluded networks or addresses for each stack. The DMD procedure reads the configuration file, and the DMD user ID must have read access to the file and execute authority to the directories under which this file resides. Therefore, you need to change to the `/etc/security` directory and copy the DMD configuration file sample into the current (security) directory as follows:

```
cd /etc/security
cp /usr/lpp/tcpip/samples/dmd.conf . (The dot represents the current directory.)
```

Example 14-11 shows how we used the **`oedit dmd.conf`** to edit the `dmd.conf` file.

Example 14-11 Configuration of /etc/security/dmd.conf

```
# IBM Communications Server for z/OS
# SMP/E distribution path: /usr/lpp/tcpip/samples/dmd.conf
#
# 5694-A01 Copyright IBM Corp. 2008.
# Licensed Materials - Property of IBM
#
# /etc/security/dmd.conf (Defense Manager daemon configuration)
#
DMConfig  1
{
# The supported levels are:
# 0 - DM_SYSLOG_LEVEL_NONE - Disable the Defense Manager daemon syslog
#      messages
# 1 - DM_SYSLOG_LEVEL_MINIMUM - Minimal Defense Manager daemon syslog
#      output
# 2 - DM_SYSLOG_LEVEL_LIFECYCLE_CLIENT - Include info about the connect
#      and disconnect of clients.
# 4 - DM_SYSLOG_LEVEL_LIFECYCLE_STACK - Include info about the cycling
#      of stacks and the installation, deletion or modification of
#      defensive filters to the stack.
# 8 - DM_SYSLOG_LEVEL_VERBOSE - Include cascaded internal error messages
```

```
#      (for IBM service)
SyslogLevel 15 1a
DefensiveFilterDirectory /var/dm/filters 1b
}
DmStackConfig TCPIPE  2
{
# Mode  Active|Simulate|Inactive  (dynamically modifiable)
  Mode SIMULATE 2a
  MaxLifetime 60 2b
  Exclude 10.1.100.223 2c
}
```

In this example, the numbers correspond to the following information:

1. The `DMConfig` statement establishes the general operating environment of the DMD.
   - For testing purposes, we set logging to `15` (as shown at 1a). The default level is `7`. The number 15 represents all four of the logging levels, which cumulatively amount to the value of 15. You can reset logging back to 7 by editing this file and issuing a MODIFY DMD,REFRESH command.
   - The persistent store for filters is identified at 1b. We maintained the default setting.
2. The `DmStackConfig` statement establishes the TCP/IP stack for which defensive filtering is enabled. We have one `DmStackConfig` statement for TCPIPE, the TCP/IP stack on z/OS SC33 with which we are testing. Your configuration might require more than one `DmStackConfig` statement.
   - We set the defensive filter mode to SIMULATE for TCPIPE (2a) so that we could test terminals to ensure that they are locked out of the system if they are not on the exclude list.
   - The maximum lifetime for any defensive filter installed for this stack will be 60 minutes (as shown at 2b). If you add a filter with the **ipsec** command with a lifetime greater than 60 minutes, this value overrides the value specified in the **ipsec** command, and the filter will have a lifetime of 60 minutes.
   - We excluded the System Programmer's terminal from any filters that might be set so that we can continue to monitor the system from at least that terminal (as shown at 2c). There is a limit of 10 `exclude` statements.

Only one copy of DMD can run in a z/OS image. DMD can be started from the UNIX shell, in which case you might want to set the jobname to something meaningful with `_BPX_JOBNAME`. You can also start DMD with JCL, using either the EZADMD program module from SEZALOAD or BPXBATCH. COMMNDxx in PARMLIB can also be the vehicle for starting DMD, as can AUTOLOG in the TCP/IP profile. However, AUTOLOG is problematic in a CINET environment and should be used only if the z/OS image is running with a single stack.

Tip: When the DMD is started using `PGM=DMD`, the STDENV DD card, if used, is passed directly to the DMD program. Language Environment does not gain access to the STDENV environment variables. As a result, any Language Environment runtime options set in the STDENV DD data set using the `_CEE_RUNOPTS` environment variable are ignored. In this case, Language Environment runtime options must be passed on the PARM parameter, and the options must be specified before any DMD options. However, the PARM parameter allows a maximum of 100 characters.

If the Language Environment runtime options plus DMD parameters that you want exceed 100 characters, consider using BPXBATCH to start the DMD. When `PGM=BPXBATCH` is used, the Language Environment variable `_CEE_RUNOPTS` can be included on the STDENV DD card to specify runtime options in excess of 100 characters long.

For more information, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

The DMD procedure finds its configuration file in one of two ways:

- If the STDENV card points to a file or a z/OS data set that contains the environment variable `DMD_FILE`, then the DMD uses the named file for its configuration information. The DMD user must have read access to this file and execute access to the directories under which the file resides.
- Otherwise, the JCL looks for the default in `/etc/security/dmd.conf`.

Example 14-12 shows the JCL that we used to run DMD.

Example 14-12 Startup JCL for DMD

```
//DMD PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZADMD
//*
//* 5694-A01 Copyright IBM Corp. 2008
//* Licensed Materials - Property of IBM
//*
//*
//DMD EXEC PGM=DMD,REGION=0K,TIME=NOLIMIT,
// PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/' a
//*
//* Provide environment variables to run with the desired
//* configuration. As an example, the data set or file specified by
//* STDENV could contain:
//*
//* DMD_FILE=/etc/security/dmd.conf   b
//* DMD_CTRACE_MEMBER=CTIDMD00 b
//* DMD_PIDFILE=/var/dm/dmd.pid b
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference.
//*
//STDENV DD DUMMY c
//* 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=*
```

We copied this JCL sample from `hlq.SEZAINST(DMD)`, which in our scenario was named `TCPIP.SEZAINST(DMD)`. We ran CINET in this z/OS image, but we wanted to associate the DMD with only one of the many stacks. Notice at **a** in the sample how we have *not* specified stack affinity as you might be accustomed to with other UNIX procedures in z/OS. Typically for stack affinity you set an environment variable named `_BPXK_SETIBMOPT_TRANSPORT=TCPIPE` (**a**) in the EXEC parameters of the JCL. DMD does not require stack affinity through this environment variable because its configuration file simplifies coding with a statement, `DmStackConfig`, that identifies the stacks for which IP defensive filtering should be active. Example 14-11 on page 630 shows the specification of this statement.

The lines of the JCL that are labeled with **b** represent what can be included as a set of Language Environment variables if you code a data set or file to represent `STDENV`. The DMD user ID must have the appropriate access to this data set or file and also to its members. We used default settings for all these variables and, therefore, `STDENV` (**c**) points to a DD card of `DUMMY`.

Data set format for STDENV: The three possible statements that can be included in a standard environment file for DMD are honored by the startup JCL if the STDENV points to a zFS (or HFS) file. For performance reasons, zFS is preferred. If STDENV points to a z/OS data set, the RECFM must be variable or variable blocked, not fixed blocked. Otherwise, the contents of the data set are ignored.

14.3.4 Operations and verification with defensive filtering

We tested our scenario, shown in Figure 14-4 on page 626, with one terminal that was permitted to access TN3270 and another terminal that was not. The diagram does not depict the three stacks running in this z/OS system (TCPIP, TCPIPD, and TCPIPE). Only TCPIPE is enabled for defensive filtering. That is, TCPIPE is the only stack defined to the DMD with a `DmStackConfig` statement in the DMD configuration file.

To verify the operations of defensive filtering with DMD, we ensured that SYSLOG daemon and TRMD were running. We installed IP policies in the TCPIPE stack and then started the DMD procedure from the MVS console as shown in Example 14-13.

Example 14-13 DMD initialization

```
S DMD
$HASP100 DMD      ON STCINRDR
IEF695I START DMD      WITH JOBNAME DMD      IS ASSIGNED TO USER DMD
   , GROUP OMVSGRP
$HASP373 DMD      STARTED
EZD1609I DEFENSE MANAGER DAEMON RELEASE CS V1R13 SERVICE LEVEL
CS110601 CREATED ON Jun  1 2011
EZD1610I THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN
IEE252I MEMBER CTIDMD00 FOUND IN SYS1.IBM.PARMLIB
EZD1622I DEFENSE MANAGER DAEMON CONFIGURATION PROCESSING IS COMPLETE
USING FILE /etc/security/dmd.conf
EZD1611I THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS
COMPLETED
```

Example 14-14 shows the status of the running DMD.

Example 14-14 Display of DMD executing in SC33

```
F DMD,DISPLAY
 Defense Manager Configuration Settings
   SyslogLevel              = 15
   DefensiveFilterDirectory = /var/dm/filters
   DM Config for TCP/IP stack TCPIPE
     Mode        = Simulate
     MaxLifetime = 60
     Exclude     10.1.100.223
```

In Example 14-14, the configuration file settings from Example 14-11 on page 630 are reflected in the running procedure.

We entered the UNIX shell to install a defensive filter using the **ipsec** command as shown in Example 14-15.

Example 14-15 Adding a defensive filter to block all Telnet connections

```
CSO3 @ SC33:/SC33/tmp/syslog>ipsec -F add srcip all destip all prot tcp destport
23 -N Block_TELNET23_onTCPIPE -p TCPIPE    a

EZD1540I Defensive filter Block_TELNET23_onTCPIPE successfully added to stack
TCPIPE b
```

We added a defensive filter named Block_TELNET23_onTCPIPE with the command at a. This defensive filter blocks inbound traffic. The message EZD1540I confirms that the filter was added successfully (b). When we changed to the /var/dm/filters directory, we found the following entry to indicate an active filter for the TCPIPE stack:

```
-rwxr-x--x   1 SYSPROG  SYS1        2048 Aug  1 14:20 active.TCPIPE
```

Tip: This file is one of several binary files that are managed by the DMD. Do *not* modify such files manually. If you edit or alter this file, it will become corrupt and marked as *untrusted* in the first field of the binary file name. You might also encounter a similar type of binary file marked *inactive* instead of *active*. This situation can occur if the DMD configuration file's mode for that stack is changed to inactive or if the MODIFY FORCE_INACTIVE command is used to render a stack inactive to IP defensive filtering.

From the UNIX shell, we displayed all active filters for the TCP/IP stack with the following command:

```
ipsec -p TCPIPE -F display
```

Example 14-16 shows the output for this command.

Example 14-16 Displaying stack-specific defensive filter

```
CS V1R13 ipsec  Stack Name: TCPIPE  Mon Aug  1 14:21:57 2011
Primary:  Defensive Filt1 Function: Display           Format:   Detail
Source:   Stack           Scope:    n/a               TotAvail: 47
Logging:  n/a             Predecap: n/a               DVIPSec:  n/a
NatKeepAlive:  20         FIPS140:  n/a
Defensive Mode: Simulate             2
Exclusion Address: 10.1.100.223/24   3
```

```
FilterName:                  Block_TELNET23_onTCPIPE      4
FilterNameExtension:         1
GroupName:                   n/a
LocalStartActionName:        n/a
VpnActionName:               n/a
TunnelID:                    n/a
Type:                        Defensive                    5
DefensiveType:               Stack                        6
State:                       Active
Action:                      Defensive Block              7
Scope:                       Local
Direction:                   Inbound
OnDemand:                    n/a
SecurityClass:               0
Logging:                     All                          8
Protocol:                    TCP(6)
ICMPType:                    n/a
ICMPTypeGranularity:         n/a
ICMPCode:                    n/a
ICMPCodeGranularity:         n/a
OSPFType:                    n/a
TCPQualifier:                None
ProtocolGranularity:         n/a
SourceAddress:               0.0.0.0
SourceAddressPrefix:         0
SourceAddressRange:          n/a
SourceAddressGranularity:    n/a
SourcePort:                  All
SourcePortRange:             n/a
SourcePortGranularity:       n/a
DestAddress:                 0.0.0.0
DestAddressPrefix:           0
DestAddressRange:            n/a
DestAddressGranularity:      n/a
DestPort:                    23
DestPortRange:               n/a
DestPortGranularity:         n/a
OrigRmtConnPort:             n/a
RmtIDPayload:                n/a
RmtUdpEncapPort:             n/a
CreateTime:                  2011/08/01 14:51:54
UpdateTime:                  2011/08/01 14:51:54
DiscardAction:               Silent
MIPv6Type:                   n/a
MIPv6TypeGranularity:        n/a
TypeRange:                   n/a
CodeRange:                   n/a
RemoteIdentityType:          n/a
RemoteIdentity:              n/a
FragmentsOnly:               No
FilterMatches:               0
LifetimeExpires:             2011/08/01 15:21:54
AssociatedStackCount:        n/a
```

In this example, the numbers correspond to the following information:

1. Indicates this is a defensive filter display (**-F**)
2. A stack's defensive filter mode of Simulate (not in Active mode)
3. Indicates the address of the system administrator's workstation, which is excluded from defensive filtering actions
4. The name of the filter
5. Indicates this is a defensive filter
6. Indicates a stack-specific filter and not a global filter
7. The type of action (block and discard packet if not in simulate mode)
8. All logging enabled for this filter

We logged in to Telnet on port 23 from a terminal that was not on the exclude list (10.1.100.222) and were permitted entry into Telnet. However, the log messages in the syslog daemon log (`local4.log` in Example 14-17) showed the message `EZD1722I`, indicating that, had the defensive filter mode for this stack been active instead of simulate, our connect would not have succeeded (a).

Example 14-17 Local4.log messages when Defensive Filter is in Simulate mode

```
EZD1723I Defensive filter added: 08/01/2011 18:51:54.75 filter
rule=Block_TELNET23_onTCPIPE ..............................

EZD1722I Packet would have been denied by defensive filter: 08/01/2011 19:01:02.91
filter rule= Block_TELNET23_onTCPIPE ext= 1 sipaddr= 10.1.100.222 dipaddr=
192.168.1.40 proto= tcp(6) sport= 2509 dport= 23 -= Interface= 192.168.2.42 (I)
secclass= 255 dest= local len= 40 ifcname= OSA20E0I fragment= N
```

We changed our `dmd.conf` file to indicate a mode of ACTIVE for stack TCPIPE and caused DMD to re-read its configuration file by executing the following command from z/OS:

```
MODIFY DMD,REFRESH
```

Then we log in to Telnet in again from 10.1.100.222 and from 10.1.100.223. This time our connection from 10.1.100.222 was blocked as shown in Example 14-18, and our connection from 10.1.100.223 was permitted.

Example 14-18 Packet denied with DMD

```
EZD1721I Packet denied by defensive filter: 08/01/2011 19:29:54.59 filter rule=
Block_TELNET23_onTCPIPE ext= 1 sipaddr= 10.1.100.222 dipaddr= 192.168.1.40 proto=
tcp(6) sport= 2512 dport= 23 -= Interface= 192.168.2.42 (I) secclass= 255 dest=
local len= 48 ifcname= OSA20E0I fragment= N
```

We added a global filter next to determine what effect this would have on the other stacks that were not enabled for IP defensive filtering and not enabled for IPSECURITY. Example 14-19 shows that the global defensive filter was added successfully to the stacks shown in Example 14-20 on page 637.

Example 14-19 Adding a global defensive filter when only one stack is eligible

```
CS02 @ SC33:/u/cs02>ipsec -F add srcip all destip all prot tcp destport 23 -G -N
G_Block_TELNET23_onTCPIPE
EZD1541I Global defensive filter G_Block_TELNET23_onTCPIPE successfully added
```

Example 14-20 Stacks eligible and ineligible for defensive filtering

```
CSO2 @ SC33:/u/csO2>ipsec -f display -c current -p TCPIP
EZDO861I Stack TCPIP is not configured for IPSECURITY  1
CSO2 @ SC33:/u/csO2>ipsec -f display -c current -p TCPIPD
EZDO861I Stack TCPIPD is not configured for IPSECURITY 1

CSO2 @ SC33:/u/csO2>ipsec -f display -c current -p TCPIPE > TCPEfilter 2
```

In this example, the numbers correspond to the following information:

1. The TCPIP and TCPIPD stacks are not eligible for defensive filtering because the DMD configuration file has not enabled them with a `DmStackConfig` statement. In addition, IPSECURITY is not enabled on those stacks.
2. We know that the TCPIPE stack is eligible for defensive filtering and that many filters are defined to it through a policy agent. Therefore, we pipe the output of the command to a file named `TCPEfilter` as shown in Example 14-21.

Example 14-21 Display of all filters on TCPIPE (policy agent IP filters and defensive filters)

```
CS V1R13 ipsec  Stack Name: TCPIPE  Mon Aug  1 15:35:47 2011
Primary:  Filter          Function: Display           Format:   Detail
Source:   Stack Profile   Scope:    Current           TotAvail: 12    1
Logging:  On              Predecap: Off               DVIPSec:  No
NatKeepAlive:  20         FIPS140:  No
Defensive Mode: Active
Exclusion Address: 10.1.100.223

FilterName:                   G_Block_TELNET23_onTCPIPE               2
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     n/a
Type:                         Defensive
DefensiveType:                Global
State:                        Active
Action:                       Defensive Block
.................
********************************************************************************
FilterName:                   Block_TELNET23_onTCPIPE                 3
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     n/a
Type:                         Defensive
DefensiveType:                Stack
State:                        Active
Action:                       Defensive Block
.................
********************************************************************************
FilterName:                   SYSDEFAULTRULE.1                        4
FilterNameExtension:          1
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                n/a
TunnelID:                     0x00
Type:                         Generic
DefensiveType:                n/a
..................
```

In this example, the numbers correspond to the following information:

1. 12 filters in total are installed for this stack.
2. The first filter to be searched or compared against activity is the last Defensive Filter we activated (the global filter).
3. The second filter to be searched or compared against activity is the stack-specific filter.
4. The remaining filters to be searched are the 10 remaining filters installed with a policy agent.

Next, we deleted all the defensive filters installed in stack TCPIPE using the following command:

```
ipsec -F delete -N all -p TCPIPE
```

This action deletes the defensive filters installed in the TCPIPE stack and also deletes the stack-specific binary defensive filter store (`active.TCPIPE`) in `/var/dm/filters`.

After this deletion, a repeated display of the existing filters for stack TCPIPE shows that only the 10 filters installed in the stack by policy agent are now available. The two defensive filters installed on TCPIPE are deleted.

Next, we deleted all the global defensive filters with the commands shown in Example 14-22.

Example 14-22 Deletion of global defensive filters

```
CS02 @ SC33:/u/cs02>ipsec -F delete -N all -G
EZD1543I All global defensive filters successfully deleted
```

With this deletion, the binary global defensive filter store (`active._globals_`) in `/var/dm/filters/` is deleted from the z/OS system.

You might be wondering if the global defensive filter that was installed in the DMD would be successful if the TCPIPE stack was not running. Yes, the installation would be successful. If the DMD is active, recall that it creates a global filter store in `/var/dm/filters/` from which it retrieves the temporary defensive filter definitions. If a stack-specific filter is installed while the stack for which it is designated is not yet active, the same concept applies. DMD creates a stack-specific filter store that is used after the pertinent stack is initialized.

Another method you can use to delete defensive filters from a stack is to edit the `DmStackConfig` statement in the DMD configuration file and subsequently issue the MODIFY DMD,REFRESH command to re-read the file. For example, by specifying `DmStackConfig` TCPIPE mode INACTIVE in the configuration file, the refresh of DMD makes the TCPIPE stack ineligible for defensive filters. Alternatively, you might want to execute the following command to make the TCPIPE stack ineligible for defensive filtering:

```
MODIFY DMD,FORCE_INACTIVE,TCPIPE
```

Finally, we concluded our testing by stopping the DMD with the appropriate P command (Example 14-23).

Example 14-23 Stopping the DMD

```
P DMD
EZD1636I THE DEFENSE MANAGER DAEMON RECEIVED THE STOP COMMAND
EZD1604I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS BEGUN
EZD1605I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED
$HASP395 DMD      ENDED
```

Concepts of filter deletion

DMD does not necessarily delete the filter from the persistent stores immediately upon deletion of the filter. Rather DMD performs sweeps on a number of occasions to clean up expired filters. DMD never propagates an expired filter to a stack.

DMD is the sole manager of the persistent store files. Therefore, if DMD is down, expired filters in the persistent stores are not cleaned up. This is intentional because filters are supposed to persist even if DMD terminates.

For example, assume that there is a defensive filter in stack A and a persistent file for stack A. If DMD goes down, the filter in stack A and in the persistent file is unaffected. If DMD reinitializes immediately, it reads the persistent file and is still synchronized with stack A. If, however, DMD remains inactive for awhile and the filter expires, the stack will delete the filter. At this point, the filter is no longer in effect. In fact, the filter is not even evident in an **`ipsec -F`** display. When DMD is restarted, it sweeps the persistent store file, deletes any expired filters, and loads the active filters into DMD memory. Any persistent store file that contains no active filters, and is thus empty, is deleted.

14.3.5 Conclusions

Although our tests with defensive filtering used commands executed by a system programmer, in reality the most useful implementation of defensive filtering revolves around system automation. Such an implementation requires coordination among the technical specialists who are monitoring the system with the external security and event monitors and the System z networking specialists to write the CLISTS with variables that can be replaced with values derived from the monitoring techniques.

However, even with no system automation, the defensive filtering can usually be more efficiently implemented in an emergency than can a policy created with the IBM Configuration Assistant. For example, policy changes to long-term, permanent policies might require an extended change management review that precludes their usage in an emergency situation. In contrast, IP defensive filtering is temporary, introducing only short-term filters, and requires only the execution of a command by an authorized user.

Tip: If you already have IP security filtering in place, have a DMD environment tested and ready to use for IP defensive filtering when an urgent situation arises. Identify the users who are authorized to issue the **`ipsec`** command for defensive filtering and that you ensure that they understand where to find a text file or script with working samples of defensive filtering commands that they can copy and paste quickly into the UNIX shell for emergency execution. If you have already identified a known pattern of attacks that occur periodically, you can automate the **`ipsec`** invocation to respond to those attacks.

Can DMD be permanently active

If IP security filtering is already active in your environment, the DMD procedure can run continually without adding any significant overhead to the z/OS image. If you want to make IP defensive filtering available to all stacks, regardless of whether permanent IP security filtering policies apply to them, consider that such stacks incur the overhead of the TCP/IP IPSECRule processing to permit all traffic. Your risk tolerance can help weigh the benefits of enabling IP security filtering in such stacks if the enablement is there only for an eventual introduction of an IP defensive filter.

14.4 Additional information

Consult the following resources for more detailed information about defensive filtering:

- *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

15

Policy-based routing

Policy-based routing (PBR) 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 PBR tables are defined to the stack for optimized routing.

We discuss the following topics in this chapter.

Section	Topic
15.1, “Policy-based routing concept” on page 642	Basic concepts of PBR
15.2, “Routing policy” on page 643	Configuration options for routing policies
15.3, “Implementing policy-based routing” on page 646	Implementation procedure for PBR

15.1 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 can, however, have a requirement to separate different types of traffic sent to that destination. For example, you might 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 separate 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 can contain static routes, dynamic routes, and 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 can 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 eight 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 can 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.

Tip: 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 PBR 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 might not select the best stack for the connection because it only has information from the main routing table of each stack.

15.2 Routing policy

The PBR policy is configured in a policy agent flat-file. It includes the following definitions.

- Routing rules
- Routing actions
- Routing tables

You can use z/OSMF Configuration Assistant to generate the PBR configuration flat file, or you can code it directly to the flat file. You might want to use z/OSMF Configuration Assistant to avoid syntax errors.

Tip: 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 PBR 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

Tip: 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).

  Tip: 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, only 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 the following information:

- 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 PBR 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, define the name using all upper case letters to avoid confusion because z/OS commands display the routing table name in all upper case letters regardless. The existing main route table is called EZBMAIN. The names EZBMAIN and ALL (in upper, lower, and mixed case) are reserved.

Tip: 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 PBR 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, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775, and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Tip: 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.

15.3 Implementing policy-based routing

In this section we show two scenarios for implementing policy-based routing (PBR):

- 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 PBR 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.

z/OSMF Configuration Assistant 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 PBR 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 4.2, “Implementing PAGENT on z/OS” on page 109.

2. Configure the PBR policies:
   - Policy-based routing using jobname, protocol, and destination IP address
   - Policy-based routing using protocol and port numbers

15.3.1 Policy-based routing using jobname, protocol, and destination IP address

In this scenario, we implement PBR to force all TN3270 traffic to use a specific OSA link. On the main routing table, OSA20E0I and OSA20C0I are preferred links, but we configure OSA2080I as the most preferred link and OSA 20A0I as the backup link for TN3270 traffic. TCPIPD in Figure 15-1 illustrates our PBR scenario.

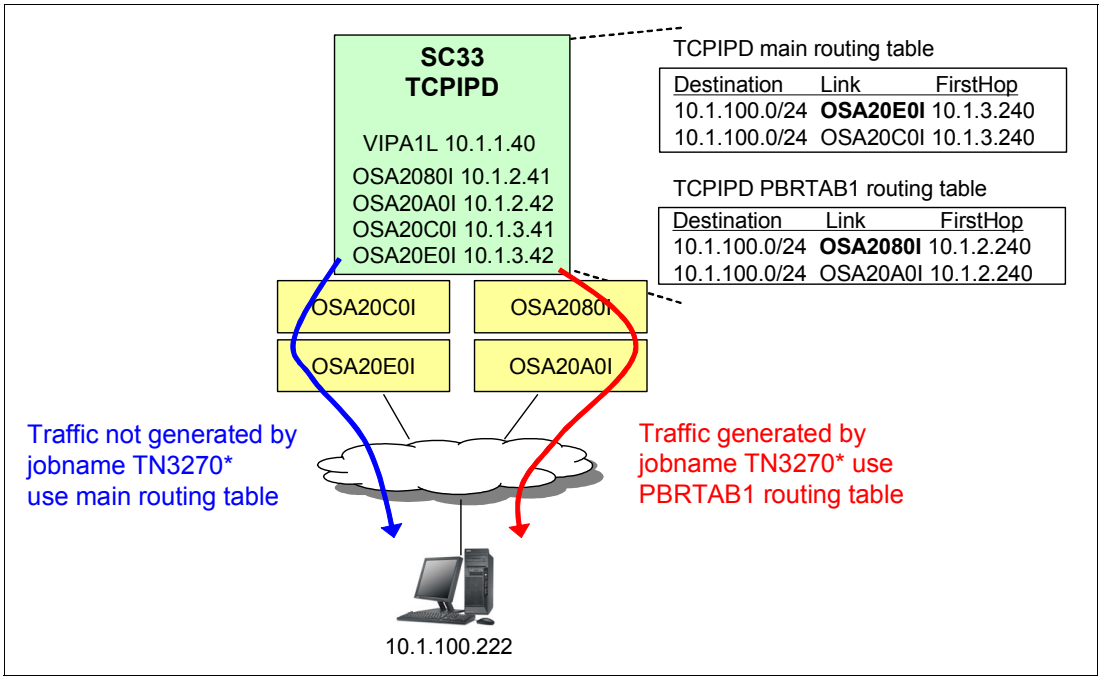


Figure 15-1 PBR using jobname, protocol, and destination IP address

Implementation tasks

We implemented PBR by using the following steps:

1. Configure the PBR policy with jobname, protocol, and destination IP address.
2. Save 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

Complete the following steps:

1. Start the z/OSMF Configuration Assistant. In the Main Perspective panel, click the **Add a New z/OS image** option.
2. Type the name of the z/OS image. In our example, we entered `SC33`, as shown in Figure 15-2. Click **OK**. A panel might appear to ask if we want to define the TCP/IP stack now. In this case, click **Yes** to proceed.

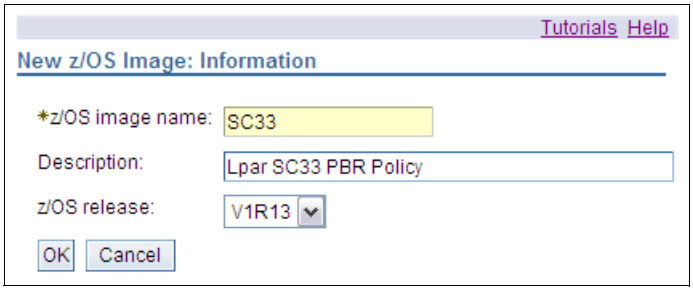


Figure 15-2 New z/OS Image panel: Adding the SC33 z/OS image

3. The New TCP/IP Stack: Information panel opens. Type the name of the TCP/IP stack, in our case, `TCPIPD` as shown in Figure 15-3.

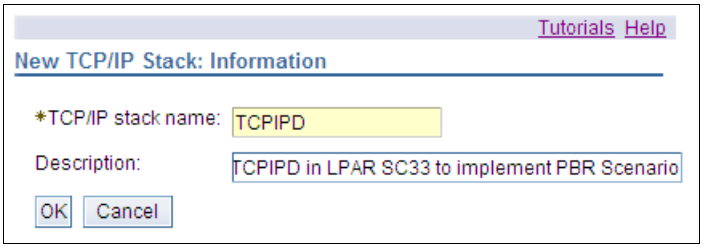


Figure 15-3 New TCP/IP Stack panel: Adding the TCPIPD stack

4. You return to the Main Perspective panel shown in Figure 15-4. Select **PBR** in the z/OS Communications Server technologies list and click **Select Action** → **Enable**. The PBR technology line shows a status of `Incomplete` in red. Click **Select Action** → **Configure**.

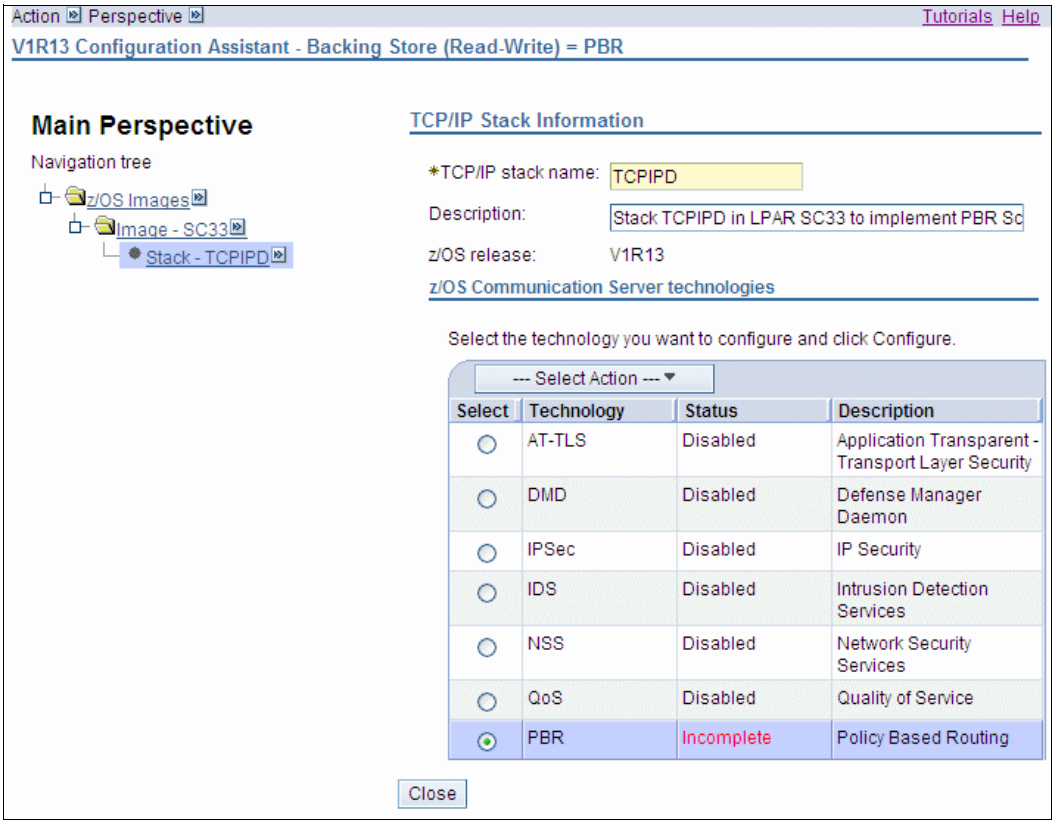


Figure 15-4 Main Perspective panel: Enabling the PBR policy

5. If a panel opens that prompts you to define the connectivity rule now, click **Yes**. If you click **No** or you are not prompted, the PBR Perspective panel is displayed (Figure 15-5). Notice the stack in the left panel shows `Incomplete Stack` because the PBR policy is not yet configured. Click **Select Action** → **Add** in the Connectivity Rules tab to create a new connectivity rule.

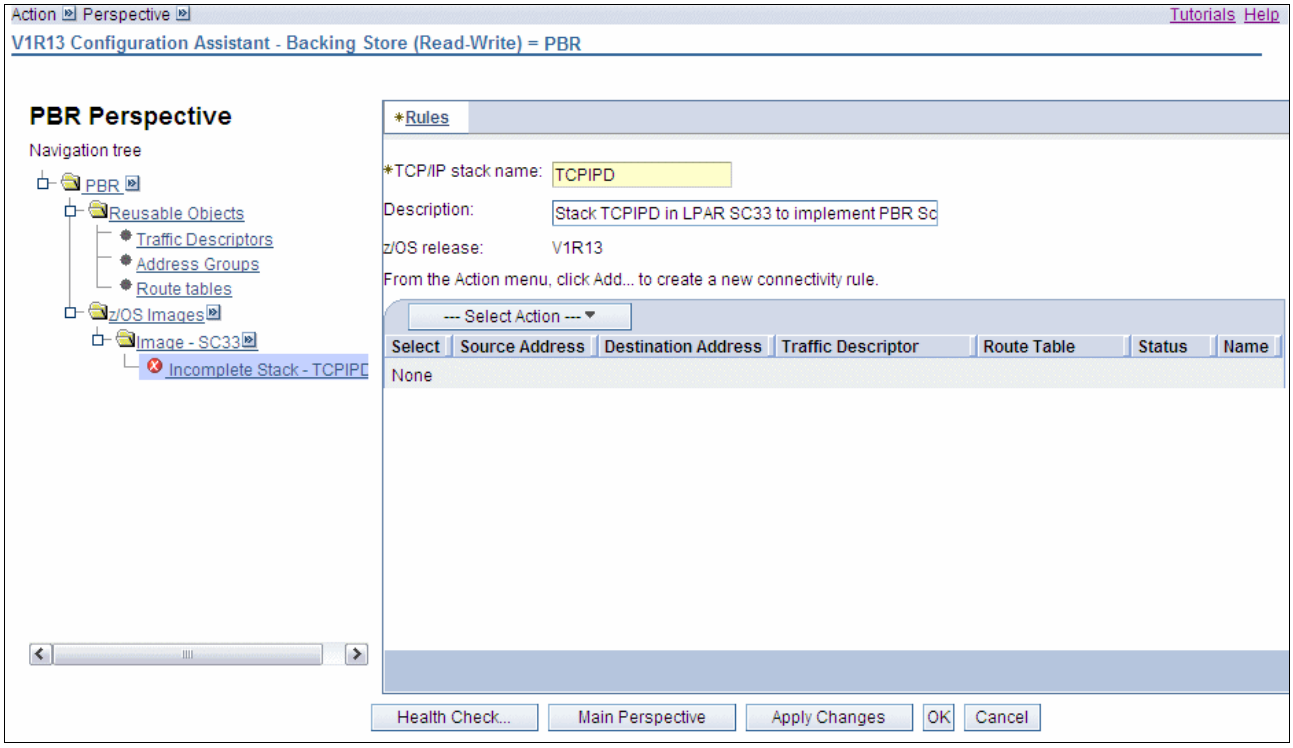


Figure 15-5 PBR Perspective panel: Adding the PBR policy

6. In the New Connectivity Rule - Welcome panel, we define the name for the connectivity rule, in our example `PBRRule1`. Click **Next**.

7. In the New Connectivity Rule - Traffic Descriptor panel (Figure 15-6), select the **Yes-Specify the traffic descriptor parameters.** Because we use TN3270 traffic in this scenario, we select protocol **TCP** and click **Add**.

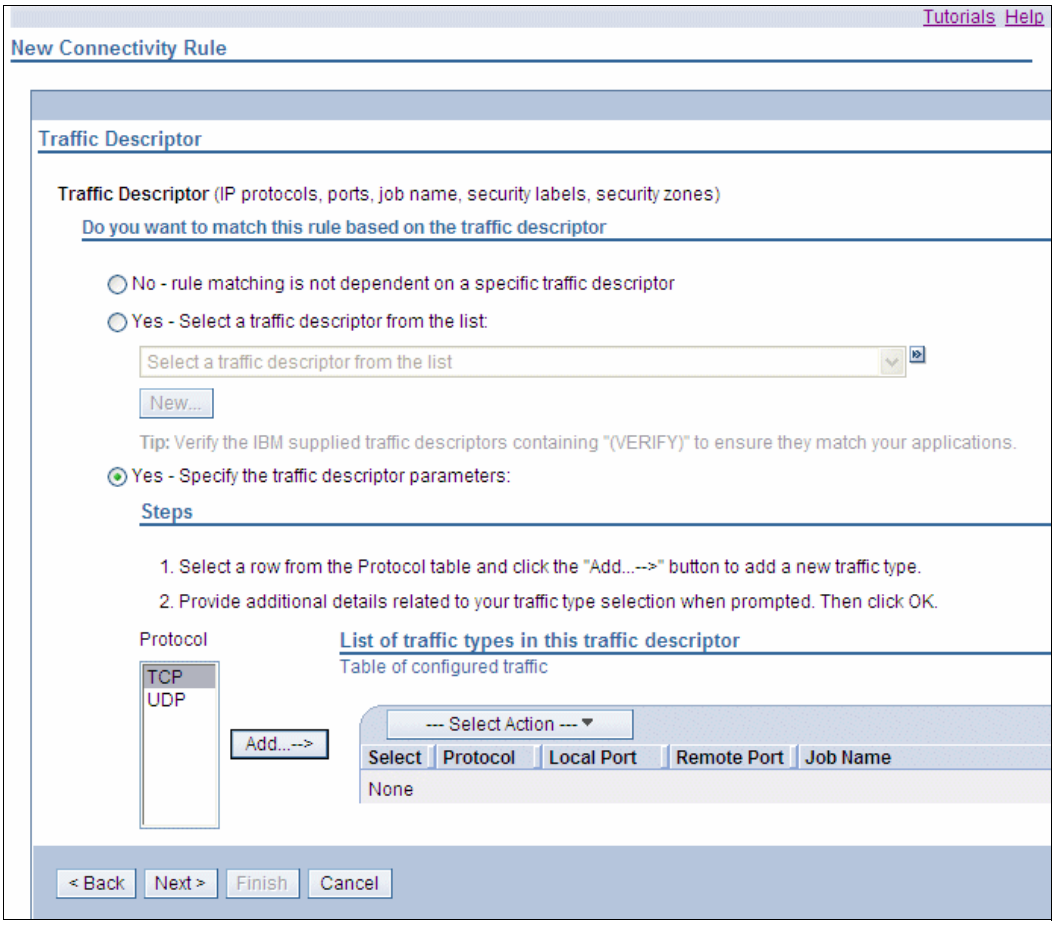


Figure 15-6 Defining the traffic description of ports and applications (Part 1)

8. In the Modify Traffic Type - TCP panel, we select all ports for both source port and destination port, and specified the jobname `TN3270*` as shown in Figure 15-7. Click **OK**. Back at the New Connectivity Rule - Traffic Descriptor panel, notice that TCP traffic with a jobname of TN3270* is added to the list of traffic types. Click **Next**.

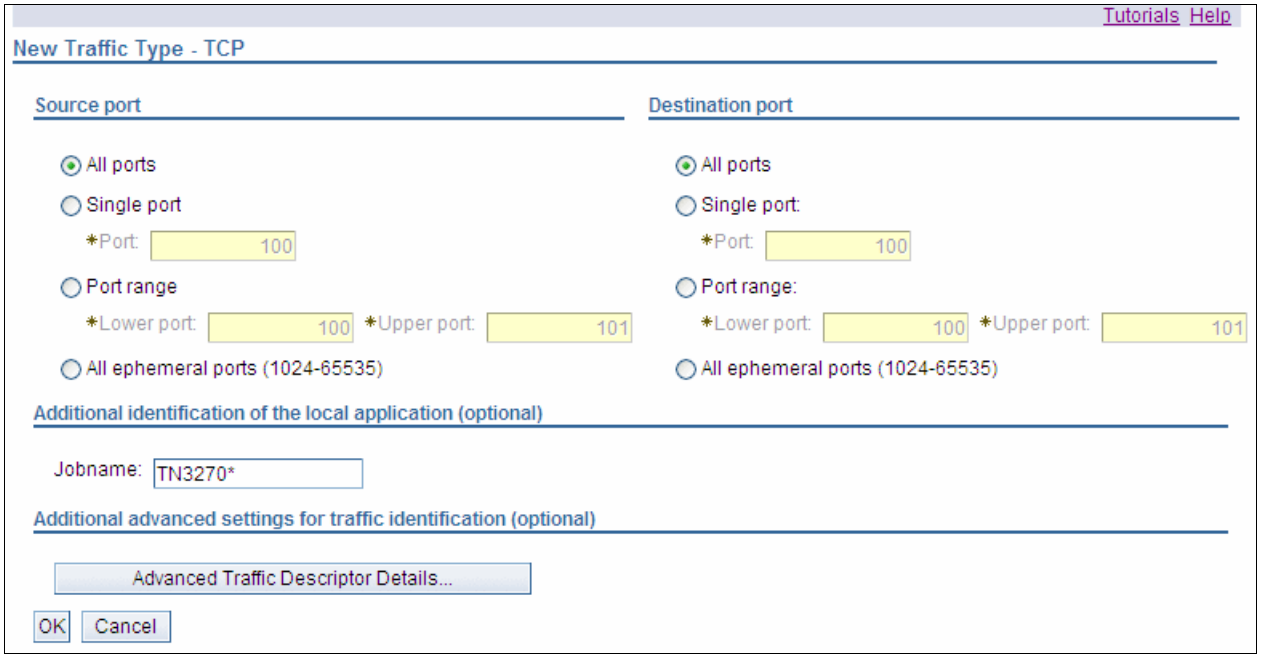


Figure 15-7 Defining the traffic description of ports and applications (Part 2)

9. In the New Connectivity Rule - Source Address panel, we select **No**, **rule matching is not dependent on the source address** because we do want to specify the source IP address as part of our traffic descriptor. Click **Next**.

10. In the New Connectivity Rule - Destination Address panel, we select **Yes - Specify the IP address**, and include the IP address of one of our workstations, 10.1.100.222, as shown in Figure 15-8. You can also specify the time period when you want to activate this routing rule by clicking **(Optional) Time Conditions** (not defined in our scenario). Click **Next**.

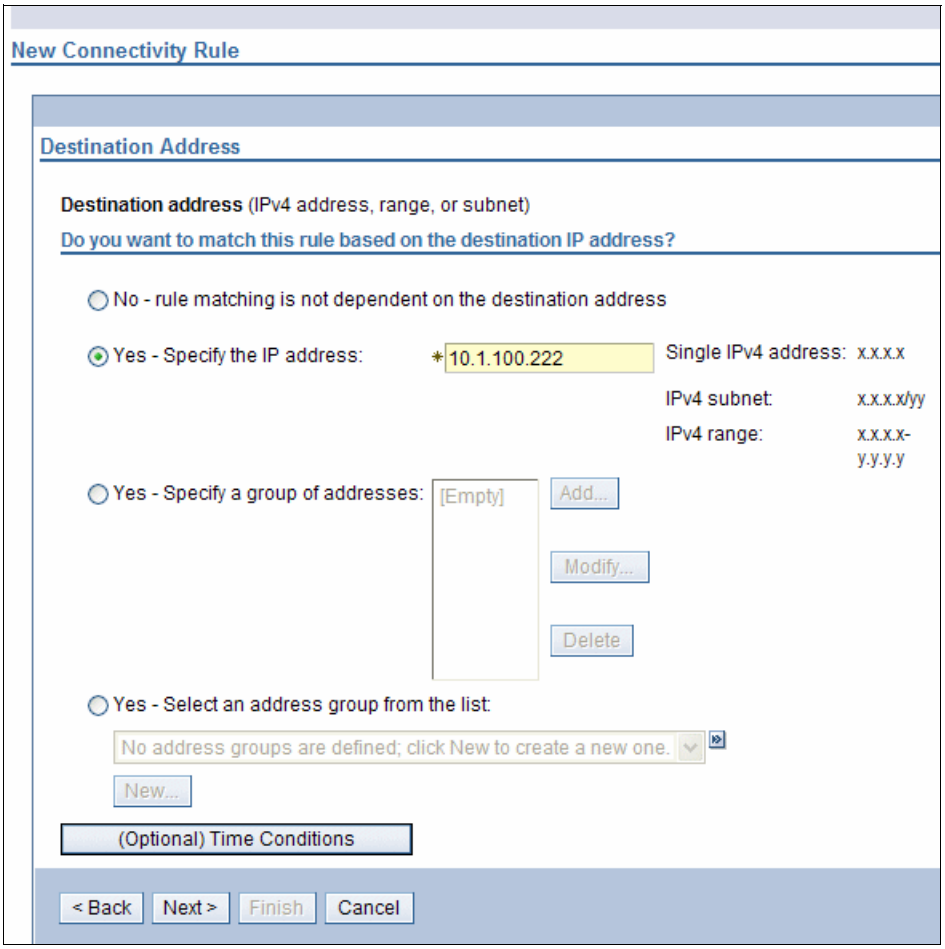


Figure 15-8 Defining the traffic description of destination IP address

11. In the New Connectivity Rule-Route Table panel, we define a new routing table and routing actions, by selecting **New** in the Primary Route Table view (Figure 15-9).

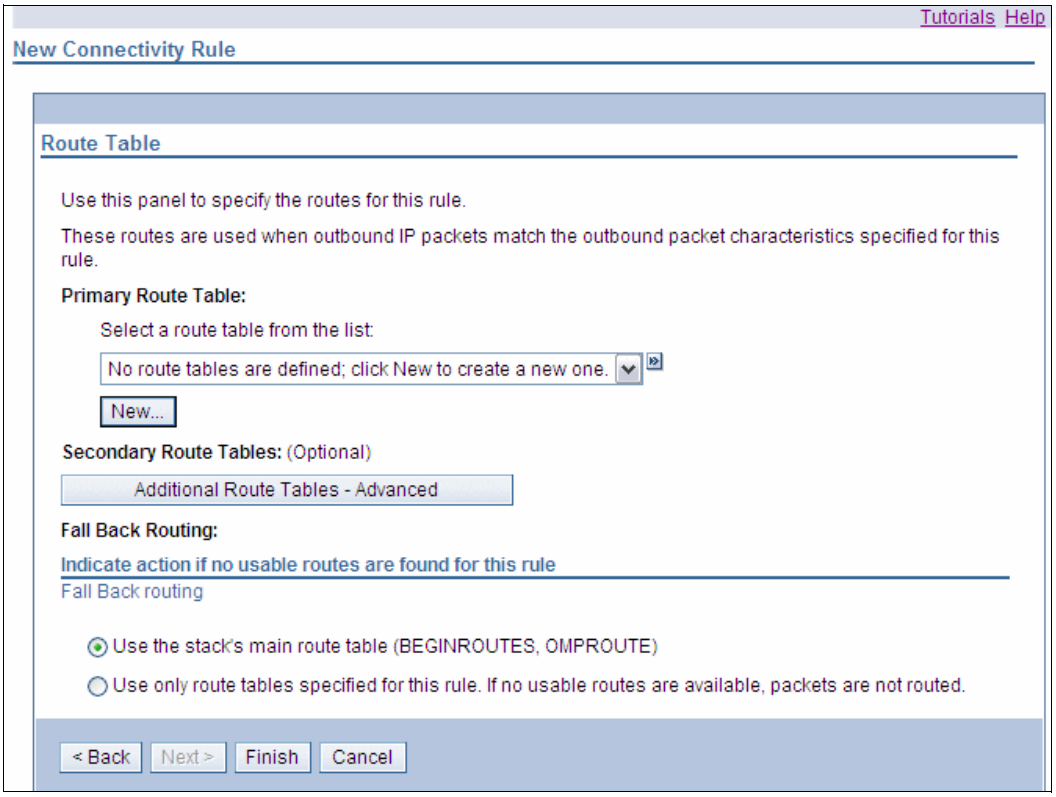


Figure 15-9 Route Table panel

12. In the New Route Table panel, we enter the name of our routing table, `PBRTAB1`, and define our routing entries as shown in Figure 15-10. In our example, we click **Select Action** → **Add** under the **Dynamic routes** section. Enter the name of the link that you want to send the traffic from (`OSA2080I` in our example), as shown in Example 15-11 on page 662. Click **OK**.

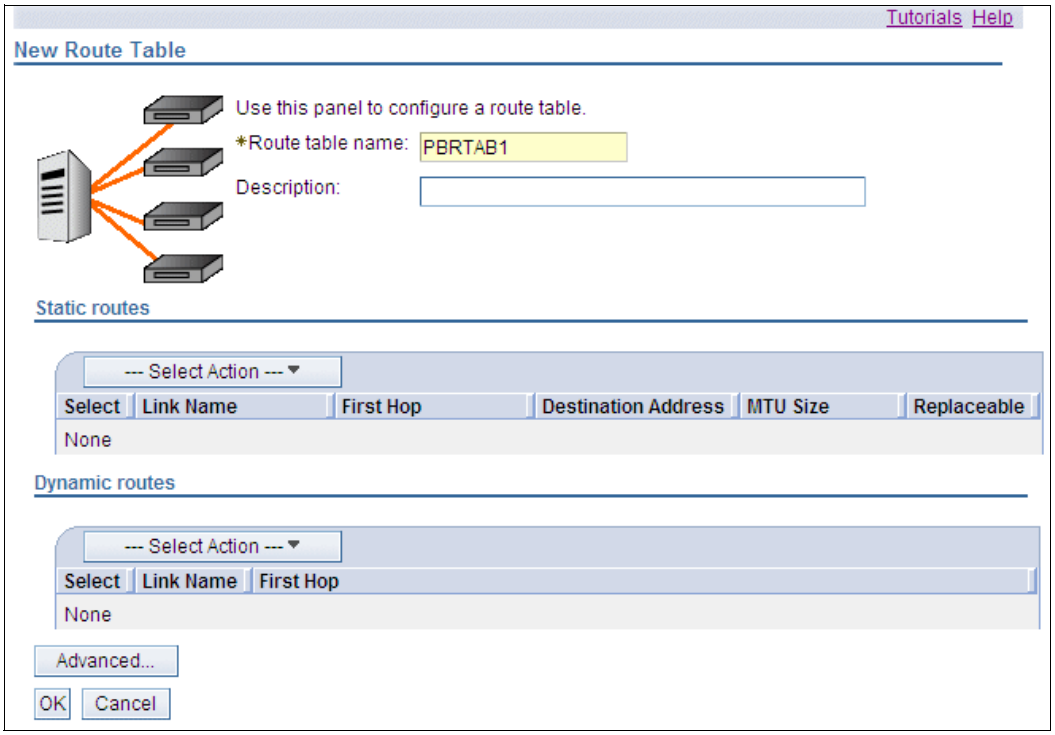


Figure 15-10 Route Table panel: Defining the routing table and routing entries (Part 1)

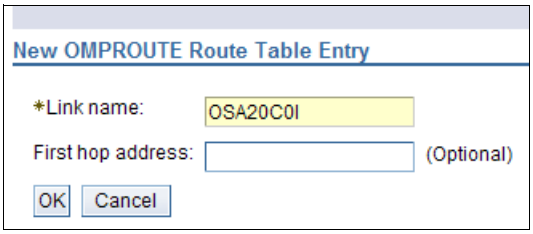


Figure 15-11 Route Table panel: Defining the routing table and routing entries (Part 2)

13. Verify that the link for dynamic routing is added to the Dynamic Routes list. We do the same to create another route table entry for `OSA20A0I`. To make OSA2080I as the preferred link over the OSA20A0I, select OSA2080I and click **Select Action** → **Move Up**. Optionally, you can also define advanced options (MULTIPATH, Path MTU update, Dynamic XCF Routes) by selecting **Advanced**. Click **OK**.

14. Back in the New Connectivity Rule - Route Table panel, make sure that the routing table that we created is selected in the Primary Routing Table list (Figure 15-12). You can define additional routing tables by selecting the **Additional Route Tables - Advanced** option. We also select 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**.

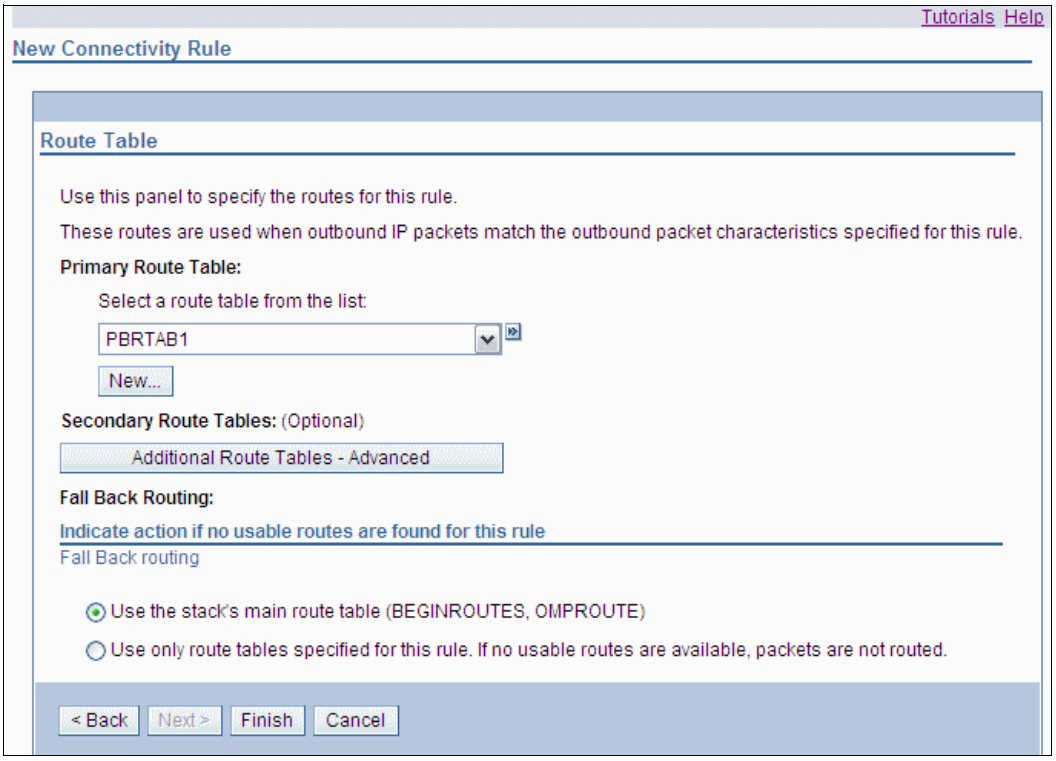


Figure 15-12 Route Table panel: Defining routing actions

15. Back in the PBR Perspective panel, click **Apply Changes**.

The definition is complete. Example 15-1 shows the flat file generated by the z/OSMF Configuration Assistant.

Example 15-1 The policy configuration file generated by 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 13
## Backing Store = PBR
## FTP History:
## 2011-08-03 21:08:57 : Save To Disk
##
## End of Configuration Assistant information
RoutingRule PBRRule1 1
{
IpDestAddr 10.1.100.222 2
TrafficDescriptorGroupRef PBRRule1~
Priority 500000
```

```
RoutingActionRef PBRRule1 4
}

## Action for RoutingRule:PBRRule1 5
RoutingAction PBRRule1
{
UseMainRouteTable Yes
RouteTableRef PBRTAB1 6
}

RouteTable PBRTAB1 7
{
DynamicRoutingParms OSA2080I 8
DynamicRoutingParms OSA20A0I 8
}

TrafficDescriptorGroup PBRRule1~
{
TrafficDescriptor
{
Protocol TCP 9
SourcePortRange 0
DestinationPortRange 0
Jobname TN3270* 3
}
}
```

In this example, the numbers correspond to the following information:

1. The routing rule PBRrule1 is defined.
2. The destination IP address 10.1.100.222 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 OSA2080I andOSA20A0I specified as the sending interface name.
9. The routing rule refers to the TCP protocol.

Save the generated PBR policy file to z/OS

To install and save the generated file, complete the following steps:

1. In the Configuration Assistant Navigation Tree, click the small symbol to the right of TCP/IP stack name (**Stack - TCPIPD**, in our example) and select **Install Configuration Files** (Figure 15-13).

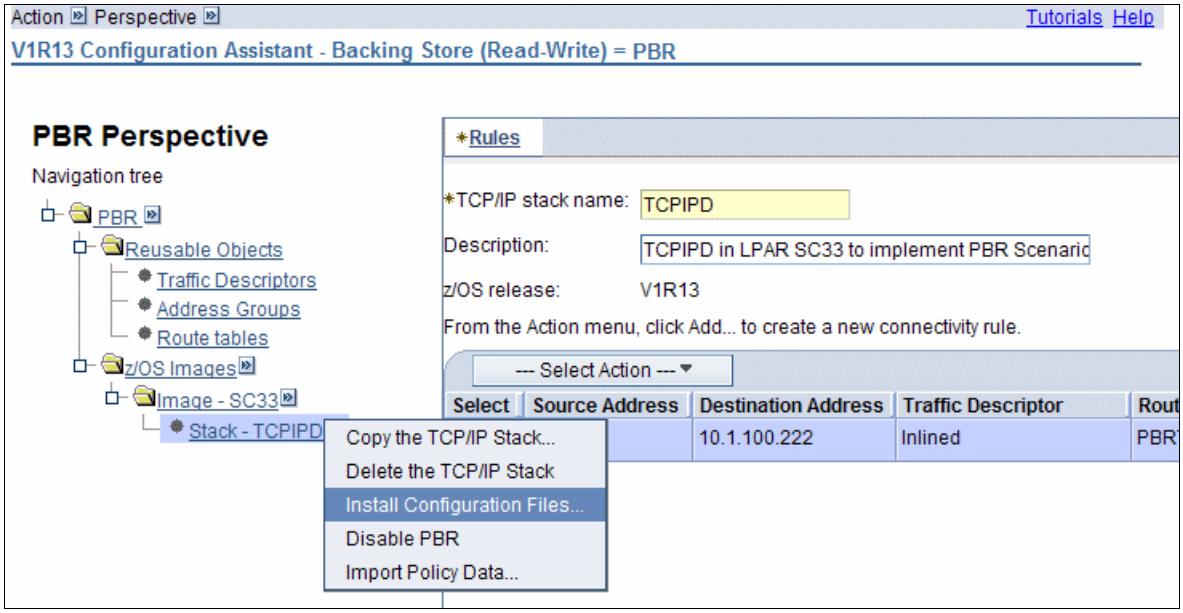


Figure 15-13 PBR Perspective panel with defined policy

2. In the List of Configuration Files for Stack TCPIPD panel, our new routing policy is listed with a status of `Needs install` in red (Figure 15-14). Select the policy and the click **Select Action** → **Install**.

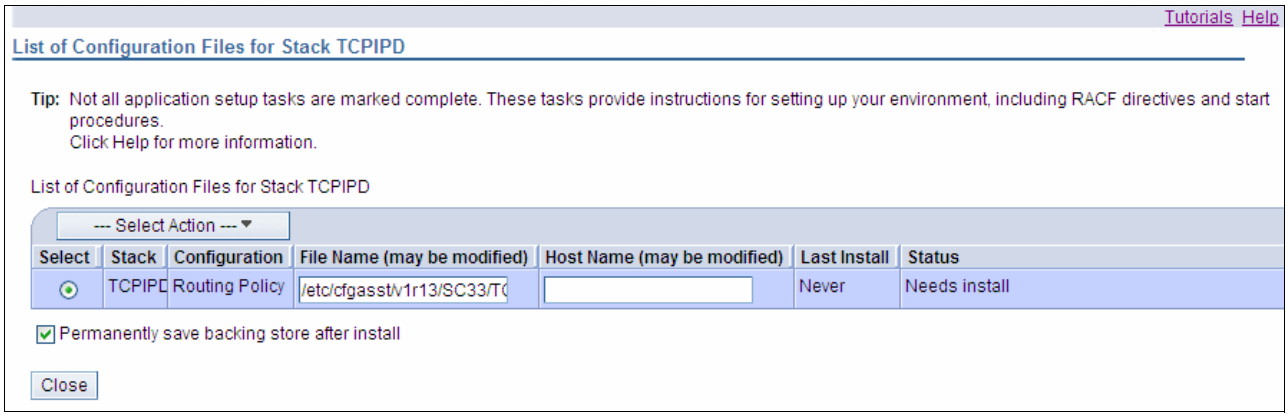


Figure 15-14 Installation panel: Installing Configuration File

3. In the Install Files to Remote Host panel (Figure 15-15), specify the file name and select **Save to disk** to save the policy to the z/OS local disk. Another option is to use FTP to move the file to another z/OS image. In that case, enter the FTP server IP address, port number, FTP user ID and password, and the file name to be saved as. Click **Go**.

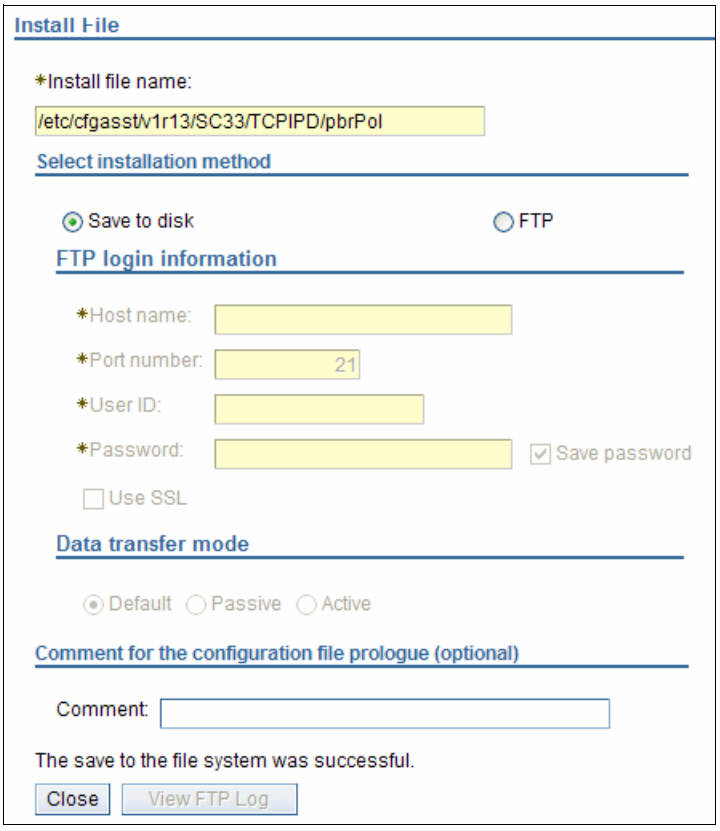


Figure 15-15 FTP Configuration File panel: Transfer the configuration file to z/OS

4. When the save or transfer is complete, the panel indicates the result at the bottom. Click **Close**. In the next panel, you may provide a comment about this saving, and click **OK**. Back in List of Configuration Files for Stack TCPIPD panel, click **Close**.

   The uploaded policy agent configuration file in our scenario is located in the `/etc/cfgasst/v1r13/SC33/TCPIPD` directory where our PAGENT configuration files and policies are located, as shown in Example 15-2.

Example 15-2 Verifying the uploaded configuration file to /etc directory

```
CS02 @ SC33:/u/cs02>cd /etc/cfgasst/v1r13/SC33/TCPIPD
CS02 @ SC33:/etc/cfgasst/v1r13/SC33/TCPIPD>ls
pbrPol
```

Update the policy agent configuration file

Example 15-3 shows the main configuration file of the policy agent, named `/etc/pagent.sc33.conf`. It defines the stack-specific configuration file name for TCPIPD stack.

Example 15-3 Main policy agent configuration file

```
#  ******************************************************************
#  /etc/pagent.sc33.conf in SC33
#  ******************************************************************
# TcpImage and PEPInstance Statements (synonyms)
  TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600
```

Example 15-4 shows the stack-specific configuration file for TCPIPD stack. We added the RoutingConfig statement that points to our PBR policy configuration file.

Example 15-4 Policy agent configuration file for TCPIPD

```
######################################################################
# This file contains                                                 #
# image statements for TCP/IP stack TCPIPD in z/OS image SC33      #
######################################################################
#    RoutingConfig policy configuration                              #
######################################################################
  RoutingConfig /SC33/etc/cfgasst/v1r13/SC33/TCPIPD/pbrPol
```

Tip: To enable the same policy configuration file to all TCP/IP stacks on the z/OS image, you can 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 PBR. The message at **1** in Example 15-5 shows the PBR policy is processed and now in effect.

Example 15-5 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
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCPIPD
```

If you have the policy agent started already, use F *jobname*,REFRESH command as shown in Example 15-6. The message at **2** informs the PBR policy is loaded (or reloaded).

Example 15-6 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 PBR performed as it was defined.

List the defined policies

The **pasearch -R** command lists the PBR rules as shown in Example 15-7.

Example 15-7 The pasearch -R display

```
CS02 @ SC33:/u/cs02>pasearch -R

TCP/IP pasearch CS V1R13                  Image Name: TCPIPD
  Date:                 08/03/2011        Time:  17:15:17
  Routing Instance Id:  1312405749

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
   ActionType:          Routing
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111
   Last to First:       11111111111111111111111111111
   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:           10.1.100.222 4
    ToAddr:             10.1.100.222
   TrafficDescriptor:
    Protocol:           TCP  (6) 5
    SourcePortFrom      0                 SourcePortTo       0
    DestinationPortFrom 0                 DestinationPortTo  0
    JobName             TN3270*  6         SecurityZone
    SecurityLabel
  Policy created: Wed Aug  3 17:09:09 2011
  Policy updated: Wed Aug  3 17:09:09 2011
  Policy updated: Wed Aug  3 17:09:09 2011

  Routing Action:       PBRRule1
    Version:            4                 Status:            Active
    UseMainRouteTable   Yes
    RouteTable:         PBRTAB1 7
    Policy created: Wed Aug  3 17:09:09 2011
    Policy updated: Wed Aug  3 17:09:09 2011
```

In this example, the numbers correspond to the following information:

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 10.1.100.222.

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 PBR tables as shown in Example 15-8.

Example 15-8 The pasearch -T display

```
CSO2 @ SC33:/SC33/etc>pasearch -T

TCP/IP pasearch CS V1R13                  Image Name: TCPIPD
  Date:                 08/03/2011        Time:  17:17:39
  Routing Instance Id:  1312405749

  Route Table:          PBRTAB1      1
    Version:            1                 Status:           Active
    IgnorePathMtuUpdate No                Multipath         UseGlobal
    DynamicXCFRoutes    No
    DynamicRoutingParms
     link_name             OSA2080I  2
    DynamicRoutingParms
     link_name             OSA20A0I  2
    Policy created: Wed Aug  3 17:09:09 2011
    Policy updated: Wed Aug  3 17:09:09 2011
```

In this example, the numbers correspond to the following information:

1. The routing table PBRTAB1 is defined.

2. The routing table has a dynamic routing entry with link name OSA2080I and OSA20A0I.

Verify the Routing Table

Example 15-9 and Example 15-10 on page 662 show the main routing table managed by the TCP/IP stack and OMPROUTE. Notice that there are two equally-costed OSA interfaces to reach destination 10.1.100.0.24, but OSA20E0I is to be used (since this is the top of the routing table). This route table will be used for all traffic that does not match the criteria that we defined.

Example 15-9 Main routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE
EZD0101I NETSTAT CS V1R13 TCPIPD 357
IPV4 DESTINATIONS
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
10.1.100.0/24      10.1.3.240      UGO      0000000000 OSA20E0I 1
10.1.100.0/24      10.1.3.240      UGO      0000000000 OSA20C0I
```

Example 15-10 Main routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE
IEF196I IEF237I 9502 ALLOCATED TO SYS00112
EZZ7847I ROUTING TABLE 371
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)
...
  SPF   10.1.100.0       FFFFFF00  91      22285   10.1.3.240       (2)  1
```

In these examples, the number corresponds to the following information:

1. OSA20E0I (or OSA20C0I) is to be used for sending the packet destined to 10.1.100.0/24.

Example 15-11 and Example 15-12 show the PBR table managed by the TCP/IP stack and OMPROUTE. Notice that the sending interface for all routing entries is OSA2080I. This route table will be used for all traffic generated by TN3270* and destined for 10.1.100.222.

Example 15-11 Partial PBR table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R13 TCPIPD 283
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
  DYNAMICXCFROUTES:    NO
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
DEFAULT            10.1.2.240      UGO      0000000000 OSA2080I
DEFAULT            10.1.2.240      UGO      0000000000 OSA20A0I
...
10.1.100.0/24      10.1.2.240      UGO      0000000000 OSA2080I   2
10.1.100.0/24      10.1.2.240      UGO      0000000000 OSA20A0I   2
90 OF 90 RECORDS DISPLAYED
END OF THE REPORT
```

Example 15-12 PBR table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE,PRTABLE=PBRTAB1
EZZ7847I ROUTING TABLE 339
TABLE NAME: PBRTAB1
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                 0  101     2446    10.1.2.240       (2)
...
 SPF   10.1.100.0       FFFFFF00  101     2446    10.1.2.240       (2)  2

DEFAULT GATEWAY IN USE.

TYPE COST     AGE       NEXT HOP
SPIA 101      2446      10.1.2.240       (2)
                       0 NETS DELETED
DYNAMIC ROUTING PARAMETERS
  INTERFACE:  OSA2080I         NEXT HOP: ANY
  INTERFACE:  OSA20A0I         NEXT HOP: ANY
```

In these examples, the number corresponds to the following information:

2. OSA2080I is the most preferred link be used for sending the packets destined for 10.1.100.0/24, followed by OSA20A0I.

Now we start a new TN3270 session from terminal with IP address 10.1.100.222. The traffic uses our defined policy and therefore uses the PBR table.

As shown in Example 15-13, 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, OSA2080I is used for sending traffic on this connection.

Example 15-13 Netstat All Display

```
CS02 @ SC33:/u/cs02>netstat -p TCPIPD -c
MVS TCP/IP NETSTAT CS V1R13       TCPIP Name: TCPIPD          11:41:15
User Id  Conn     State
-------  ----    -----
TN3270D  00000B0F Establsh
  Local Socket:   ::ffff:10.1.1.40..23
  Foreign Socket: ::ffff:10.1.100.222..2693

CS02 @ SC33:/u/cs02>netstat -p TCPIPD -A -P 2693
MVS TCP/IP NETSTAT CS V1R13       TCPIP Name: TCPIPD          11:42:08
Client Name: TN3270D                  Client Id: 00000B0F
  Local Socket: ::ffff:10.1.1.40..23
  Foreign Socket: ::ffff:10.1.100.222..2693
    RoutingPolicy:      Yes
      RoutingTableName: PBRTAB1    1
      RoutingRuleName:  PBRRule1   2
```

In this example, the numbers correspond to the following information:

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 10.1.100.222 is initiated, RoutingPolicy appears, but with a value of No. However, RoutingTableName and RoutingRuleName do not appear.

To verify that the traffic is actually going through the defined OSA, you can use **Netstat Devlinks** command as shown in Example 15-14 to see the BytesOut and Outbound Packets are incremented on the specific OSA.

Example 15-14 Netstat Devlinks display

```
CS02 @ SC33:/u/cs02>netstat -p TCPIPD -d -K OSA2080I
MVS TCP/IP NETSTAT CS V1R13       TCPIP Name: TCPIPD          11:45:03
IntfName: OSA20C8I          IntfType: IPAQENET   IntfStatus: Ready
    PortName: OSA2080   Datapath: 2082     DatapathStatus: Ready
  Interface Statistics:
    BytesIn                          = 0
    Inbound Packets                  = 0
    Inbound Packets                  = 0
    Inbound Packets In Error         = 0
    Inbound Packets Discarded        = 0
    Inbound Packets With No Protocol = 0
    BytesOut                         = 242363
    Outbound Packets                 = 2678
    Outbound Packets In Error        = 0
    Outbound Packets Discarded       = 0
```

Take down OSA2080I

If we stop the most preferred OSA2080I link, the OSA20A0I floats up in the routing table, as shown in Example 15-15.

Example 15-15 OSA20A0I is preferred next to OSA2080I

```
V TCPIP,TCPIPD,STOP,OSA2080I
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPD,STOP,OSA2080I
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY

D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R13 TCPIPD 348
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
  DYNAMICXCFROUTES:    NO
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
DEFAULT            10.1.2.240      UGO      0000000000 OSA20A0I
10.1.100.0/24      10.1.2.240      UGO      0000000000 OSA20A0I
```

Take down OSA20A0I

If we also take down the OSA20A0I link, there are no available routes in the PBRTAB1 routing table, as shown in Example 15-16. We can still reach 10.1.100.0/24 by using the main routing table.

Example 15-16 Main routing table is used

```
V TCPIP,TCPIPD,STOP,OSA20A0I
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPD,STOP,OSA20A0I
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY

D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R13 TCPIPD 357
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
  DYNAMICXCFROUTES:    NO
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
0 OF 0 RECORDS DISPLAYED
END OF THE REPORT

D TCPIP,TCPIPD,N,ROUTE
EZD0101I NETSTAT CS V1R13 TCPIPD 359
IPV4 DESTINATIONS
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
DEFAULT            10.1.3.240      UGO      0000000000 OSA20E0I
DEFAULT            10.1.3.240      UGO      0000000000 OSA20C0I
10.1.100.0/24      10.1.3.240      UGO      0000000000 OSA20E0I
10.1.100.0/24      10.1.3.240      UGO      0000000000 OSA20C0I
```

15.3.2 Policy-based routing using protocol and port numbers

In this scenario we implement PBR 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 IBM HiperSockets™ connection. Figure 15-16 illustrates the PBR scenario configuration.

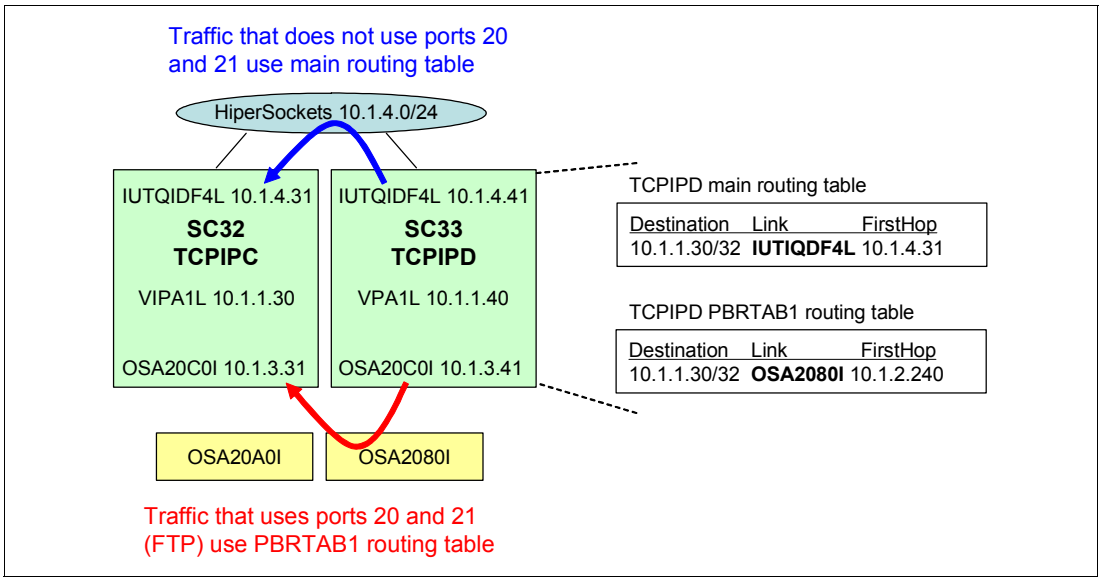


Figure 15-16 PBR using protocol and port numbers

Implementation tasks

The tasks are as follows:

1. Configure the PBR policy with protocol and port numbers.
2. Save 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 that we created for scenario 1 as described in 15.3.1, “Policy-based routing using jobname, protocol, and destination IP address” on page 646. We added the additional rule. If you are starting the PBR configuration from the beginning, follow steps 1 through 4 described in 15.3.1, “Policy-based routing using jobname, protocol, and destination IP address” on page 646, and then continue with the steps in this section.

To add this rule, use the z/OSMF Configuration Assistant and complete the following steps:

1. Start z/OSMF Configuration Assistant. In the Main Perspective, click **Stack - TCPIPD** under Image-SC33 in Configuration Assistant Navigation Tree.

   In the “z/OS Communications Server Technologies” field, the PBR technology is shown as `Enabled`, meaning policies are already saved for PBR. Select **PBR** and click **Select Action** → **Configure**.

2. The Configuration Assistant opens the PBR perspective connected to stack TCPIPD. Click **Select Action** → **Add** to add the new PBR rule.

3. In the New Connectivity Rule - Welcome panel (Figure 15-17), define a name for the new PBR rule (`PBRRule2`). Click **Next**.

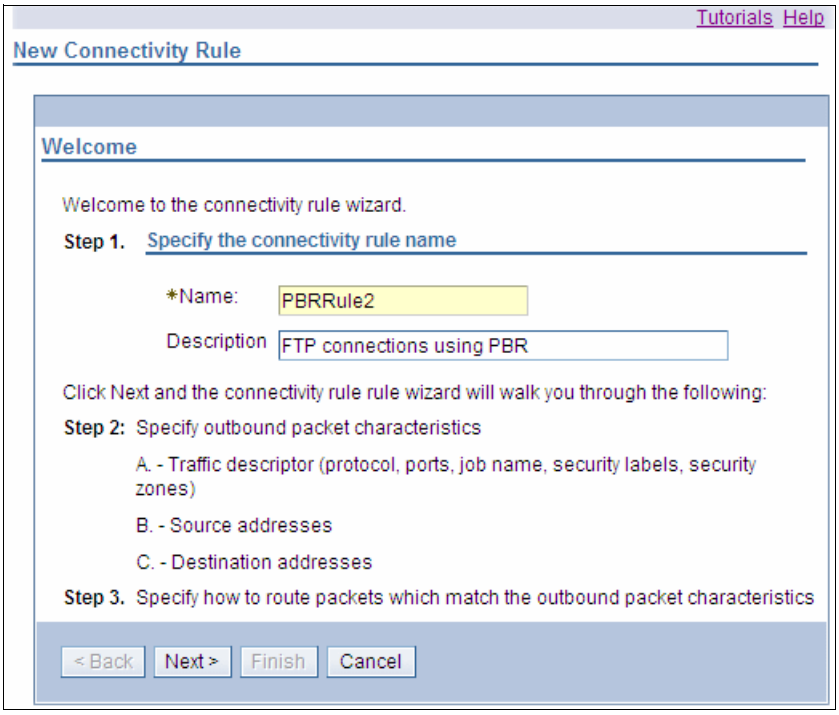


Figure 15-17 Connectivity rule: Welcome

4. In the New Connectivity Rule - Traffic Descriptor panel, select **Yes - Specify the traffic descriptor parameters**, select the **TCP** protocol in the Protocol field, and click **Add.**
5. In the New Connectivity Rule - New Traffic Type panel, define a rule for outbound traffic with source ports 20 and 21 (the FTP data and control ports).

6. To create this policy, go to the Source port field, click **Port range**, and type the port numbers (in the Lower Port field we entered 20, and in the Upper Port field we entered 21 as shown in Figure 15-18. Then, click **OK** to save the input data and save the policy.

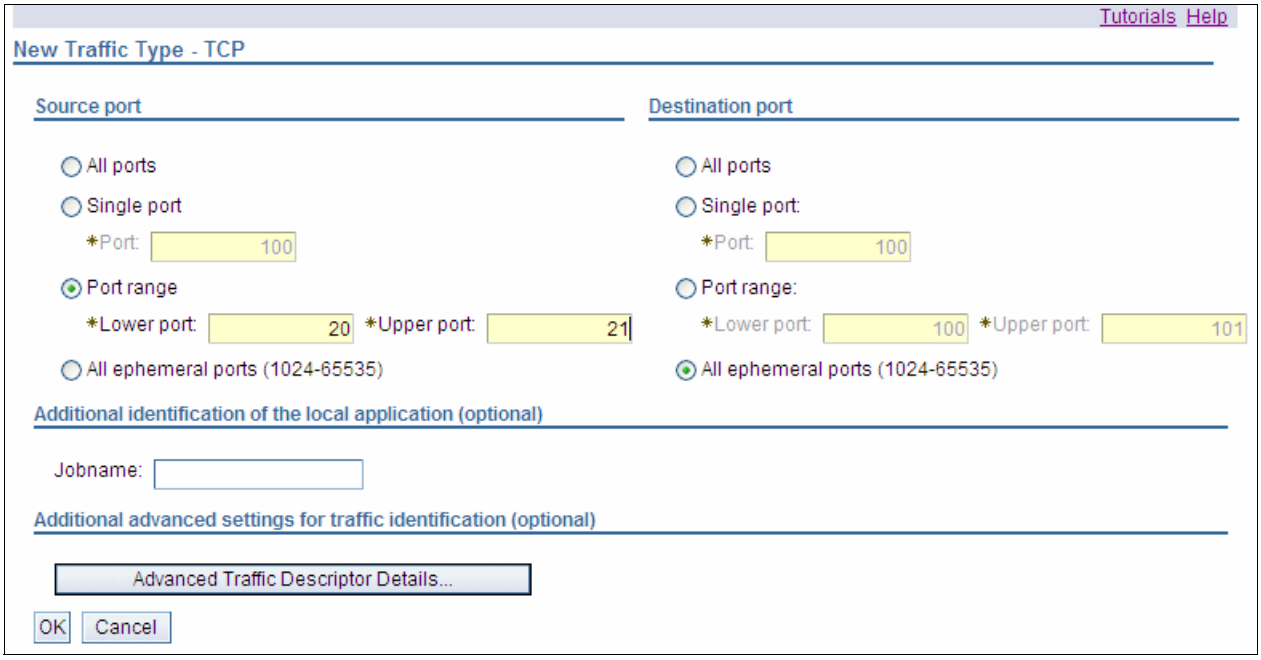


Figure 15-18 New Traffic Type for FTP traffic only

7. Back in the New Connectivity Rule - Traffic Descriptor panel, confirm that the policy is added in the list of traffic types in this traffic descriptor. Continue by clicking **Next**,
8. The next two Connectivity Rule panels are used to define the source and destination address for this PBR rule. In our scenario, we used the default rule, **NO - rule matching is not dependent on the source (or destination) address** on both panels, and click **Next**.

9. In the New Configuration Rule - Route Table panel, we use the same route table that we created in scenario 1 (`PBRTAB1`) in this chapter, which routes all outbound traffic through link `OSA2080I` and `OSA20A0I`. From the drop-down list in the Primary Route Table field, select the desired route table, **PBRTAB1**, as shown in Figure 15-19.

   We also select the **Use the Stack's main route table** option for fallback routing. Click **Finish** to end the configuration and return to the PBR Perspective.

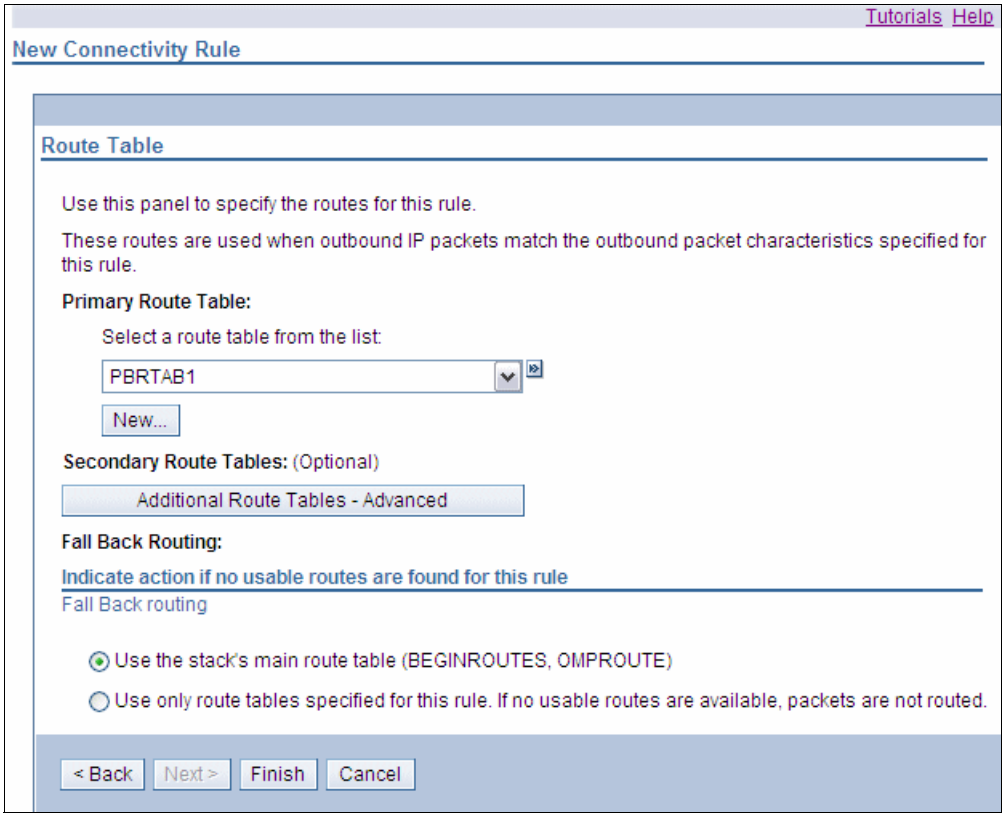


Figure 15-19 Rule Step 3: Route Table

10. Back at the PBR Perspective panel, the connectivity rule is created. Click **Apply Changes** to save it in the configuration file.

Save the generated PBR policy file to z/OS

To upload the generated file, follow the same steps used to upload the scenario 1 configuration files in 15.3.1, “Policy-based routing using jobname, protocol, and destination IP address” on page 646.

1. Install the PBR policy configuration file by sending the file to z/OS to be included in the policy agent configuration. To install the file, click the small symbol to the right of **Stack - TCPIPD**, and select **TCPIPD** → **Install Configuration Files**.
2. Select the file and click **Select Action** → **Install** (Figure 15-20).

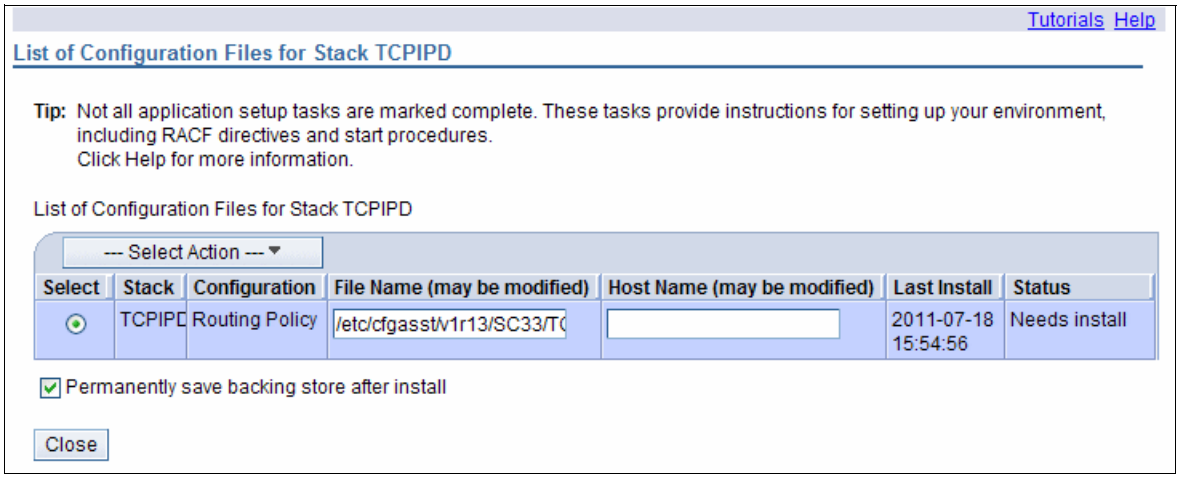


Figure 15-20 Configuration files installation

3. In the Install File panel, we select the **Save to disk** option to save the policy to the local z/OS file stem (another option is FTP to the other z/OS image). Click **Go**.
4. After sending the files to z/OS, confirm that the files were sent by viewing the bottom of the same panel. Click **Close**. In the next panel, you may provide a comment for this saving. Click **OK**. Back in the List of Configuration Files for Stack TCPIPD panel, click **Close**.
5. Verify the uploaded PBR policy file in the `/etc/cfgasst/v1r13/SC33/TCPIPD` directory where we stored it as shown in Example 15-17.

Example 15-17 verifying the uploaded configuration file is in the /etc directory

```
CS02 @ SC33:/u/cs02>cd /etc/cfgasst/v1r13/SC33/TCPIPD
CS02 @ SC33:/SC33/etc/cfgasst/v1r13/SC33/TCPIPD>ls
pbrPol
```

Update the policy agent configuration file

Example 15-18 shows the policy agent configuration file that defines the stack-specific configuration file for TCPIPD.

Example 15-18 Global policy agent configuration file

```
#  ******************************************************************
#  /etc/pagent.sc33.conf in SC33
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600
```

We added the RoutingConfig statement that points to the policy configuration file as shown in Example 15-19.

Example 15-19 Policy agent configuration file for TCPIPD

```
###################################################################
# This file contains                                              #
# image statements for TCP/IP stack tcpipd in z/OS image sc33     #
###################################################################
#    RoutingConfig policy configuration                           #
###################################################################
  RoutingConfig /SC33/etc/cfgasst/v1r13/SC33/TCPIPD/pbrPol
```

Tip: To enable the same policy configuration file to all TCP/IP stacks on the z/OS image, you can 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 PBR in effect as shown in Example 15-20.

Example 15-20 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
EZZ4249I TCPIPD INSTALLED TTLS POLICY HAS NO RULES
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : ROUTING
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCPIPD
```

If you have the policy agent started already, use the F ***pagent*,REFRESH** command.

Verification

List the defined policies and verify the routing table.

List the defined policies

We used the **pasearch -R** command to list the PBR rules and to verify that the rule we created for our scenario 2 is applied, as shown in Example 15-21.

Example 15-21 The pasearch -R command

```
CS02 @ SC33:/u/cs02>pasearch -R

TCP/IP pasearch CS V1R13                 Image Name: TCPIPD
  Date:                 08/03/2011       Time:  17:36:40
 Routing Instance Id:  1312405749

<<PBRRule1 from scenario1 is omitted from this output>>

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
 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
Policy created: Wed Aug  3 17:09:09 2011
Policy updated: Wed Aug  3 17:09:09 2011

Routing Action:       PBRRule2  5
  Version:            4                Status:            Active
  UseMainRouteTable   Yes
  RouteTable:         PBRTAB1
  Policy created: Wed Aug  3 17:09:09 2011
  Policy updated: Wed Aug  3 17:09:09 2011
```

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 PBR tables that are active in our environment as shown in Example 15-22.

Example 15-22 The pasearch -T command

```
CS02 @ SC33:/u/cs02>pasearch -T

TCP/IP pasearch CS V1R13                  Image Name: TCPIPD
  Date:                 08/03/2011        Time:  17:39:06
  Routing Instance Id:  1312405749

  Route Table:          PBRTAB1
    Version:            1                 Status:            Active
    IgnorePathMtuUpdate No                Multipath          UseGlobal
    DynamicXCFRoutes    No
    DynamicRoutingParms
     link_name             OSA2080I
    DynamicRoutingParms
     link_name             OSA20A0I
    Policy created: Wed Aug  3 17:09:09 2011
    Policy updated: Wed Aug  3 17:09:09 2011
```

Verify the routing table

Example 15-23 and Example 15-24 show the main routing table managed by TCP/IP stack and OMPROUTE. Notice that 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 15-23 Main routing table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE
EZD0101I NETSTAT CS V1R13 TCPIPD 711
IPV4 DESTINATIONS
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
10.1.1.30/32       10.1.4.31       UGH0     0000000000 IUTIQDF4L
10.1.1.30/32       10.1.5.31       UGH0     0000000000 IUTIQDF5L
```

Example 15-24 Main routing table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE
EZZ7847I ROUTING TABLE 713
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                0  91      23499   10.1.3.240       (2)
...
 SPF   10.1.1.30        FFFFFFFF  90      33807   10.1.4.31        (2)
 SPF*  10.1.4.0         FFFFFF00  80      33807   IUTIQDF4L
```

Example 15-25 and Example 15-26 show the PBR table managed by TCP/IP stack and OMPROUTE. Notice that OSA2080I is to be used for sending traffic to any destination if the conditions established in our routing policy are met, as we defined.

Example 15-25 PBR table managed by TCP/IP stack

```
D TCPIP,TCPIPD,N,ROUTE,PR=PBRTAB1
EZD0101I NETSTAT CS V1R13 TCPIPD 715
IPV4 DESTINATIONS
POLICY ROUTING TABLE: PBRTAB1
  IGNOREPATHMTUUPDATE: NO   MULTIPATH: NO(PROFILE)
  DYNAMICXCFROUTES:    NO
DESTINATION        GATEWAY         FLAGS    REFCNT     INTERFACE
DEFAULT            10.1.2.240      UGO      0000000000 OSA2080I
DEFAULT            10.1.2.240      UGO      0000000000 OSA20A0I
10.1.1.30/32       10.1.2.32       UGHO     0000000000 OSA2080I
10.1.1.30/32       10.1.2.31       UGHO     0000000000 OSA2080I
10.1.1.30/32       10.1.2.32       UGHO     0000000000 OSA20A0I
10.1.1.30/32       10.1.2.31       UGHO     0000000000 OSA20A0I
10.1.2.0/24        0.0.0.0         UO       0000000000 OSA2080I
10.1.2.0/24        0.0.0.0         UO       0000000000 OSA20A0I
```

Example 15-26 PBR table managed by OMPROUTE

```
D TCPIP,TCPIPD,OMPR,RTTABLE,PRTABLE=PBRTAB1
EZZ7847I ROUTING TABLE 728
TABLE NAME: PBRTAB1
TYPE   DEST NET         MASK      COST    AGE     NEXT HOP(S)

SPIA   0.0.0.0                 0  101     2085    10.1.2.240       (2)
...
 SPF   10.1.1.30        FFFFFFFF  110     2085    10.1.2.32        (4)
 SPF*  10.1.2.0         FFFFFF00  100     2085    OSA2080I         (2)
```

Next, we test whether our routing policy is working as expected. We start an FTP session from a TSO user on LPAR SC32, using the CS02 user. This session uses ports 20 and 21, which should match the routing policy. Outgoing traffic should use the PBR 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 15-27.

Example 15-27 TSO command Netstat conn tcp tcpipd

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS V1R13 TCPIPD 816
USER ID  CONN     STATE
BBNSO01  OOOO43CD LISTEN
  LOCAL SOCKET:   ::..32202
  FOREIGN SOCKET: ::..0
FTPDD1   OOOO46CC ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.30..1029
```

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 (Example 15-28).

Example 15-28 Netstat All tcp tcpipd (port 1028 command

```
D TCPIP,TCPIPD,N,ALL
EZD0101I NETSTAT CS V1R13 TCPIPD 820
CLIENT NAME: FTPDD1                  CLIENT ID: 000046CC
  LOCAL SOCKET: ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.30..1029
    ROUTINGPOLICY:      YES
      ROUTINGTABLENAME: PBRTAB1
      ROUTINGRULENAME:  PBRRULE2
```

To verify the traffic is actually going through the expected OSA link, OSA2080I, we used the command Netstat Devlinks and looked at the BytesOut and Outbound Packets fields before and after the data traffic operation to see they had incremented on the expected OSA link, as shown in Example 15-29.

Example 15-29 Netstat Devlinks tcp tcpipd (intfname OSA2080I display command)

```
***** Before data transfer *******
D TCPIP,TCPIPD,N,DEV,INTFNAME=OSA2080I
EZD0101I NETSTAT CS V1R13 TCPIPD 822
INTFNAME: OSA2080I          INTFTYPE: IPAQENET   INTFSTATUS: READY
    PORTNAME: OSA2080   DATAPATH: 2082     DATAPATHSTATUS: READY
...
    BYTESOUT                          = 245214
    OUTBOUND PACKETS                  = 2713
    OUTBOUND PACKETS IN ERROR         = 0
    OUTBOUND PACKETS DISCARDED        = 0
...

 **** After data tranfer ******

D TCPIP,TCPIPD,N,DEV,INTFNAME=OSA2080I
EZD0101I NETSTAT CS V1R13 TCPIPD 834
INTFNAME: OSA2080I          INTFTYPE: IPAQENET   INTFSTATUS: READY
    PORTNAME: OSA2080   DATAPATH: 2082     DATAPATHSTATUS: READY
...
    BYTESOUT                          = 258172
    OUTBOUND PACKETS                  = 2734
    OUTBOUND PACKETS IN ERROR         = 0
    OUTBOUND PACKETS DISCARDED        = 0
...
```

Part 4

Application-based security

In this part, we explain how you can use built-in security functions for additional security. We identify a few of the common security exposures and tools that you can use to address the issues. The major emphasis in this chapter is Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Application Transparent Transport Layer Security (AT-TLS) features that are supported by the z/OS standard applications (TN3270 and FTP). These applications provide client and server authentication and data encryption methods using digital certificates with a protocol called Transport Layer Security (TLS).

The TN3270 and FTP servers support the following TLS methods:

- TLS is provided natively by internal code, and is the basic or less functional method.
- AT-TLS is provided externally through the policy agent and is the preferred method because of the additional functionality it provides.

This part contains the following chapters:

16

Telnet security

In this chapter, we explain how to 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.

We discuss the following topics in this chapter.

Section	Topic
16.1, “Conceptual overview of TN3270 security” on page 678	An overview of TN3270 security features
16.2, “TN3270 native TLS connection security” on page 680	TN3270 native security functions
16.3, “Basic native TLS configuration example” on page 685	Implement a basic TN3270 connection security with native TLS
16.4, “TN3270 with AT-TLS security support” on page 692	TN3270 TLS connection security using AT-TLS
16.5, “Basic AT-TLS configuration example” on page 696	Implement a basic TN3270 connection security with AT-TLS
16.6, “Problem determination for Telnet server security” on page 718	Problem determination methods for TN3270 server issues
16.7, “Additional information sources for TN3270 AT-TLS support” on page 718	References to additional information about TN3270 security topics

16.1 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

16.1.1 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. AT-TLS is provided externally through the policy agent, and is the preferred method because of the additional functionality it provides.

16.1.2 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 IBM 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. 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 AT-TLS to manage secure connections. TLS managed by AT-TLS (TTLSPORT) supports more security functions than TLS managed by the TN3270 (SECUREPORT).

Tip: 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, you should migrate to the use of AT-TLS as a consistent solution for all of your TCP applications.

Be aware of the following 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.

Tip: 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 to achieve a consistent solution for *all* of your TCP applications.

16.1.3 How TN3270 security can be applied

To implement data overrun protection, statements can be placed into the TN3270 server configuration profile data set, into the TCP/IP stack configuration profile data set, and into VTAM start options.

To implement network access control, statements are placed into the TN3270 server configuration profile data set 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 data set that instruct the server on 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 Configuration Assistant is used to create the appropriate policy agent statements for AT-TLS.

16.2 TN3270 native TLS connection security

This section provides an overview of executing the Telnet server with native connection security features.

16.2.1 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 to)
- 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. See *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 key ring 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.

  Tip: 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 listed in Table 16-1.

Table 16-1 Encryption algorithms Default ordered list

ENCRYPTION algorithm_name	Abbreviation in Telnet displays
SSL_RC4_SHA	4S
SSL_RC4_MD5	4M
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

- More statements are available for customizing secure ports. For a complete list of parameter statements and syntax, see *z/OS Communications Server: IP Configuration Reference*, SC31-8776. For usage discussions, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Tip: 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 about 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
- Network Access Control

The most commonly used of these is the Transport Layer Security (TLS). In this section, we discuss the following benefits:

- Benefits of implementing Data Overrun security
- Benefits of implementing 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 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 the following 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 can adversely affect existing methods of connectivity and cause interruption of service.
- Costs associated with additional security measures might 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, certain type of access security and connection security will be necessary. Some organizations might implement security at the networking level, not requiring any authentication or encryption at the application level. Others might 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. See the following site for a better understanding of security methodology:

`http://www.verisign.com/corporate/index.html?tid=footer`

Secure TN3270 connections

Depending on the business requirements of your organization, some type of access security and connection security is probably necessary. Your organization might implement security at the networking level, not requiring any authentication or encryption at the stack or application level. Other organizations might 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.

Tip: 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 AT-TLS capabilities. For more information about this topic, see Chapter 8, “IP Security” on page 245, to Chapter 12, “Application Transparent Transport Layer Security” on page 551, and to Chapter 3, “Certificate management in z/OS” on page 39, according to your requirements.

16.2.2 Configuring TN3270 native connection security

The Telnet server supports various methods and levels of connection security. We discuss these methods in the sections that follow.

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. See *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 can 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 an implementation example of the secure connection using the native TLS support, we describe a minimum configuration scenario using native TLS in 16.3, "Basic native TLS configuration example" on page 685.

For further information about configuring the TN3270 stand-alone started task, see *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997.

16.3 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, see Appendix B, "Telnet security advanced settings" on page 823.

We configure TN3270 with native TLS using a stand-alone 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 16-1 depicts the environment used for this scenario.

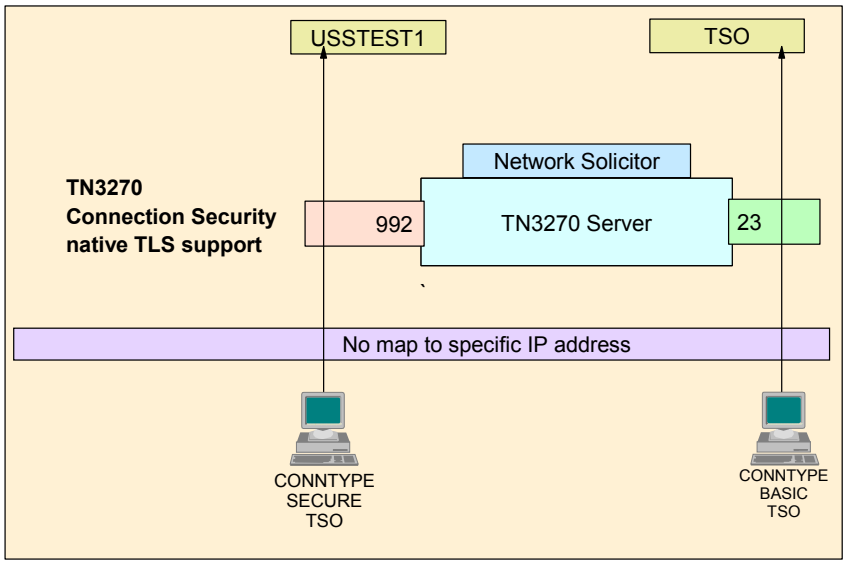


Figure 16-1 Minimum TN3270 native TLS configuration

16.3.1 Enabling native TSL/SLL support for TN3270

To enable native TLS/SSL support for TN3270 with server authentication:

1. Generate the key ring and certificates.
2. Add a TLS-enabled port and security parameters to TN3270 profile.
3. Start the TN3270 server.

Generate the key ring and certificates

Tip: Chapter 3, “Certificate management in z/OS” on page 39, contains detailed examples and explanations for the steps involved in preparing the key ring and certificates used in this scenario. See that chapter for a complete discussion about using a shared key ring and SITE certificate.

Example 16-1 shows the RACF statements necessary to create the shared key ring, the CA certificate, and the SITE certificate used by this scenario.

Example 16-1 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)
```

In this example, the numbers correspond to the following information:

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

Configure TN3270 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 16-2.

Example 16-2 Minimum TN3270 profile for native TLS secure connection

```
TELNETGLOBALS
    TCPIPJOBNAME TCPIPD
    TNSACONFIG ENABLED COMMUNITY jOs9m2ap 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          ;Port 992 will support native TLS
    CONNTYPE SECURE       ; Support SSL
    KEYRING SAF TCPIP/SharedRing1       ;keyring shared by servers
;   CLIENTAUTH SSLCERT    ; Client Certificate required
    CLIENTAUTH NONE       ; Client Certificate required
    ENCRYPT SSL_DES_SHA   ; use these only, do not consider any others
            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

; DEFAULTAPPL SC33TS        ; All users go to TSO
  USSTCP   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
    PORT 23
    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
```

In this example, the numbers correspond to the following information:

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 16-3 shows the messages given at the initialization of the TN3270 server.

Example 16-3 Starting the TN3270 server

```
S TN3270D
.....
EZZ6001I TELNET SERVER STARTED
.....
EZZ6003I TELNET LISTENING ON PORT   992 1
EZZ6003I TELNET LISTENING ON PORT    23 2
```

In this example, the numbers correspond to the following information:

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).

16.3.2 Activating and verifying the configuration

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 V5.7 to establish the connection. Because we used a self-signed certificate, we downloaded and installed the certificate of certificate authority into Personal Communications.

Use TELNET CONN displays to show TN3270 connections

Example 16-4 shows the display of existing TN3270 connections using the TELNET CONN command.

Example 16-4 Display Telnet CONN to see TN3270 connections

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TN3270D CONN DISPLAY 625
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00004AC1 DS ::FFFF:10.1.100.222..1356 1
                                   SC33DS01           TPE 2
----- PORT:   992 3 ACTIVE           PROF: CURR CONNS:      1
```

In this example, the numbers correspond to the following information:

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 16-5 shows the display of a detailed information, including TLS, for a specific connection. Specify the connection ID that was displayed in Example 16-4 for CONN= option.

Example 16-5 Display Telnet CONN, CONN detail for a specific connection

```
D TCPIP,TN3270D,T,CONN,CONN=4AC1,DETAIL
EZZ6065I TN3270D CONN DISPLAY 628
  CONNECTED: 16:32:58  07/18/2011  STATUS: SESSION PENDING
  CLIENT IDENTIFIER FOR CONN: 00004AC1   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.222..1356 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: **N/A**
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED:           APPL SPECIFIED:
  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)(OPATSKTQSSHRT)(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
```

In this example, the numbers correspond to the following information:

1. The client IP address and port number.
2. The connection is secure, and uses the DS (SSL_DES_SHA) cipher. See Table 16-1 on page 682 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 16-6.

Display PROF to show profile SSL information

The Display PROF command shows the information about the profile being used. Example 16-6 shows the profile defined for port 992.

Example 16-6 Display Telnet PROFILE for TLS information

```
D TCPIP,TN3270D,T,PROF,PORT=992,DET,MAX=*
EZZ6080I TN3270D PROFILE DISPLAY 631
  PERSIS   FUNCTION       DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OPATSKTQSSHRT)(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/TTLS/**N/A**
   ENCRYPTION        DS,3S                            4
   CLIENTAUTH        NONE                             5
   NOEXPRESSLOGON
   NONACUSERID
   NOSSLV2
 TIMERS
   INACTIVE               0 (OFF)
   PROFILEINACTIVE     1800
   KEEPINACTIVE           0 (OFF)
   PRTINACTIVE            0 (OFF)
   SCANINTERVAL         120
   TIMEMARK             600
   SSLTIMEOUT             5
   KEYRING           SAF TCPIP/SharedRing1            6
```

In this example, the numbers correspond to the following information:

1. 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). See Table 16-1 on page 682 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).

16.4 TN3270 with AT-TLS security support

z/OS Communications Server provides the AT-TLS support for TN3270E to implement TLS connection security.

16.4.1 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 z/OSMF Configuration Assistant can be used to generated the AT-TLS configuration flat file. You can also code it directly to the file. Using z/OSMF Configuration Assistant will help you 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 can 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 16-2 describes the equivalent statement in AT-TLS policy configuration for these security parameters.

Table 16-2 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
SSLV2	SSLv2 in: ► TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction ► TTLSConnectionAdvancedParms within TTLSConnectionAction
SSLTIMEOUT	HandshakeTimeout in: ► TTLSEnvironmentAdvancedParms within TTLSEnvironmentAction ► TTLSConnectionAdvancedParms within TTLSConnectionAction
ENCRYPTION	TTLSCipherParms in: ► TTLSConnectionAction ► 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:

- All SSL V3/TLS V1 cipher choices in preferred order are listed in Table 16-3.

Table 16-3 All 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
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

- All SSL Version 2 cipher choices in preferred order are listed in Table 16-4.

Table 16-4 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
TLS_RC4_128_EXPORT40_WITH_MD5	0x2
TLS_RC4_128_WITH_MD5	0x1

The CONNTYPE statement can be defined for TTLSPORT in 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 certain 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 that 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

Generally, use 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 certain 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.

16.4.2 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 can 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 16.5, “Basic AT-TLS configuration example” on page 696.

For further information about how the TN3270 stand-alone started task is configured, see *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997.

16.5 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, see Appendix B, “Telnet security advanced settings” on page 823.

We configure TN3270 with AT-TLS using a stand-alone 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 16.3, “Basic native TLS configuration example” on page 685. 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 16-2 depicts the environment we used for this scenario.

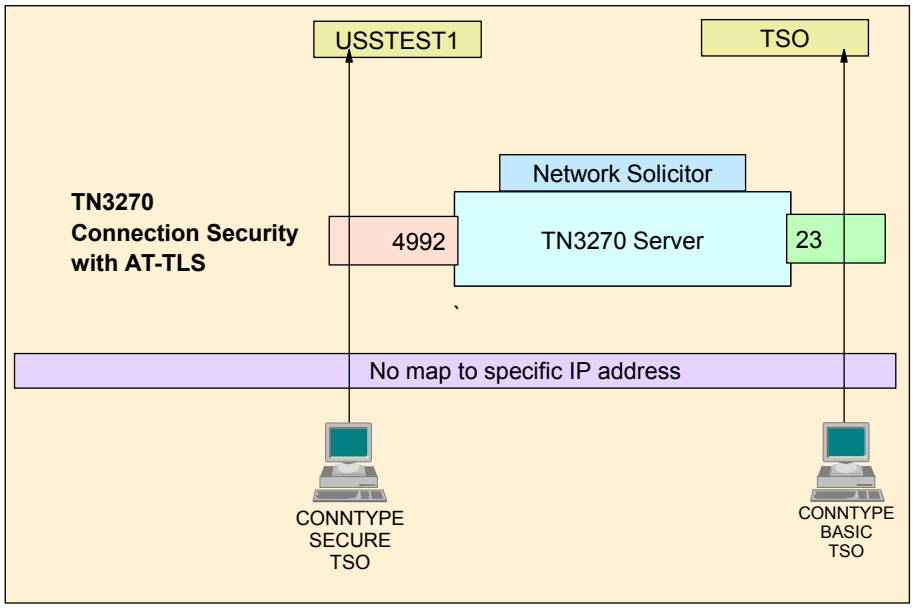


Figure 16-2 TN3270 with minimum AT-TLS configuration diagram

16.5.1 Implementing TN3270 AT-TLS support

To configure TN3270 AT-TLS support, complete the following steps:

1. Set up the policy agent.
2. Create a certificate.
3. Add authorization for the **pasearch** command in RACF.
4. Modify the TCP/IP profile.
5. Define an AT-TLS policy.
6. Save the policy to z/OS.
7. Modify the policy agent configuration file.
8. Define the AT-TLS port in the TN3270 configuration file.

Set up the policy agent

We set up the policy agent as shown in 4.2, “Implementing PAGENT on z/OS” on page 109.

Create a certificate

We can continue to use the certificate created for native TLS connection security as shown in 16.2, “TN3270 native TLS connection security” on page 680.

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 to only administrators and operators. We set up the RACF authorization as shown in 4.2, “Implementing PAGENT on z/OS” on page 109.

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 16-7. To apply the change, either restart the TCP/IP stack or use the OBEYFILE command.

Example 16-7 Modify TCP/IP profile

```
TCPCONFIG TTLS
```

Define an AT-TLS policy

We use the z/OSMF Configuration Assistant to define an AT-TLS policy as follows:

1. Start the z/OSMF Configuration Assistant. In the Main Perspective panel, shown in Figure 16-3, select the **Add a New z/OS image** option.

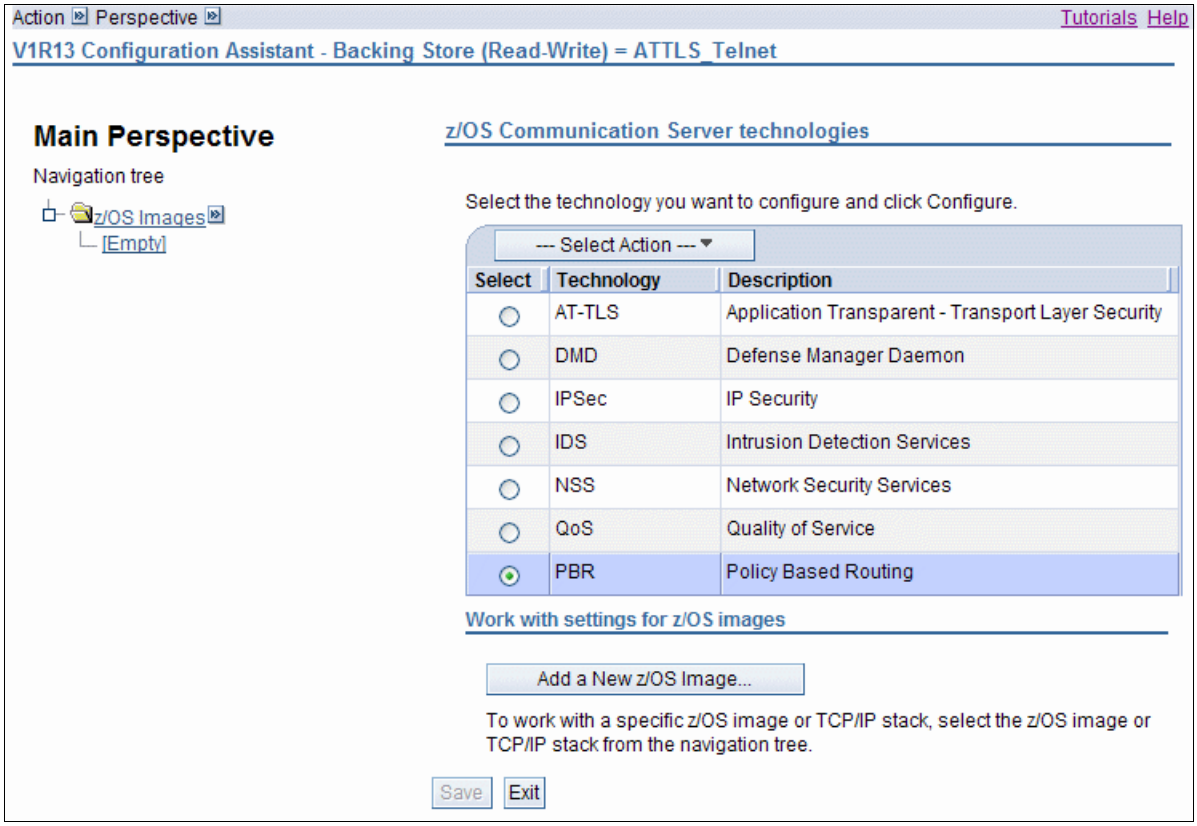


Figure 16-3 Main Perspective panel

2. Enter the name of the z/OS image (in our example, we typed in `SC33`). Click **OK**.
3. If you are prompted to add a new TCP/IP stack to the z/OS image, click **Yes.** If this panel is not shown, then in the Main Perspective, 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 (Figure 16-4), select AT-TLS in the z/OS Communications Server technologies list. Click **Select Action** → **Enable**, and then click **Select Action** → **Configure**.

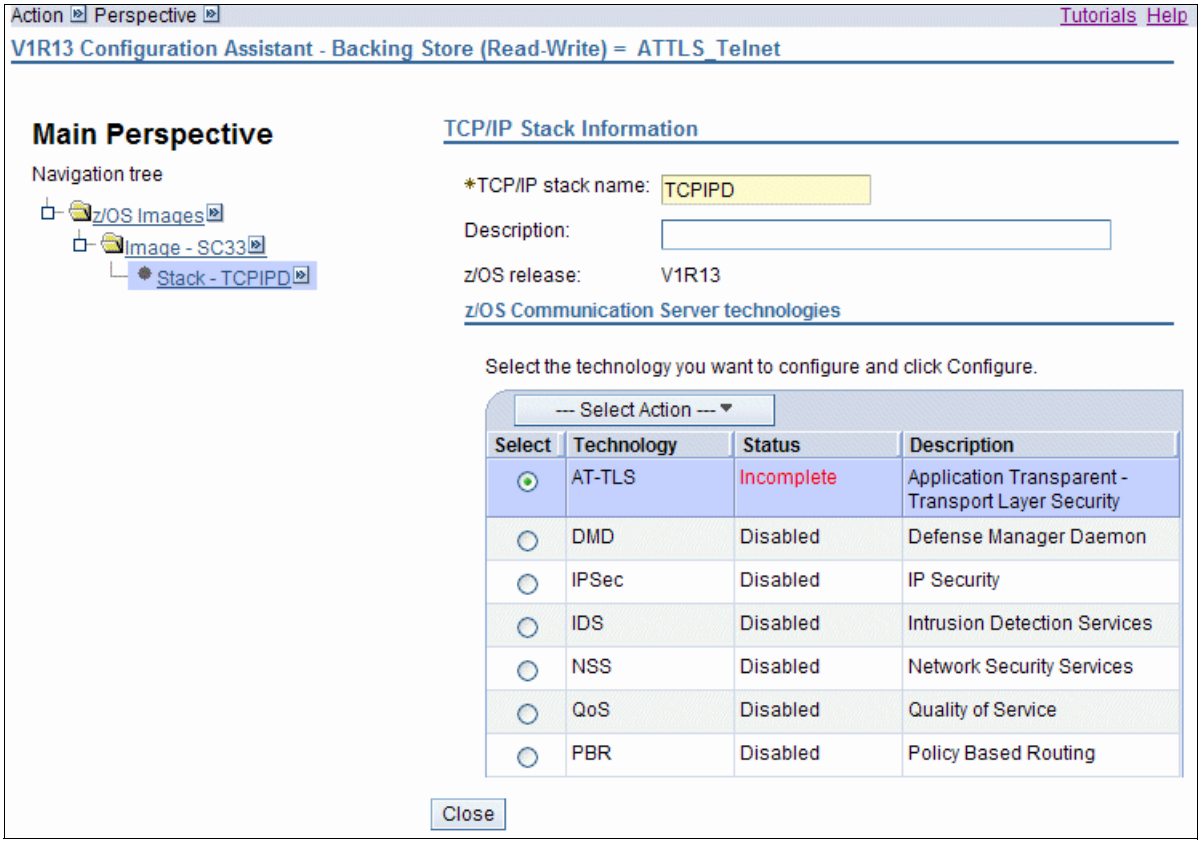


Figure 16-4 Main Perspective panel: Selecting the policy type

5. In the AT-TLS Perspective panel, shown in Figure 16-5, notice the stack in the left pane shows `Incomplete Stack` because the AT-TLS policy is not yet configured. Click **Select Action** → **Add**.

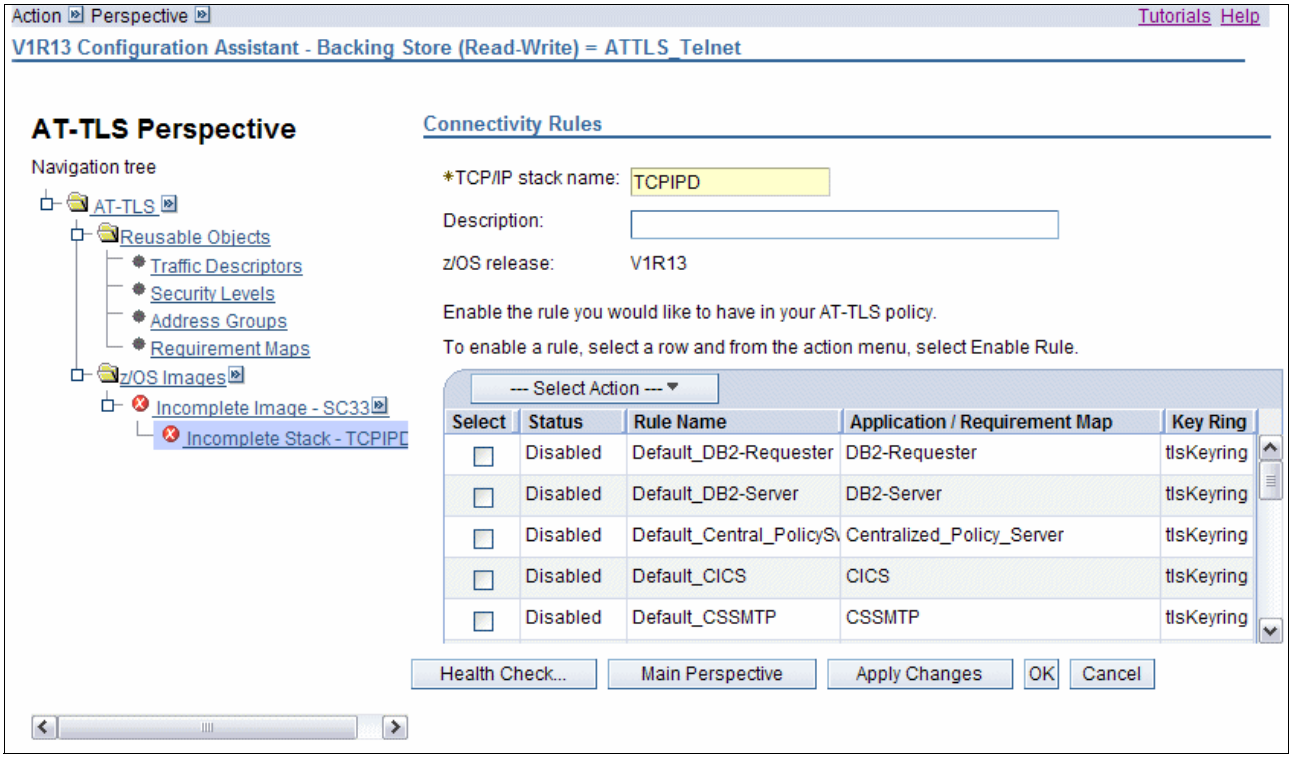


Figure 16-5 AT-TLS Perspective panel

6. In the New Connectivity Rule Welcome panel, click **Next**.

7. In the New Connectivity Rule: Data Endpoints panel (Figure 16-6), 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**.

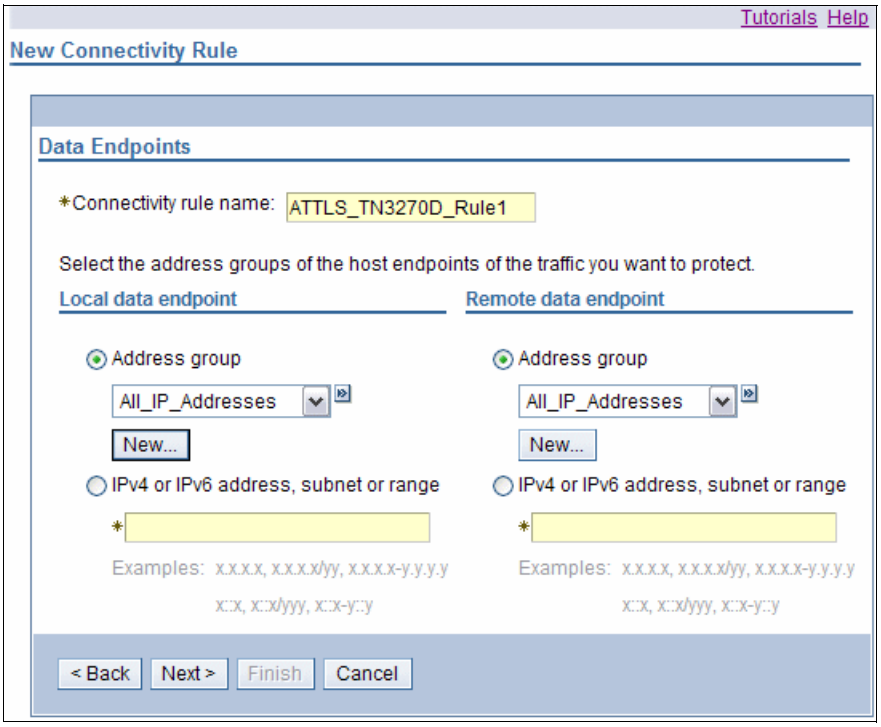


Figure 16-6 New Connectivity Rule: Data Endpoints panel

8. In the Select Requirement Map panel, define the description of the traffic you want to apply the policy. You can select the predefined requirement map from the list if it fits your requirements. There is a predefined AT-TLS Sample entry which contains the traffic descriptor for TN3270E server with TCP port 23 and CICS with TCP port 3000. We use TCP port 4992 for TN3270E in our example, so we create a new requirement map instead of using the predefined one. After selecting Traffic Descriptors, click **Select Action** → **Add**.

9. In the Select Requirement Map panel (Figure 16-7), enter the name of the new requirement map. We named it `ATTLS_TN3270D_ReqMap`. Click **Traffic Descriptors** option.

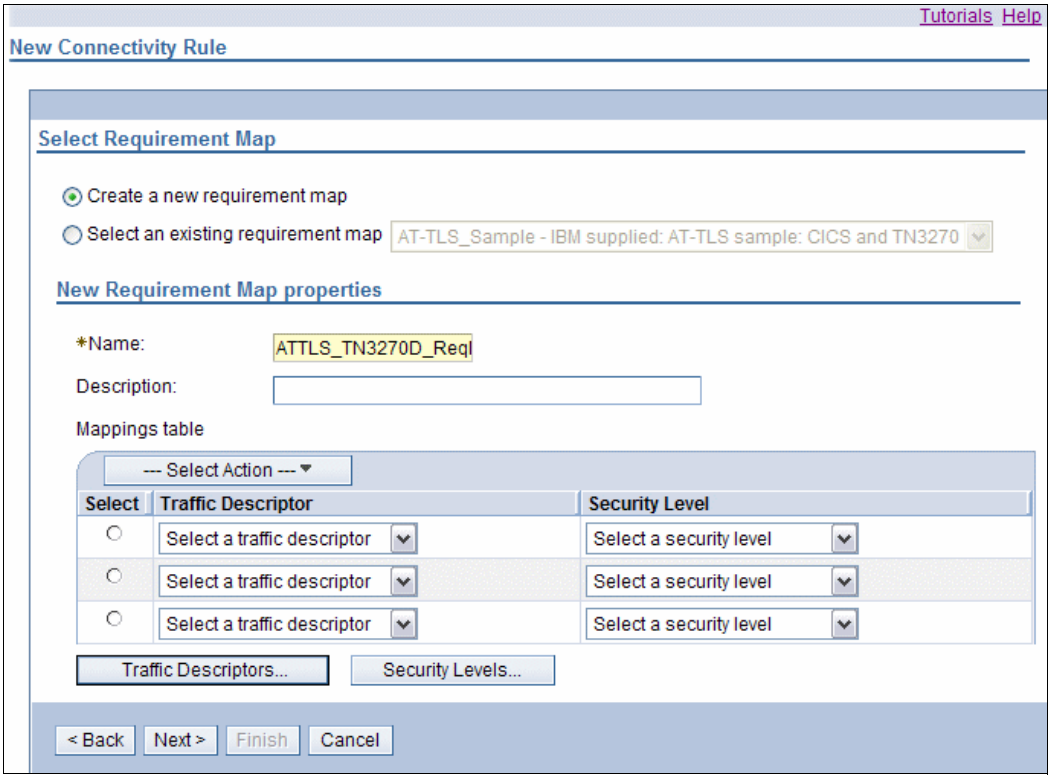


Figure 16-7 Select Requirement Map panel: Defining new requirement map

10. Click **Select Action** → **Add** to define a new traffic descriptor.
11. Enter the name of the new traffic descriptor. As shown in Figure 16-8, we name it `ATTLS_TN3270D_4992`. Click **Select Action** → **Add**.

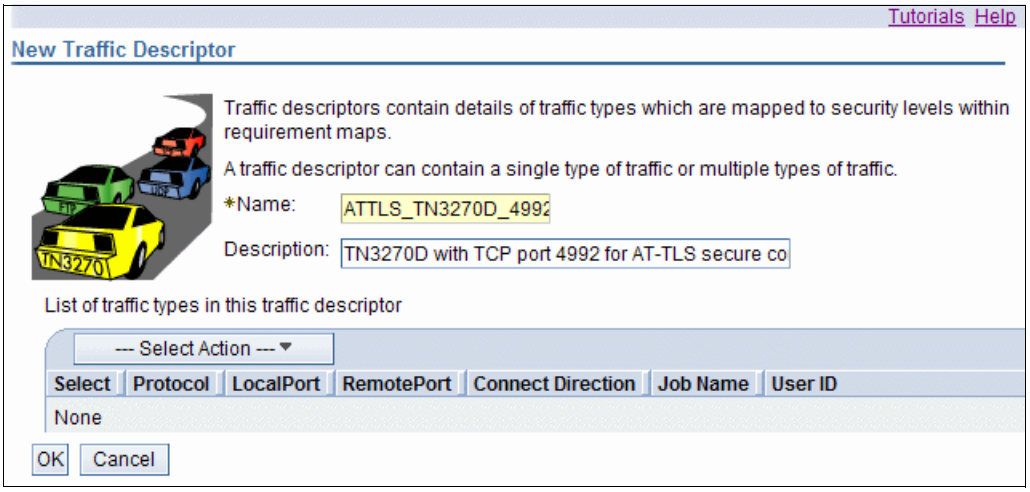


Figure 16-8 New Traffic Descriptor panel

12. Define the local port number and remote port number as shown in Figure 16-9. 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 Advanced tab.

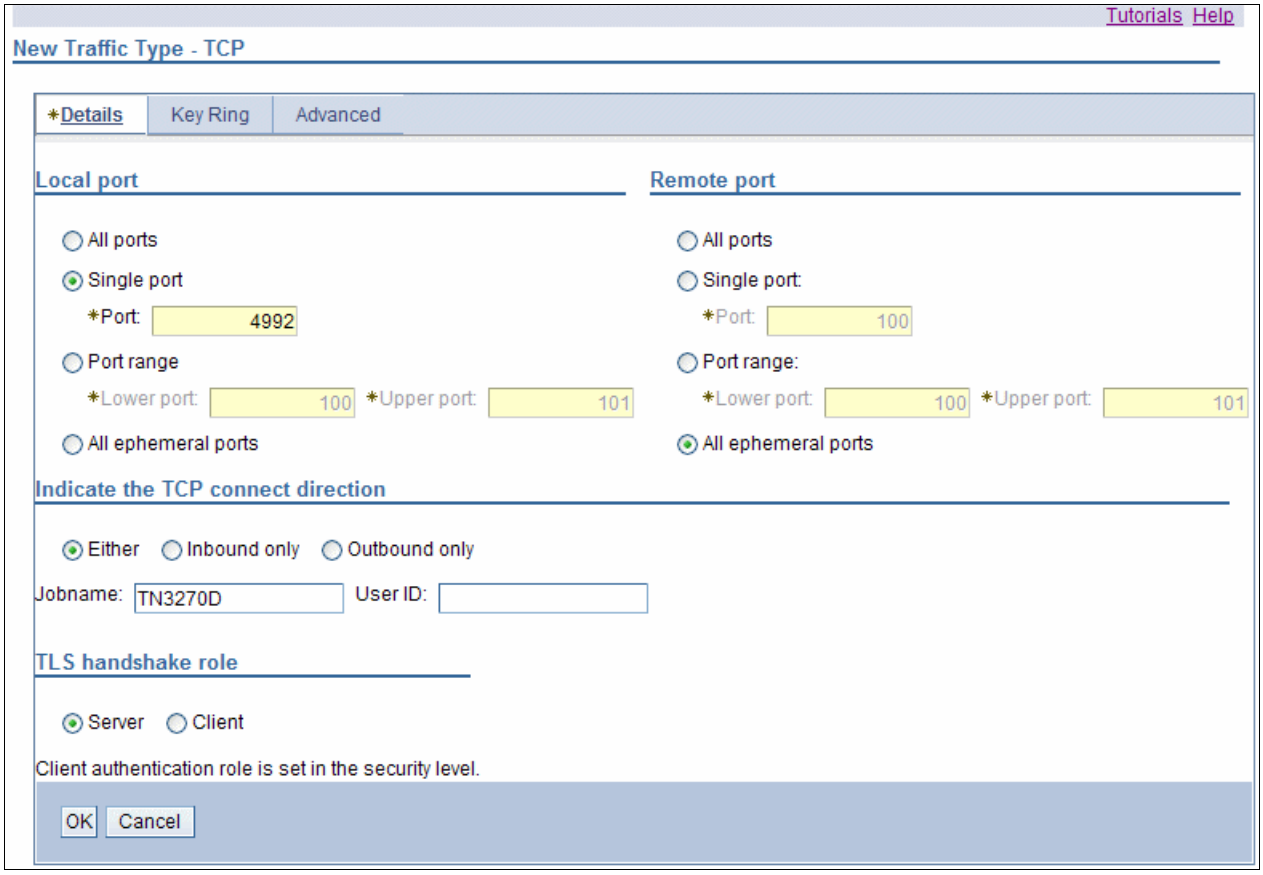


Figure 16-9 New Traffic Type panel: Defining the traffic description

13. In the Advanced tab (Figure 16-10), set the application controlled option to **On**. Click **OK**.

> **Tip:** 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 on the AT-TLS Tuning tab.

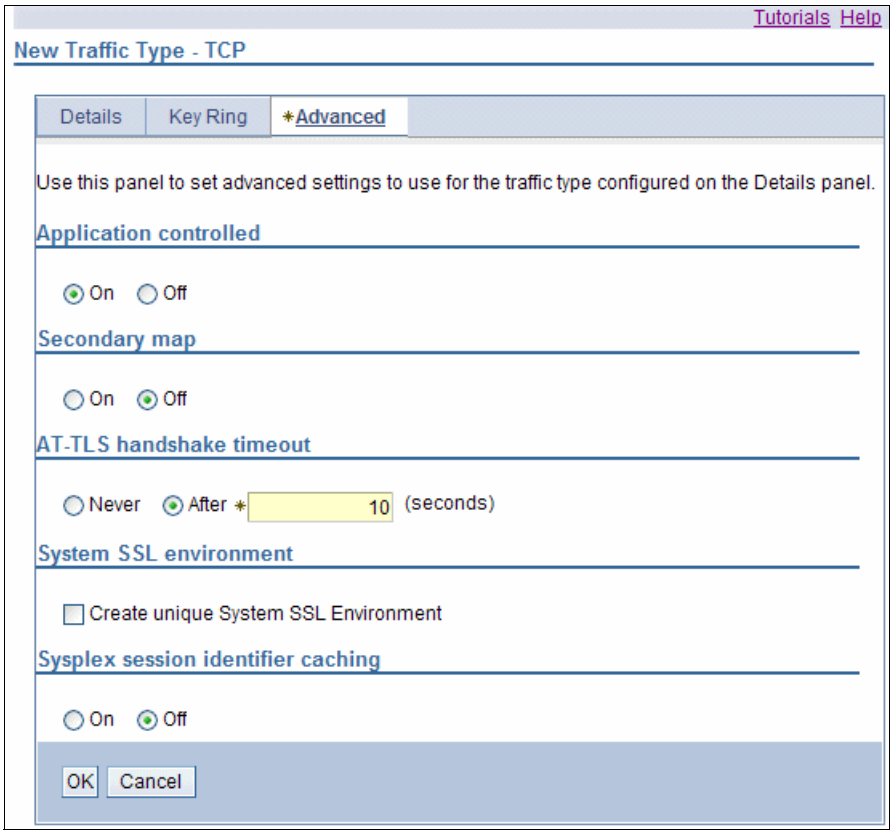


Figure 16-10 Key Ring and Advanced AT-TLS settings

14. The New Traffic Descriptor panel lists the entry that you just created. Click **OK**.
15. The Traffic Descriptor Objects panel lists the traffic descriptor entry that you created. Click **Close**.

16. Back in the Select Requirement Map panel, select the traffic descriptor that you created, as shown in Figure 16-11. In our example, we select **ATTLS_TN3270D_4992**.

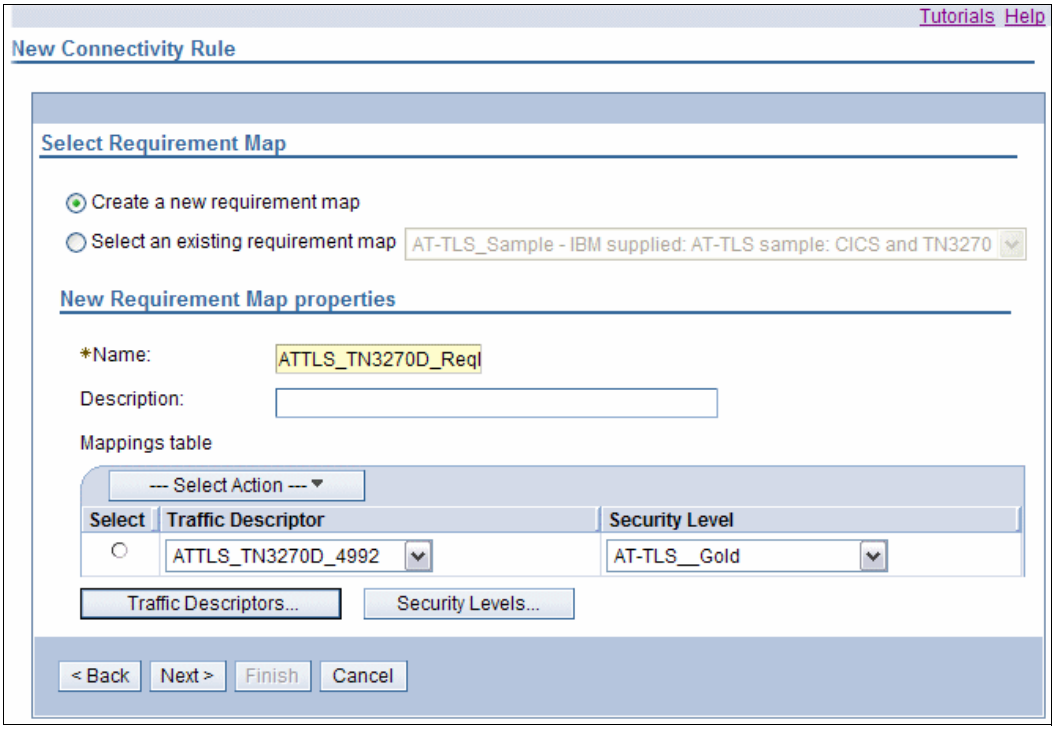


Figure 16-11 Select Requirement Map panel: Mapping traffic descriptor and security level

17. Select the AT-TLS Security Level in the mappings table. In our example, we select the **AT-TLS_Gold** option. Click **Next**. Table 16-5 shows the list of IBM-Supplied security levels.

Table 16-5 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

Tip: Select the security level depending on the supported ciphers of the host you establish the TLS connection with. You can also create a new security level to support the ciphers that fit your requirement by selecting the **Work with Security Levels** option.

18. In the New Connectivity Rule Advanced panel, you can set optional advanced settings. Click **Finish**.
19. The AT-TLS Perspective panel shows the defined connectivity rule, as shown in Figure 16-12, but there is an `Incomplete Image` message in the left pane because the key ring is not yet defined. Click **Apply Changes** to save the configuration.

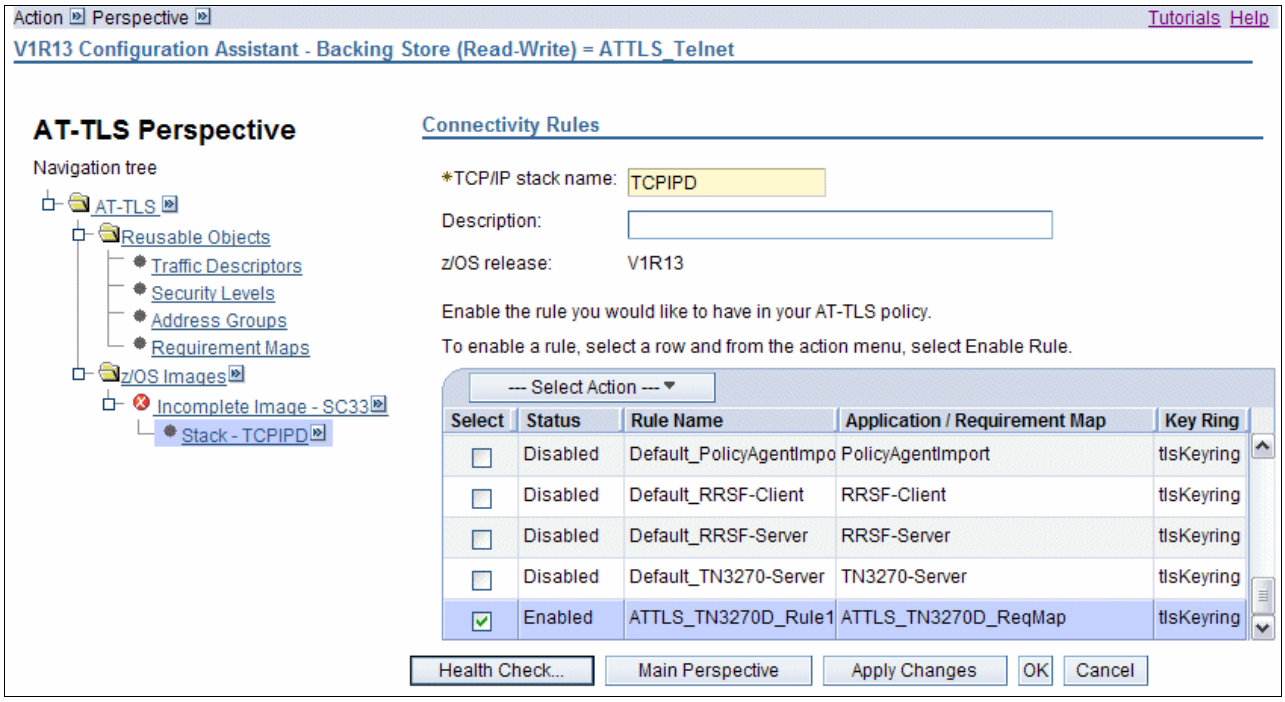


Figure 16-12 AT-TLS Perspective panel with defined connectivity rule

20. Select the image name (SC33, in our case), and then select **Settings** tab. Specify the key ring name or the key database name that you have created in the step “Create a certificate” on page 697. In our example, we specify SharedRing1 as shown in Figure 16-13. Click **OK**.

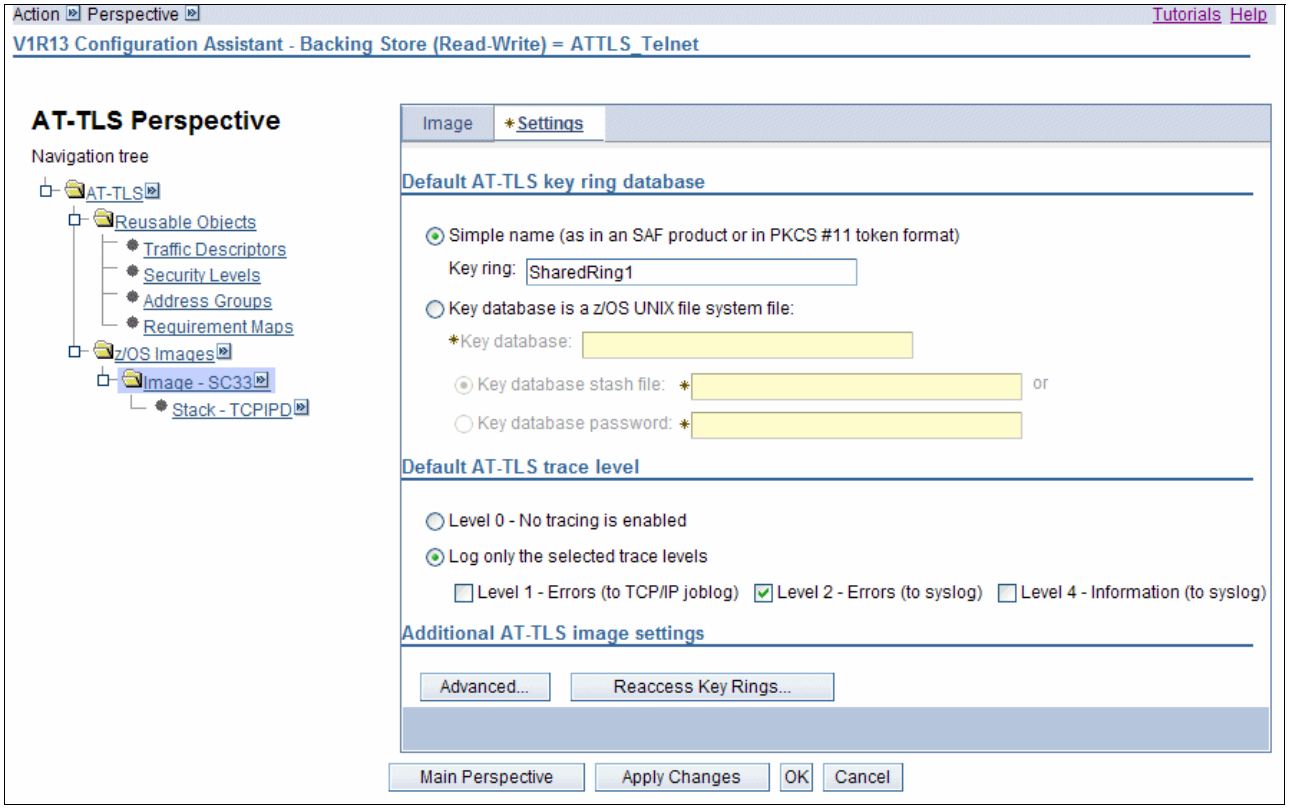


Figure 16-13 AT-TLS Image Level Settings panel: Specifying the key ring

21. At this point, the policy definition is complete. Example 16-8 shows the configuration file generated by the z/OSMF Configuration Assistant.

Example 16-8 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 13
## Backing Store = ATTLS_Telnet_V1R13_BackingStore
## FTP History:
## 2011-07-18 18:01:31 : Save To Disk
##
## End of Configuration Assistant information
TTLSRule ATTLS_TN3270D_Rule1~1
{
LocalAddr ALL
RemoteAddr ALL
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
CtraceClearText Off
Trace 2
}
TTLSConnectionAdvancedParms cAdv1~ATTLS_TN3270D_4992
{
ApplicationControlled On
SecondaryMap Off
}
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
}
PortRange portR1
{
Port 4992
}
PortRange portR2
{
Port 1024-65535
}
```

Save the policy to z/OS

To upload the policy, complete the following steps:

1. In the Configuration Assistant navigation tree, click he small symbol to the right of TCP/IP stack name (`Stack - TCPIPD` in our example), and click **Install Configuration Files** as shown in Figure 16-14.

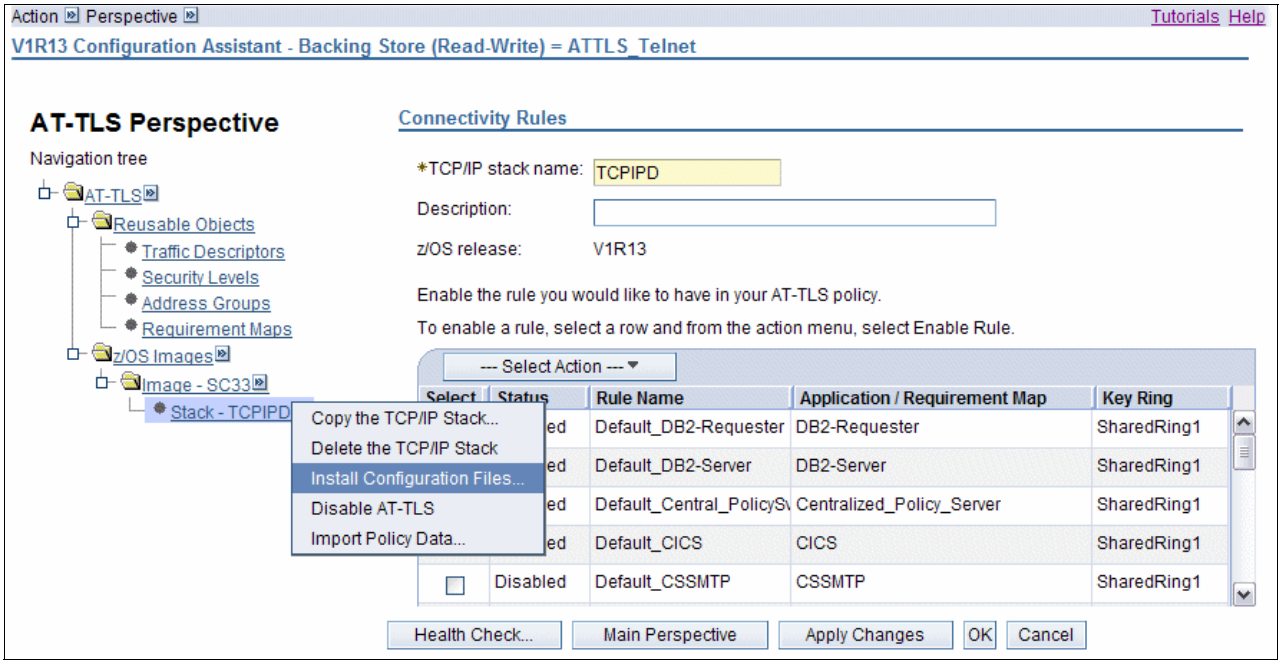


Figure 16-14 AT-TLS Perspective panel with defined policy

2. In the list (Figure 16-15), select the policy that we just created and click **Select Action** →**Install**.

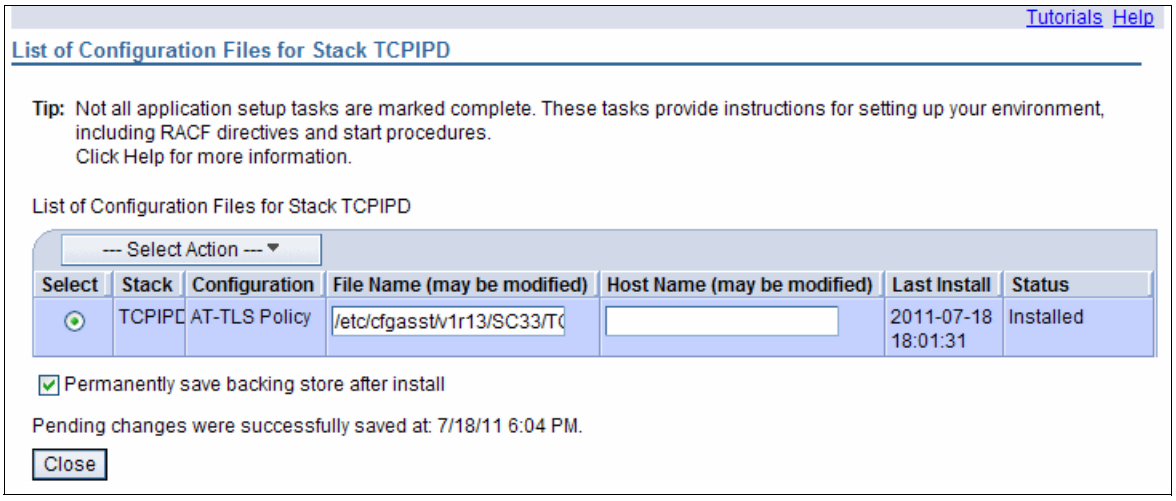


Figure 16-15 Installation panel: Installing Configuration File

3. Specify the file name to be saved as, and select Save to disk to save the policy to z/OS local file system (Figure 16-16 on page 710). Another option is to use FTP to the other z/OS image. In that case, enter the FTP server IP address, port number, FTP user ID, and password. Click Go.

4. We specify the file name to be saved as, and select Save to disk to save the policy to z/OS local file system, as shown in Figure 16-16. Another option is to FTP to the other z/OS image. In that case, enter the FTP server IP address, port number, FTP user ID, and password. Click Go.
5. When the save or transfer is complete, its result is shown at the bottom of the same panel. Click **Close**. You can optionally leave a comment, then click **OK**. Back in the List of Configuration Files for Stack TCPIPD panel, click **Close**.

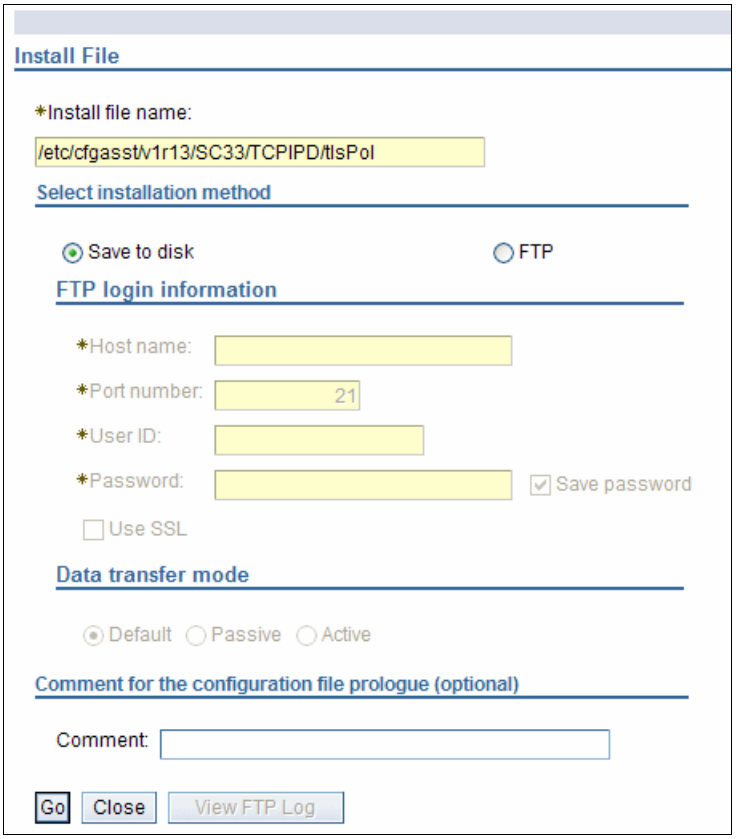


Figure 16-16 Installing files to remote host

Modify the policy agent configuration file

Example 16-9 shows the main configuration file of the policy agent. It defines the stack-specific configuration file name for the TCPIPD stack.

Example 16-9 Main configuration file of policy agent

```
#  ******************************************************************
# /SC33/etc/pagent.sc33.conf
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent.sc33.tcpipd.conf  FLUSH PURGE 600
```

Example 16-10 shows the stack-specific configuration file for the TCPIPD stack. Add the TTLSConfig statement that points to the policy configuration file we created.

Example 16-10 Stack-specific configuration file of policy agent

```
# This file /etc/pagent.sc33.tcpipd.conf contains                  #
# image statements for TCP/IP stack tcpipd in z/OS image sc33
####################################################################
# A policy for TN3270D for AT-TLS support                           #
####################################################################
TTLSConfig /etc/cfgasst/v1r13/SC33/TCPIPD/tlsPol
```

Define the AT-TLS port in the TN3270 configuration file

To enable AT-TLS security, specify the TTLSPORT statement. As shown in Example 16-11, TCP port 4992 is used for accepting the secure connections with AT-TLS.

Example 16-11 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
```

In this example, the numbers correspond to the following information:

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.

16.5.2 Activating and verifying TN3270 AT-TLS support

To activate and verify TN3270 AT-TLS support, complete the following steps:

1. Start the policy agent.
2. Start the TN3270E server.
3. Display PROF to show profile AT-TLS information.
4. Display the AT-TLS profile using the **pasearch** command.
5. Display CONN to show connection AT-TLS information.
6. 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 16-12. Message **1** shows the AT-TLS policy is processed and now in effect.

Example 16-12 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
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCPIPD
```

If you have the policy agent started already, use the **F *jobname*,REFRESH** command as shown in Example 16-13. Message **2** informs you that the AT-TLS policy is loaded (or reloaded).

Example 16-13 Refreshing the policy agent

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPD : TTLS 2
```

Start the TN3270E server

Start the TN3270E server as shown in Example 16-14. Ensure that TN3270E server starts listening on TTLS port **1**.

Example 16-14 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 as shown in Example 16-15. Use the DISPLAY TELNET,PROFILE command to display the profile.

Example 16-15 Telnet Profile Display

```
D TCPIP,TN3270D,T,PROF,PORT=4992,DETAIL
EZZ6080I TN3270D PROFILE DISPLAY 104
  PERSIS   FUNCTION       DIA  SECURITY   TIMERS   MISC
 (LMTGCAK)(OPATSKTQSSHRT)(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/**N/A**          1
   CRLLDAPSERVER      NONE/TTLS/**N/A**
   ENCRYPTION         TTLS                  1
   CLIENTAUTH         TTLS                  1
   NOEXPRESSLOGON
   NONACUSERID
   SSLV2              TTLS
 TIMERS
   INACTIVE                0 (OFF)
   PROFILEINACTIVE      1800
   KEEPINACTIVE            0 (OFF)
   PRTINACTIVE             0 (OFF)
   SCANINTERVAL          120
   TIMEMARK              600
   SSLTIMEOUT           TTLS
...
   KEYRING            SAF TCPIP/SharedRing1 2
```

In this example, the numbers correspond to the following information:

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

You can use the **pasearch** command to display the AT-TLS policy configuration as shown in Example 16-16.

Example 16-16 The pasearch -t display

```
CS02 @ SC33:/u/cs02>pasearch -t -p TCPIPD

TCP/IP pasearch CS V1R13                Image Name: TCPIPD
  Date:                07/18/2011       Time:  15:53:41
  TTLS Instance Id:    1311018790

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:           All
  ToAddr:             All
 Remote Address:
  FromAddr:           All
  ToAddr:             All
 LocalPortFrom:       4992              LocalPortTo:       4992
 RemotePortFrom:      1024              RemotePortTo:      65535
 JobName:             TN3270D           UserId:
 ServiceDirection:    Both
Policy created: Mon Jul 18 15:53:10 2011
Policy updated: Mon Jul 18 15:53:10 2011

TTLS Action:                  gAct1~ATTLS_TN3270D_4992
  Version:                    3
  Status:                     Active
  Scope:                      Group
  TTLSEnabled:                On
  CtraceClearText:            Off
  Trace:                      2
  FIPS140:                    Off
  TTLSGroupAdvancedParms:
   SecondaryMap:              Off
   SyslogFacility:            Daemon
  Policy created: Mon Jul 18 15:53:10 2011
  Policy updated: Mon Jul 18 15:53:10 2011

TTLS Action:                  eAct1~ATTLS_TN3270D_4992
  Version:                    3
  Status:                     Active
  Scope:                      Environment
  HandshakeRole:              Server
  TTLSKeyringParms:
   Keyring:                   SharedRing1
  TTLSEnvironmentAdvancedParms:
   SSLv2:                     Off
   SSLv3:                     On
```

```
    TLSv1:                    On
    TLSv1.1:                  On
    ApplicationControlled:    Off
    HandshakeTimeout:         10
    ClientAuthType:           Required
    ResetCipherTimer:         0
    ResetCipherTimer:         0
    TruncatedHMAC:            Off
    CertValidationMode:       Any
    ServerMaxSSLFragment:     Off
    ClientMaxSSLFragment:     Off
    ServerHandshakeSNI:       Off
    ClientHandshakeSNI:       Off
   EnvironmentUserInstance:   0
   Policy created: Mon Jul 18 15:53:10 2011
   Policy updated: Mon Jul 18 15:53:10 2011

 TTLS Action:                 cAct1~ATTLS_TN3270D_4992
   Version:                   3
   Status:                    Active
   Scope:                     Connection
   HandshakeRole:             Server
   CtraceClearText:           Off
   Trace:                     2
   TTLSConnectionAdvancedParms:
    SecondaryMap:             Off
    ApplicationControlled:    On
   TTLSCipherParms:
    v3CipherSuites:
     0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
     2F  TLS_RSA_WITH_AES_128_CBC_SHA
   Policy created: Mon Jul 18 15:53:10 2011
   Policy updated: Mon Jul 18 15:53:10 2011
```

Display CONN to show connection AT-TLS information

We use IBM Personal Communications V5.7 to establish a secure session. Because we used self-signed certificate, we downloaded and installed the certificate of certificate authority into the Personal Communications. We defined the Personal Communications connection profile for the AT-TLS connection with exactly the same security parameters that we defined for the TN3270 native TLS secure connection. Example 16-17 shows a secure connection.

Example 16-17 TLS secure connection using port 4992

```
D TCPIP,TN3270D,T,CONN
EZZ6064I TN3270D CONN DISPLAY 590
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00004A6D 0A ::FFFF:10.1.100.222..1355
                                   SC33DT01           TPE
----- PORT:  4992  ACTIVE           PROF: CURR CONNS:      1
------------------------------------------------------------
```

In this example, AT-TLS port 4992 is used.

Example 16-18 shows the same D TCPIP command using a parameter that requests more detail.

Example 16-18 TLS secure connection using port 4992 - detail

```
D TCPIP,TN3270D,T,CONN,CONN=4A6D
EZZ6065I TN3270D CONN DISPLAY 595
  CONNECTED: 16:16:25  07/18/2011  STATUS: SESSION PENDING
  CLIENT IDENTIFIER FOR CONN: 00004A6D   SECLABEL: **N/A**
    CLIENTAUTH USERID: **N/A**
    HOSTNAME: NO HOSTNAME
    CLNTIP..PORT: ::FFFF:10.1.100.222..1355
    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: SC33DT01
  APPL: **N/A**
    USERIDS   RESTRICTAPPL: **N/A**   EXPRESSLOGON: **N/A**
    LOGMODES  TN REQUESTED:           APPL SPECIFIED:
  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)(OPATSKTQSSHRT)(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
42 OF 42 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

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 16-19.

Example 16-19 Display Netstat TTLS command - AT-TLS information

```
D TCPIP,TCPIPD,N,TTLS,CONN=4A6D,DETAIL
EZD0101I NETSTAT CS V1R13 TCPIPD 607
CONNID: 00004A6D
  JOBNAME:      TN3270D
  LOCALSOCKET:  ::FFFF:10.1.1.40..4992
  REMOTESOCKET: ::FFFF:10.1.100.222..1355
  SECLEVEL:     TLS VERSION 1
  CIPHER:       0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
TTLSRULE: ATTLS_TN3270D_RULE1~1
  PRIORITY:       255
  LOCALADDR:      ALL
  LOCALPORT:      4992
  REMOTEADDR:     ALL
  REMOTEPORTFROM: 1024               REMOTEPORTTO: 65535
  JOBNAME:        TN3270D
  DIRECTION:      BOTH
  TTLSGRPACTION:  GACT1~ATTLS_TN3270D_4992
    GROUPID:                    00000004
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
    FIPS140:                    OFF
  TTLSENVACTION:  EACT1~ATTLS_TN3270D_4992
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    SHAREDRING1
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    TLSV1.1:                    ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
    TRUNCATEDHMAC:              OFF
    CLIENTMAXSSLFRAGMENT:       OFF
    SERVERMAXSSLFRAGMENT:       OFF
    CLIENTHANDSHAKESNI:         OFF
    SERVERHANDSHAKESNI:         OFF
    CLIENTAUTHTYPE:             REQUIRED
    CERTVALIDATIONMODE:         ANY
  TTLSCONNACTION: CACT1~ATTLS_TN3270D_4992
    HANDSHAKEROLE:              SERVER
    V3CIPHERSUITES:             0A TLS_RSA_WITH_3DES_EDE_CBC_SHA
                                2F TLS_RSA_WITH_AES_128_CBC_SHA
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    APPLICATIONCONTROLLED:      ON
    SECONDARYMAP:               OFF
```

16.6 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 to `On` in `TTLSEnvironmentAdvancedParms`. Be sure the file generated by the z/OSMF Configuration Assistant 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 click **Connectivity Rule** → **Additional Settings** → **Advanced** → **Set Tracing, Timing, or Tuning** in the z/OSMF Configuration Assistant. The output is written to the syslog.

- TN3270E server trace

  TelnetParms Debug parameter

See “Problem determination for Telnet server” in *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997 and *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782 for further information.

16.7 Additional information sources for TN3270 AT-TLS support

Detailed information about the TN3270E Telnet server and protocol can be found in the following sources:

- *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, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

17

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 *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997, where basic, non-secure FTP scenarios are discussed. The assumption here is that you already have FTP clients and servers in your environment.

We discuss the following topics in this chapter.

Section	Topic
17.1, “Conceptual overview of FTP security” on page 720	General security concepts for FTP
17.2, “FTP client with SOCKS proxy protocol” on page 721	How to enable a client to support an FTP SOCKS proxy server
17.3, “FTP with native TLS security support” on page 724	How to configure and test FTP’s native TLS security
17.4, “FTP with AT-TLS security support” on page 769	How to configure and test FTP’s support of AT-TLS through the policy agent
17.5, “Migrating from native FTP TLS to FTP AT-TLS” on page 798	Summarizes migration requirements for migrating from an existing native TLS environment to one with AT-TLS
17.6, “FTP TLS and AT-TLS problem determination” on page 800	Problem determination procedures for FTP security
17.7, “Additional information” on page 801	References to additional information about FTP security

17.1 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 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 Transport Layer Security (AT-TLS) features supported by the z/OS FTP application.

We discuss the following topics in this section:

- What is FTP security
- How FTP works
- How FTP security can be applied

17.1.1 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 that 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, which 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, although 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.

Tip: 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.

17.1.2 How FTP security works

SOCKS proxy protocols enable an FTP client to access a remote FTP server, protected by a blocking mechanism, that 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. Therefore 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.

17.1.3 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 IBM Configuration Assistant GUI is used to create the appropriate policy agent statements for AT-TLS.

17.2 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 *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997. 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

17.2.1 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 data set.

Using the FTP client with SOCKS allows users to contact FTP servers that are protected by a SOCKS firewall that are otherwise unreachable.

17.2.2 Configuration of SOCKS proxy protocol

To configure the client for SOCKS support, complete 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 data set. 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 17-1 shows configuration statements.

Example 17-1 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 17-1 on page 722:

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 about the use of the statement parameters, see *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)'
```

17.2.3 Activation and verification of the SOCKS proxy FTP

To use SOCKS proxy protocol with your FTP client, complete the following steps:

- 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. The SOCKS server should be used to get there as shown by the following for example:

```
FTP 172.14.1.20
```

Connect to your local loopback address and you should connect directly without SOCKS assistance:

```
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.

17.3 FTP with native TLS security support

In this section we discuss the security parameters to be added to the basic FTP design described in *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997, 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 AT-TLS or IP Security (IPSec). This section only discusses the TLS security built into FTP. For details about securing FTP using AT-TLS, see 17.4, “FTP with AT-TLS security support” on page 769. AT-TLS and IPSec are discussed in detail in their respective chapters in this book.

Tip: 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“

You need to understand the difference between *secure FTP* and the OpenShell procedure known as *sftp*. The sftp procedure operates over an encrypted secured shell (ssh) transport and does not use FTP protocols as described in RFC959 and its related RFCs.

We discuss the following topics in this section:

- Description of FTP native TLS security
- Configuration of FTP native TLS security
- Activation and verification of FTP server without security
- Activation and verification of the FTP server with TLS security: Internet draft protocols
- Activation, verification of FTP server with TLS security: RFC4217 protocols
- Implicit secure TLS login

17.3.1 Description of FTP native TLS security

We now expand on our base FTP server (introduced in *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997) 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 both implicit and explicit TLS. See 17.3.6, “Implicit secure TLS login” on page 760 for information about how to implement implicit TLS connection.

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 39.

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 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 17.3, “FTP with native TLS security support” on page 724, we test with both the DRAFT version of TLS and the RFC 4217 version of TLS.

17.3.2 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 *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997.

We create a server certificate, which might be a self-signed server certificate, a personal server, or a site certificate that has been signed by a certificate authority. The client needs access to a key ring on the client’s system and can 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

Generally, you should use 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 `CA`. The commands can 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. See Chapter 3, “Certificate management in z/OS”

on page 39 for the details about 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.

We completed the following RACF tasks 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 key ring

Certain RACF classes must be active before rings and certificates are added with the RACDCERT command as follows:

```
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 as follows:

```
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 01 November 2014. You might want to change the time frame in which the certificate is to be valid. After the certificate creation, refresh the facility and list the generated CA certificate.

Example 17-2 Using RADCERT to assign an expiration date

```
RACDCERT CERTAUTH GENCERT                          -
  SUBJECTSDN( O('IBM CORPORATION')                 -
              OU('ITSO CERTIFICATE AUTHORITY')     -
               C('US'))                            -
  NOTBEFORE(DATE(2010-11-01))                      -   1
  NOTAFTER(DATE(2014-11-01))                       -   1
  KEYUSAGE(CERTSIGN)                               -
  WITHLABEL('CS ITSO CA1')                             2
SETROPTS RACLIST(DIGTCERT) REFRESH
RACDCERT CERTAUTH LIST(LABEL('CS ITSO CA1'))
```

In this example, the numbers correspond to the following information:

1. Define time frame for this certificate

2. Specify the label for the certificate as `CS ITSO CA1`.

Generate a RACF SITE certificate

A USER certificate is associated with a user ID and can be used only by a single process: server process or client process. A SITE certificate is also associated with a user ID, but it can be shared among several users: servers, clients, or both. A shared SITE certificate might be considered a security exposure because it is not distinctly associated with a single process.

Tip: If your enterprise needs more stringent security controls than a shared site certificate offers, use separate USER certificates for each server and client. You should create an environment that meets your organization's requirements.

Example 17-3 shows the set of RACF commands that we executed to create a shared SITE certificate for our implementation of FTP. Notice the SITE certificate is signed with the CA certificate just created.

Example 17-3 Creating a shared SITE certificate

```
RACDCERT SITE GENCERT                              -
   SUBJECTSDN(CN('ITSO.IBM.COM')                   -
   O('IBM CORPORATION')                            -
   OU('ITSO Shared Site')                          -
   C('US'))                                        -
   NOTBEFORE(DATE(2010-11-01))                     -   1
   NOTAFTER(DATE(2014-11-01))                      -   1
   WITHLABEL('CS ITSO SharedSite1')                -
   SIGNWITH(CERTAUTH LABEL('CS ITSO CA1')              2
RACDCERT SITE LIST(LABEL('CS ITSO SharedSite1'))
```

In this example, the numbers correspond to the following information:

1. Define time frame for this certificate.
2. Sign with CA certificate '`CS ITSO CA1`'.

Connect both certificates to the shared key ring

Execute the RACF commands shown in Example 17-4 to connect the two certificates to the shared key ring.

Example 17-4 Connecting two certificates to the shared key ring

```
RACDCERT ID(TCPIP) -
     CONNECT(CERTAUTH LABEL('CS ITSO CA1') RING(SharedRing1) -
     USAGE(CERTAUTH))
RACDCERT ID(TCPIP) -
     CONNECT(SITE LABEL('CS 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), which 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, CS06, who can FTP into the FTP server.

```
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(TCPIP) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(CS06) 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(CS06) 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 to maintain certificate authority certificates and to store client certificates required if the server enforces client authentication.

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 CS06. 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(CS06)  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.

Tip: 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 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. See *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('CS ITSO CA1')) -
  FORMAT(CERTB64) DSN('TCPIP.ITSO.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 17-1 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.

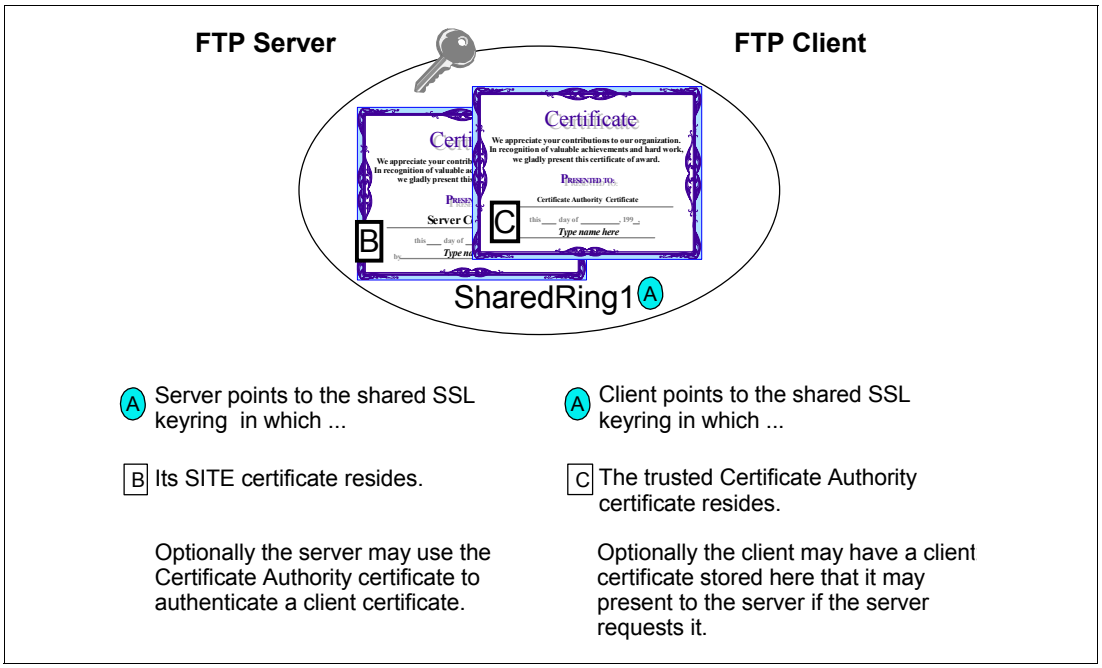


Figure 17-1 The shared key ring for FTP client and server

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 can request client authentication. In this case the client will send its certificate signed by a trusted certificate authority to the server. The client certificate can 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 17-5 shows the additional statements we added to our FTP.DATA data set to enable security.

Example 17-5 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
```

In this example, the numbers correspond to the following information:

1. Enables TLS with an EXTENSIONS AUTH_TLS statement.

2. Indicates native FTP TLS support (not AT-TLS)

> **Tip:** With FTP TLS, SSLv2 is not supported.

3. Adds 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. 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. Adds 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. Adds 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. Adds a SECURE_CTRLCONN PRIVATE statement. This statement is ignored unless the client requests a secure connection. Generally, code 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. Adds a SECURE_DATACONN CLEAR statement to allow the client to determine whether data traffic really needs encryption. Encryption carries a performance price and might not be necessary for every data transfer.
8. Takes the default of 100 on TLSTIMEOUT or code it. This specifies the maximum time between full TLS handshakes.
9. Adds a 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.

17.3.3 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 to verify that the chosen clients still work as expected with the revised FTP server.

The scenario we depict first is illustrated in Figure 17-2.

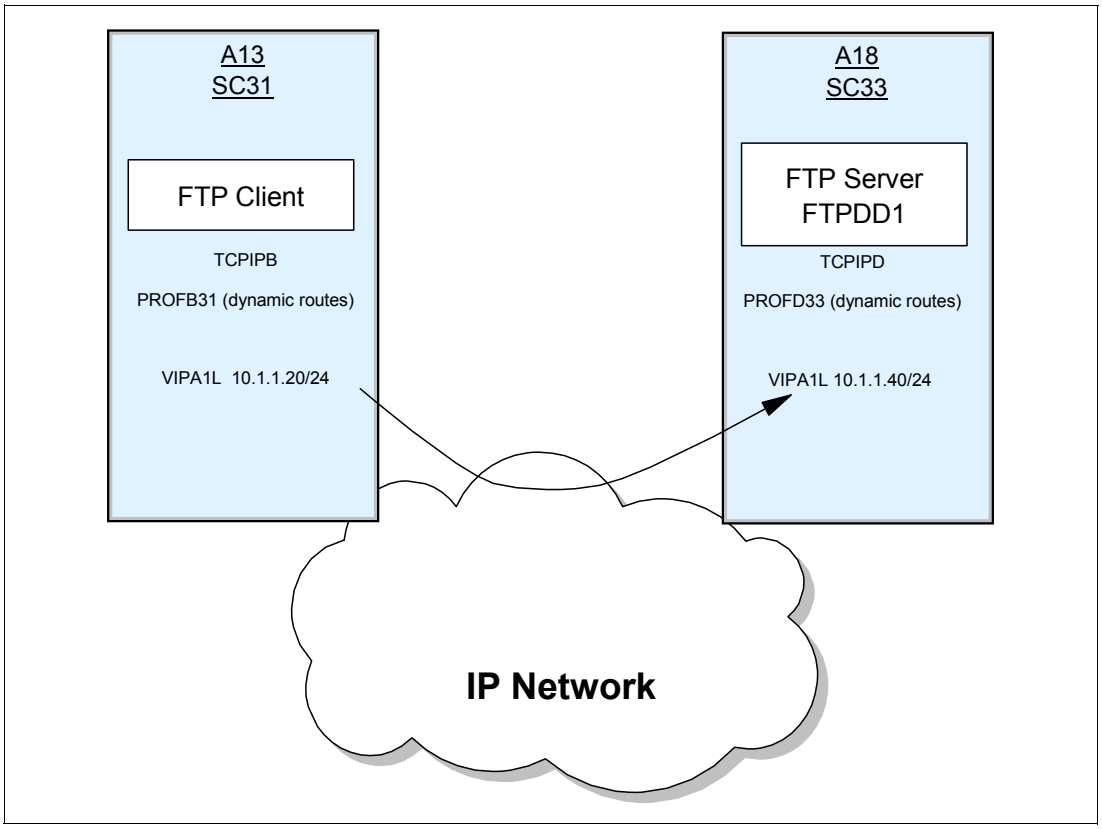


Figure 17-2 FTP Client connection to server: no security

There are a number of ways to verify that the FTP server has started and is working correctly. Perform the following tasks:

1. Check FTP server job log messages
2. Verify the non-TLS z/OS client can connect to the server
3. Display system job tasks for control and data connections
4. 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 FTP.DATA file that has been reconfigured for TLS:

- `p ftpdd1`
- `s ftpdd`

Our FTP procedure is copied after `hlq.TCPPARMS(FTPSPROC)` and begins as shown in Example 17-6.

Example 17-6 FTP procedure without _BPX_JOBNAME specified

```
//FTPDD  PROC MODULE='FTPD',PARMS='',
//        TCPDATA=DATAD&SYSCLONE.,FTPDATA=FTPSECD
//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 17-7 shows the EZY2702I message we received shortly after FTP was started.

Example 17-7 EZY2702I message received on successful startup of FTP

```
EZY2702I Server-FTP: Initialization completed at 11:14:06 on 08/02/11.
```

Example 17-8 shows that the FTP server has initialized in an address space named FTPDD1 and that it is owned by user ID TCPIP.

Example 17-8 D OMVS,A=ALL: FTPDD1 as a UNIX Address Space

```
D OMVS,A=ALL
BPXO070I 11.15.10 DISPLAY OMVS 263
OMVS     000F ACTIVE             OMVS=(3A)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS

TCPIP    FTPDD1   0055   33816660          1 1FI----- 11.14.06      .0
  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 17-9 on page 734 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 A13. 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
```

Example 17-9 shows the resulting output, prior to signing on with the user ID and password.

Example 17-9 FTP from one z/OS system to another z/OS system

```
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.ITSO.IBM.COM, 11:39:32 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.40:CS03):
```

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 17-10) from the **D OMVS,A=ALL** command that shows that we have address space 0029, 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 0029) 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 FTPDD5.

Example 17-10 OMVS address spaces and connection IDs for FTP control connections

```
D OMVS,A=ALL
BPXO070I 11.26.30 DISPLAY OMVS 295
OMVS     000F ACTIVE             OMVS=(3A)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
......
TCPIP    FTPDD1   0055   33816660          1 1F------ 11.14.06      .0
  LATCHWAITPID=         0 CMD=FTPD
.....
TCPIP    FTPDD5   0029   67371161   33816660 1FI----- 11.39.32      .0
  LATCHWAITPID=         0 CMD=FTPD

D TCPIP,TCPIPD,N,CONN
FTPDD1   00000030 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00000082 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1032
```

At TCPIPD in Example 17-10 we also see, for this connection, the output of the **D TCPIP,TCPIPD,N,CONN** netstat command. Jobname FTPDD1 shows two connections: one on port 21 in a listen state that represents the listener task, and one established connection between local port 21 and 10.1.1.20 that represents the established connection for our client.

At the client, TCPIPB, we next enter the user ID and password for our client user as shown in Example 17-11.

Example 17-11 Entering user ID and password after initial connection

```
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.ITSO.IBM.COM, 11:39:32 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.40:CS03):
EZA1701I >>> USER CS03
331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS
230 CS03 is logged on.  Working directory is "CS03.".
EZA1460I Command:
```

Example 17-12 shows the output after our user, CS03, has initially established a connection and entered a user ID and password.

Example 17-12 Change to security context of control connection

```
D OMVS,A=ALL
BPXO070I 11.40.16 DISPLAY OMVS 302
OMVS     000F ACTIVE             OMVS=(3A)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS

TCPIP    FTPDD1   0055   33816660          1 1FI----- 11.14.06      .0
  LATCHWAITPID=         0 CMD=FTPD

CS03 a   CS03 b   0029   67371161   33816660 1FI----- 11.39.32      .0
  LATCHWAITPID=         0 CMD=/usr/sbin/ftpdns 2137233224
```

Notice in Example 17-12 that address space 29 is now running under the security context of CS03 (a) and no longer under that of the superuser TCPIP. The JOBNAME has also changed to that of the user: CS03 (b). In this version of the FTP server JCL, we did not specify _BPX_JOBNAME as an environment variable, as you see in Example 17-6 on page 733.

Our user, CS03, 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 17-13.

Example 17-13 Client command, STAT, shows operating parameters of server

```
EZA1701I >>> STAT
211-Server FTP talking to host ::ffff:10.1.1.20, port 1033
211-User: CS03  Working directory: CS03.
211-The control connection has transferred 110 bytes
211-There is no current data connection.
211-The next data connection will be actively opened
211-to host ::ffff:10.1.1.20, port 1033,
....
211-TLS security is supported at the DRAFT level a
```

Example 17-14 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 10.1.1.20 that represents the established connection for our client.

Example 17-14 Control port connections: listening socket and client connection

```
D TCPIP,TCPIPD,N,CONN
USER ID  CONN     STATE
FTPDD1   00000030 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   0000022F ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1033
```

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 (Example 17-15).

Example 17-15 FTP data connection opened

```
D TCPIP,TCPIPD,N,CONN
USER ID  CONN     STATE
CS03     00000244 FINWT2
  LOCAL SOCKET:   ::FFFF:10.1.1.40..20
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1034
FTPDD1   00000030 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   0000022F ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1033
```

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 17-16.

Example 17-16 _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 17-17.

Example 17-17 Output from DA when FTP Server is started with _BPX_JOBNAME

```
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
b    FTPDD3   STEP1             STC08636 TCPIP  c   LO  FF  724   0.00   0.00
     FTPDD1   STEP1             STC08035 TCPIP      LO  FF  721   0.00   0.00
```

You see at b in Example 17-17 that the user address space has a jobname of FTPDD3 and it is still associated with the security context of user ID TCPIP c. The **netstat conn** command displays the listening connection and the established connection to the server at FTPDD1 in Example 17-18.

Example 17-18 FTP listening and established socket connections

```
USER ID  CONN     STATE
FTPDD1   00000254 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00000267 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1036
```

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 CS03 (e in Example 17-19). The jobname of the user's address space changes to FTPDDD. The name is what we coded for the _BPX_JOBNAME environment variable.

Example 17-19 DA output: Change of security context when user logs in

```
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
d    FTPDDDD  STEP1             STC08636 CS03   e   LO  FF  847   0.00  14.45
     FTPDD1   STEP1             STC08035 TCPIP      LO  FF  721   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 (f in Example 17-20).

Example 17-20 Data connection for user: Address space name

```
FTPDDDD  00000285 FINWT2           f
  LOCAL SOCKET:   ::FFFF:10.1.1.40..20
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1037
FTPDD1   00000254 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
FTPDD1   00000267 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1036
```

Many installations find it useful to have a similar job name for all FTP forked tasks because it helps with job accounting routines, WLM service class assignments, and isolation of syslogd messages. With _BPX_JOBNAME, the data connection for each FTP logged-in user remains the jobname of *bpxjobnamex*, instead of the jobname of its user ID.

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-project.org/`

Figure 17-3 shows our scenario.

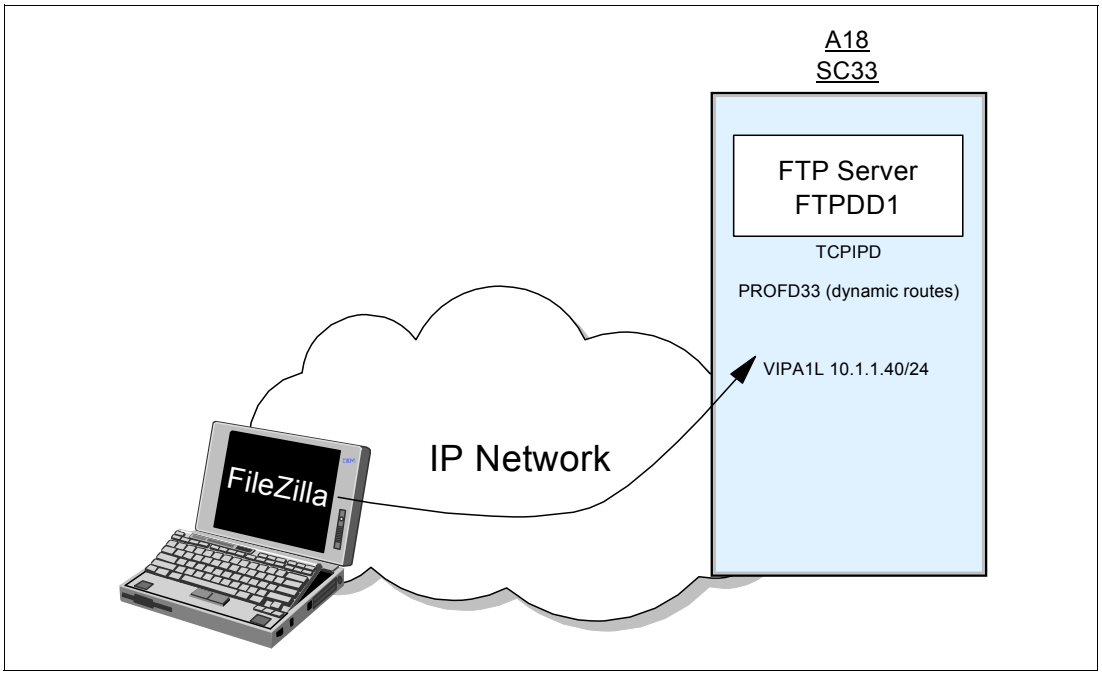


Figure 17-3 Workstation client FTP to z/OS FTP: No security

FileZilla can be configured for secure or non-secure FTP. After installing the package on our workstation, we used the File menu for FileZilla to enter the Site Manager window. We then created a site for the standard FTP on this window as shown in Figure 17-4.

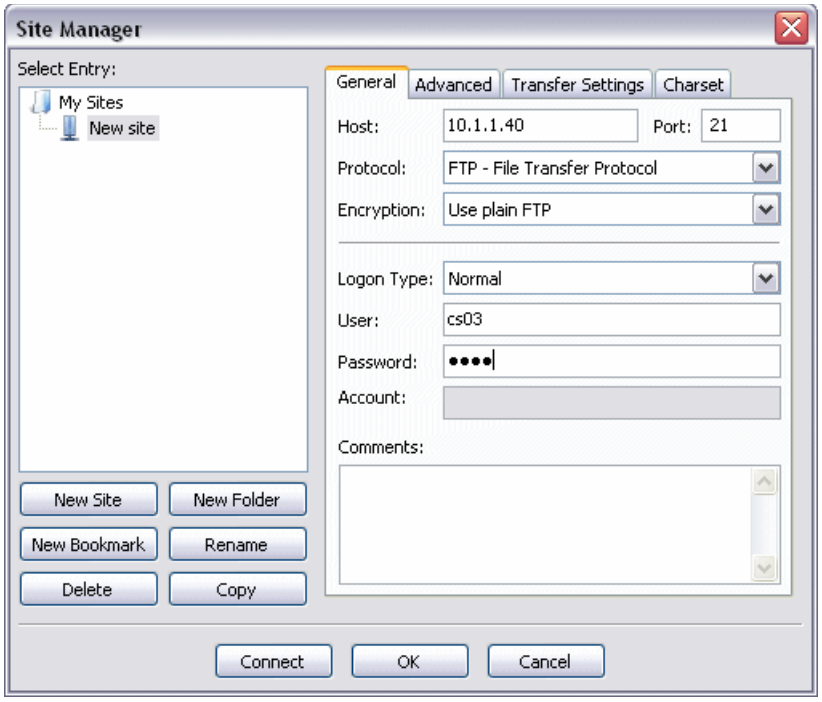


Figure 17-4 FileZilla: Basic FTP client configuration

We clicked **Connect** on the FileZilla window and opened both a control and data connection with the FTPDD server at TCPIPD. The results are shown in Figure 17-5.

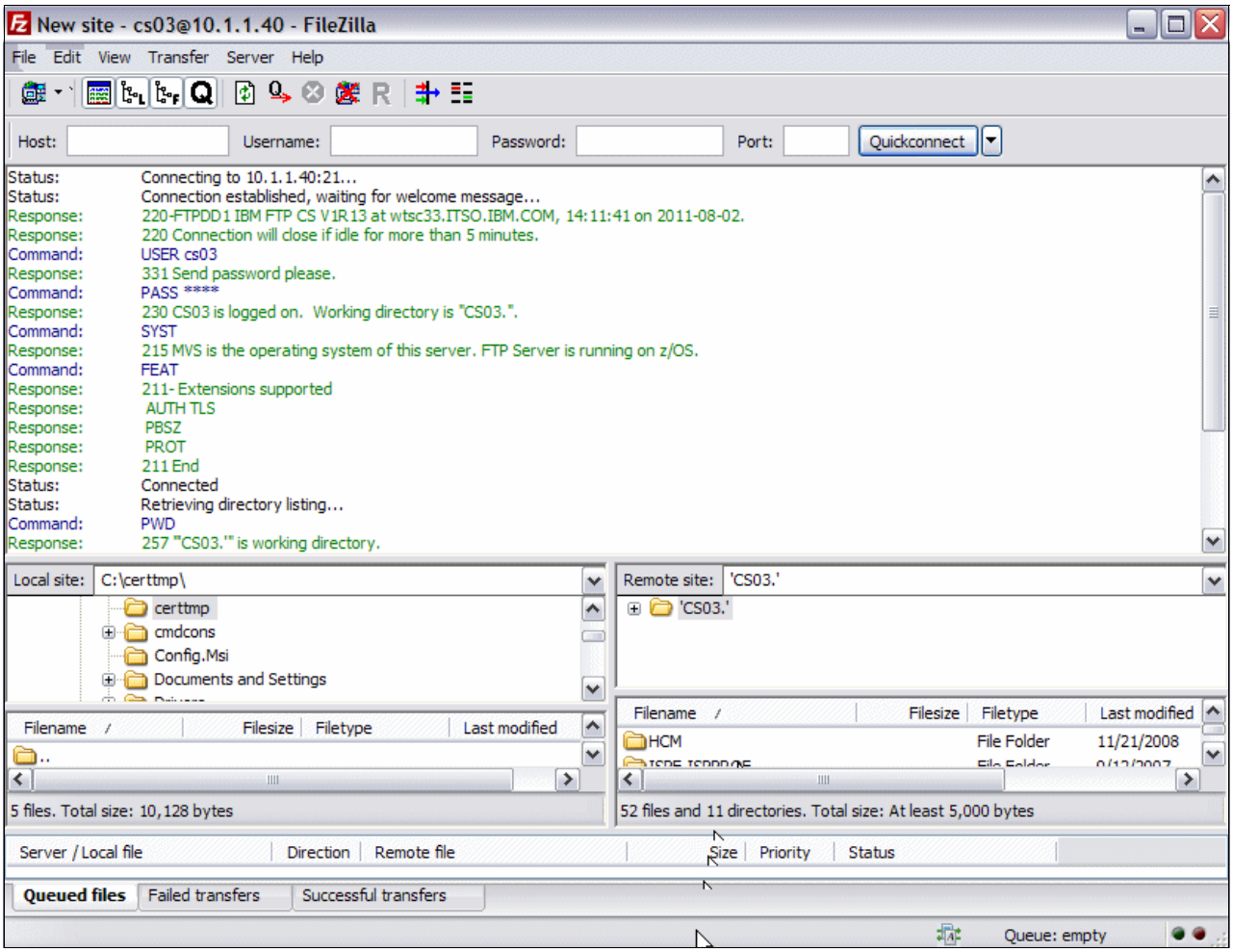


Figure 17-5 Successful FTP from FileZilla into TCPIPD

17.3.4 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 “Use a TLS-enabled workstation client to verify server TLS (RFC 4217)” on page 757.

The following tasks (as described in “Create key ring and certificates and update RACF permissions” on page 726 and “Update the FTP.DATA for the server” on page 731) have been performed already:

- “Check FTP server job log messages” on page 733
- “Verify the non-TLS z/OS client can connect to the server” on page 733
- “Display system job tasks for control and data connections” on page 734

In this section, we explain the following additional tasks:

- Use a TLS-enabled z/OS client to verify server TLS (Internet draft)
- Use a TLS-enabled workstation client to verify server TLS (Draft)

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 17-6.

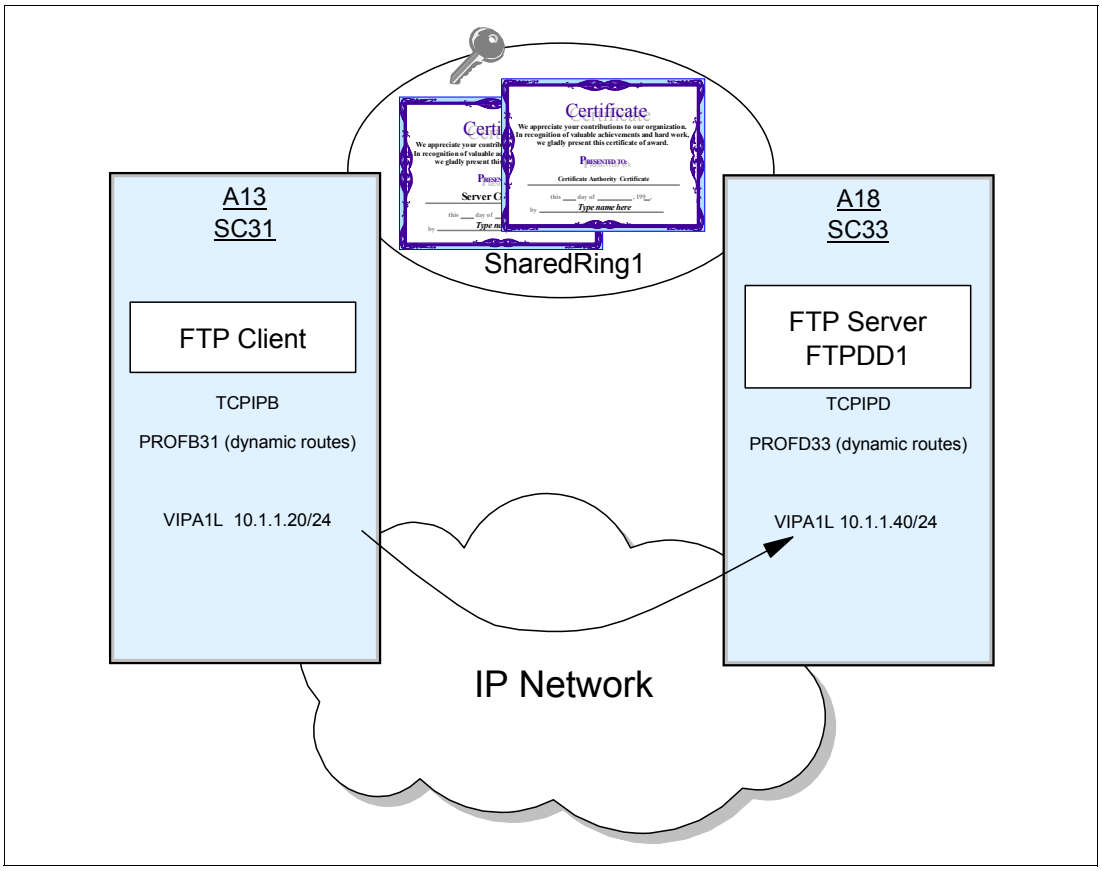


Figure 17-6 z/OS Client to FTP server: TLS Security

To set up TLS-enabled FTP session, we need to specify which key ring is used for both server and client FTP.DATA. Recall that the key ring is used to store the certificates of the signing authority (the certificate authority) and server certificates. It is also used to store client certificates that might be necessary for client authentication.

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` data set and rename it to `FTPCB31`.

Examine the security options in the sample file, and edit the client FTP.DATA file as shown in Example 17-21.

Example 17-21 Client FTP.DATA file changes

```
; ----------------------------------------------------------------------
; 7. Security options
; ----------------------------------------------------------------------

;SECURE_MECHANISM  TLS      a        ; 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 b
;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        c
;KEYRING           *AUTH*/*         d
;TLSTIMEOUT        100               ; Maximum time limit between full
;SECUREIMPLICITZOS TRUE              ; Specify whether client will
; ----------------------------------------------------------------------
```

In Example 17-21, we commented out the parameters that specify the type of security that we want (SECURE_), shown at a. 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 at b, 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.

Look at the KEYRING designation, shown at c. 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 which is owned by user TCPIP. You can also use the coding as KEYRING *AUTH*/* as shown at d. 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. See 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 execute the command again to establish the secure connection, as shown in Example 17-22 on page 743:

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" -a TLS 10.1.1.40
```

This command uses the following parameters:

-f Used to point to the FTP.DATA file. You must specify the FTP client FTP.DATA, which contains the KEYRING definition, when setting up the TLS-enabled FTP session.

-a TLS Specifies that the client requests secured communication with the server FTPDD. When SECURE_FTP REQUIRED is defined in FTP server's FTP.DATA, you must use **-a TLS** in the **ftp** log in command. Otherwise, login fails with the following message:

```
534 Server requires authentication before USER command
```

An alternate method to request TLS secure log in is to specify SECURE_MECHANISM TLS in the FTP client's FTP.DATA.

When TLSPORT *portnum* operand is defined in FTP client's FTP.DATA, do not use **-a TLS** in the **ftp** log in command. Otherwise, the following message is issued because you specified the **-a** parameter while trying to connect to the secure port:

```
EZA2892I Secure port 21 does not allow the -a or -r start parameter
```

Example 17-22 Establishing a secure connection using a specific FTP.DATA file

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" -a TLS 10.1.1.40
EZY2640I Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration paramete
         rs.
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.ITSO.IBM.COM, 15:00:07 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1701I >>> AUTH TLS
234 Security environment established - ready for negotiation
EZA2895I Authentication negotiation succeeded
EZA1459I NAME (10.1.1.40:CS03):

EZA1701I >>> USER CS03
331 Send password please.
EZA1789I PASSWORD:


EZA1701I >>> PASS
230 CS03 is logged on.  Working directory is "CS03.".
EZA1460I Command:
```

After issuing the **ftp** login command, Example 17-22 shows that the negotiation has succeeded.

Next, we entered the **status** subcommand from the client to verify that we truly had a TLS connection as shown in Example 17-23.

Example 17-23 FTP client messages

```
status
EZA1701I >>> STAT
211-Server FTP talking to host ::ffff:10.1.1.20, port 1033
211-User: CS03  Working directory: CS03.
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:10.1.1.20, port 1033,
.....................................................................Lines Deleted
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
.....................................................................Lines Deleted
211 *** end of status ***
Command:
```

In Example 17-23, 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).

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 17-24.

Example 17-24 File transfer over data connection in the clear

```
get 'cs03.output' out1
>>> PORT 10,1,1,20,4,70
200 Port request OK.
>>> RETR 'cs03.output'
125 Sending data set CS03.OUTPUT
250 Transfer completed successfully.
31583 bytes transferred in 0.030 seconds.  Transfer rate 1052.77 Kbytes/sec.
```

Tip: If SECURE_DATACONN is defined as PRIVATE or SAFE in FTP server FTP.DATA, it will always ask for securing the data transfer. If SECURE_DATACONN is defined as CLEAR in the FTP client, a user will see following error message when they try to transfer data:

```
425-Server requires protected data connection,
425-Can't open data connection.
```

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 17-25 shows setting the data connection encryption option to private and then transferring the file.

Example 17-25 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 'cs03.output' out2
>>> PORT 10,1,1,20,4,71
200 Port request OK.
>>> RETR 'cs03.output'
125 Sending data set CS03.OUTPUT
250 Transfer completed successfully.
31583 bytes transferred in 0.040 seconds.  Transfer rate 789.57 Kbytes/sec.
```

The status subcommand (**stat**) from the client shows that the data connection is indeed private as shown in Example 17-26.

Example 17-26 STAT shows the Data protection level set to Private

```
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 17-27.

Example 17-27 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:
```

Debug tracing for FTP

When you set up the TLS-enabled FTP session for the first time, you might have the FTP login fail, and you cannot find a reason in FTP server log, syslogd log, or system log. So, you can turn on client logging and tracing at the client side and enable tracing at the server side. In this section, we discuss how to use trace to debug problems.

In our scenario, we had a login issue when we requested the TLS secure log in for the first time as shown in Example 17-28.

Example 17-28 Error message for FTP from one z/OS system to another z/OS system requesting TLS

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" -a TLS 10.1.1.40
EZY2640I Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration paramete
         rs.
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 15:18:26 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA2897I Authentication negotiation failed
EZA1459I NAME (10.1.1.40:CS03):
```

You can turn on the trace in FTP client by editing the FTP client's `FTP.DATA` file as shown in Example 17-29.

Example 17-29 Turn on the DEBUG option in client FTP.DATA

```
TCPIPB.TCPPARMS(FTPCB31)
DEBUG             SEC      1
DEBUG             SOC(2)   2
```

In this example, the numbers correspond to the following information:

1. `DEBUG SEC` shows the processing of the security functions such as TLS and GSSAPI negotiations.
2. `DEBUG SOC(n)` shows details of the processing during the setup of the interface between the FTP application and the network and details of the actual amounts of data that are processed.

You can also use **-d** or **trace** in the **ftp** login command. The resulting trace activity at the client, as displayed in Example 17-30, helps to show the issue.

Example 17-30 Enable tracing in the client

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" -a TLS 10.1.1.40 (trace          1
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 15:20:04 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
GU4945 ftpSetApplData: entered
FC0459 ftpAuth: TLS init failed with rc = 202 (Error detected while opening th
e certificate database)
EZA2897I Authentication negotiation failed
GV0520 seq_stat_file(2): lrecl=0 recfm=0 blksize=0 mode=1
CZ1582 rnetrc:(3) file CS03.NETRC does not exist
EZA1459I NAME (10.1.1.40:CS03):
```

Example 17-30 uses **trace** in the **ftp** command 1 to turn on the FTP client trace. As the trace shows, the key ring name used was incorrect in the FTP client's `FTP.DATA` file, so the certificate database could not be opened. So, authentication negotiation failed 2. After correcting the key ring name, we can log in to the FTP server successfully.

Example 17-31 shows the **trace** for a successful login with the encryption algorithm.

Example 17-31 Debug trace for a successful TLS-enabled FTP login

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" -a TLS 10.1.1.40 (trace

EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 15:24:08 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
GU4945 ftpSetApplData: entered
EZA1701I >>> AUTH TLS
234 Security environment established - ready for negotiation
EZA2895I Authentication negotiation succeeded
GU4945 ftpSetApplData: entered
GV0520 seq_stat_file(2): lrecl=0 recfm=0 blksize=0 mode=1
CZ1582 rnetrc:(3) file CS03.NETRC does not exist
EZA1459I NAME (10.1.1.40:CS03):
```

We can also turn on the trace at the server side using the **modify** command for debugging at the system console (as shown at **a** in Example 17-32).

Example 17-32 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
```

After the data is collected, you turn off the trace using the following steps:

- In the FTP server site, issue the F FTPDD1,DEBUG=NONE command.
- In the FTP client site, comment out the DEBUG statements in the `FTP.DATA` file.

Use a TLS-enabled workstation client to verify server TLS (Draft)

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-project.org/

Figure 17-7 shows our scenario with the separate key rings.

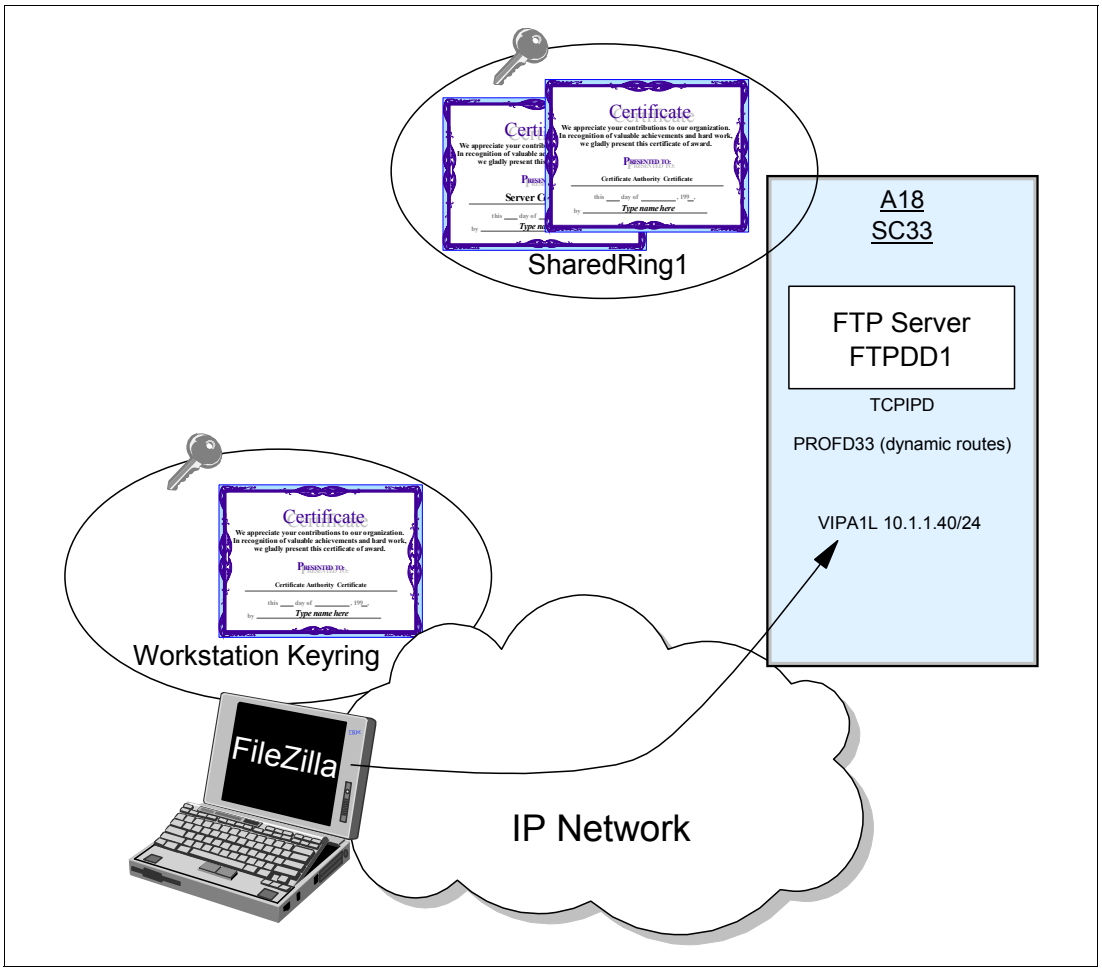


Figure 17-7 Use of a Client key ring separate from the Server key ring

The FileZilla client needs a key ring in which to store the certificate authority certificates and any server certificates it might 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

To create the key ring, complete the following steps:

1. Use FTP to copy 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 data set in “Export the CA certificate to a flat file for prepopulating client devices” on page 729.

   The certificate label is `CS ITSO CA1` and in our examples was stored as `TCPIP.ITSO.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 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 **Close** in the Add Standalone Snap-in window.
10. Click **OK**.
11. In the MMC window, click the plus sign (+) next to Certificates (Local Computer) to show the list of available tasks.
12. Right-click **Trusted Root Certification Authorities** and choose **All Tasks** from the menu. Choose **Import** from the next menu.
13. 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**.
14. Specify that you are placing the certificate “in the following store.”
15. Click **Finish**. You should receive a message that the import was successful.
16. 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 727.

FileZilla configuration for TLS

Figure 17-8 shows the configuration settings needed to connect to the TLS-enabled secure FTP server.

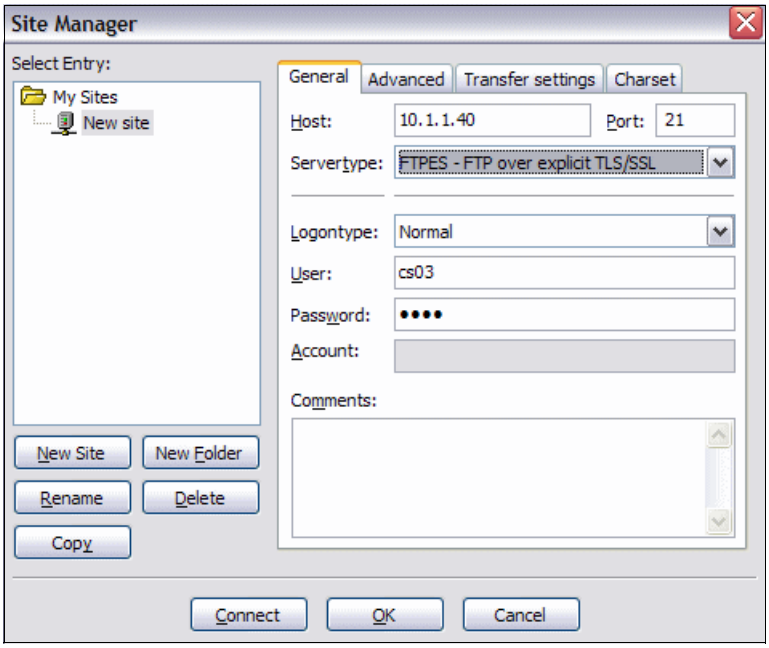


Figure 17-8 Configuration of FileZilla to connect to our TLS-enabled secure FTP server

Observe that in the Servertype menu, we selected **FTPES - FTP over explicit TLS/SSL**. We complete the other fields and click **Connect**. Next, the AUTH flow from the client that is requesting a TLS connection displays as shown in Figure 17-9.

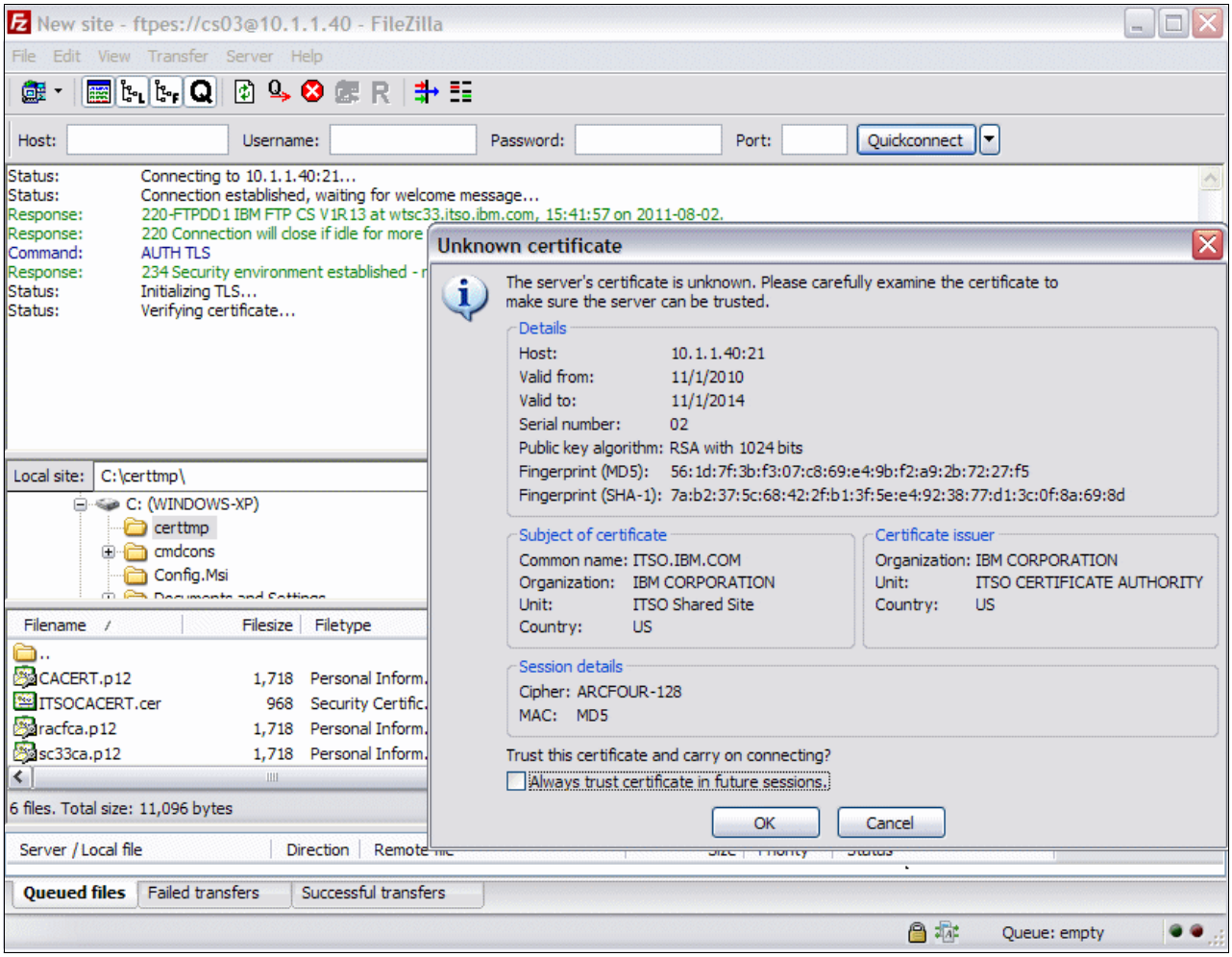


Figure 17-9 Client AUTH TLS flow and certificate confirmation window

The first time FileZilla uses the certificate, it asks for confirmation to accept or reject the unknown certificate that it receives from the server, as shown in Figure 17-9. Click **OK** to accept the certificate.

In our scenario, we had confirmation that FileZilla connected to our secure FTP server. We received messages that indicated that the TLS/SSL connection was established and that the data connection protection was set to private (PROT P) as shown in Figure 17-10.

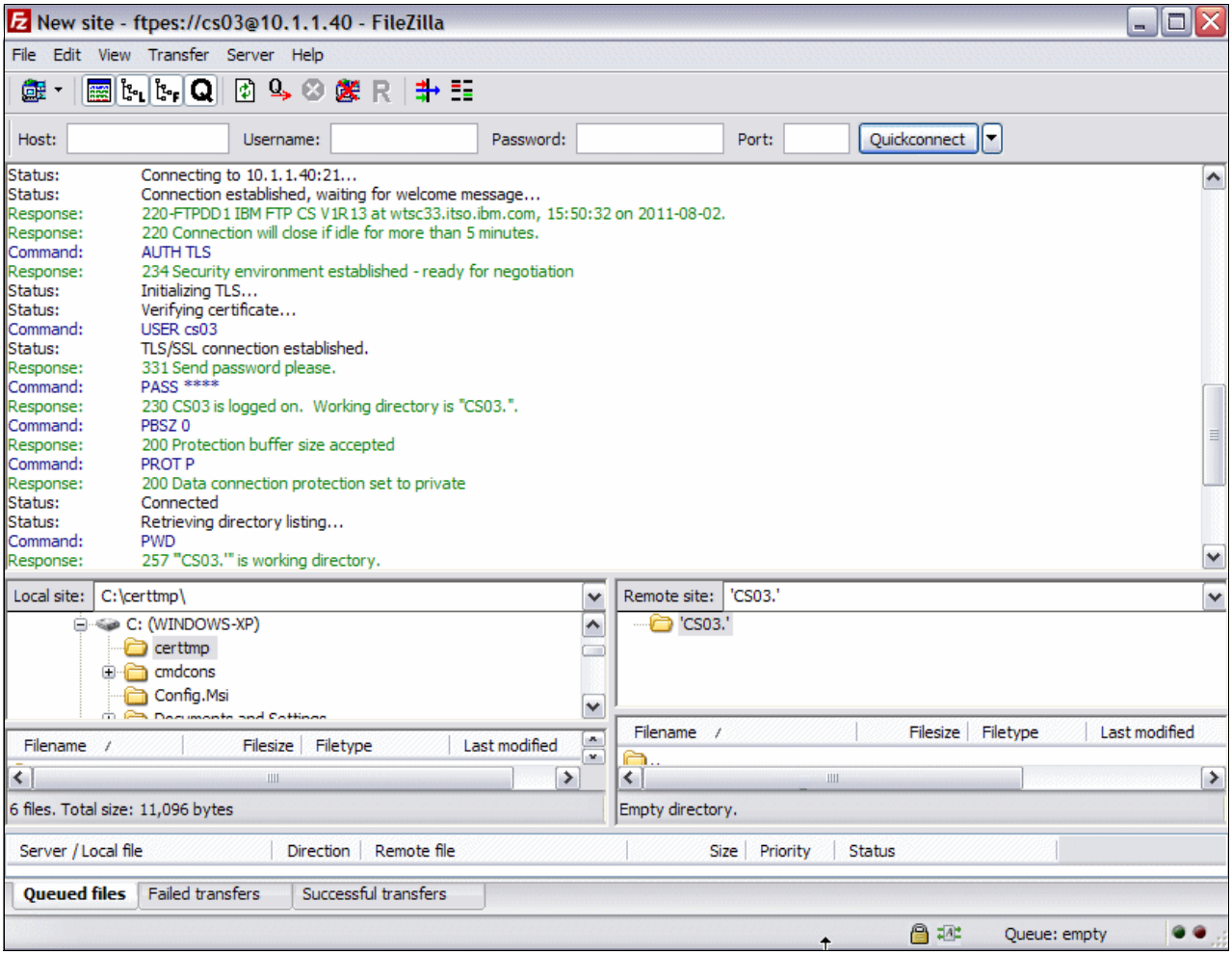


Figure 17-10 Confirmation that FileZilla connected to our system using TLS

17.3.5 Activation, verification of FTP server with TLS security: RFC4217 protocols

Now that we have enabled the Internet draft version of FTP TLS in both a z/OS-to-z/OS environment and in a workstation-to-z/OS environment as described in 17.3.4, “Activation and verification of the FTP server with TLS security: Internet draft protocols” on page 740, we enable z/OS at the RFC 4217 level and test again. The next tasks (described in “Create key ring and certificates and update RACF permissions” on page 726 and “Update the FTP.DATA for the server” on page 731) have been performed already:

- “Check FTP server job log messages” on page 733
- “Verify the non-TLS z/OS client can connect to the server” on page 733
- “Display system job tasks for control and data connections” on page 734

In this section, we explain the following additional tasks:

- Use a TLS-enabled z/OS client to verify server TLS (RFC 4217)
- Use a TLS-enabled workstation client to verify server TLS (RFC 4217)

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 17-33.

Example 17-33 Setting TLSRFCLEVEL

```
EXTENSIONS   AUTH_TLS
TLSMECHANISM FTP
KEYRING      TCPIP/SharedRing1
CIPHERSUITE       SSL_3DES_SHA      ; 0A
CIPHERSUITE       SSL_DES_SHA       ; 09
CIPHERSUITE       SSL_NULL_MD5      ; 01
CIPHERSUITE       SSL_NULL_SHA      ; 02
CIPHERSUITE       SSL_RC4_MD5_EX    ; 03
CIPHERSUITE       SSL_RC4_MD5       ; 04
SECURE_FTP   ALLOWED
SECURE_CTRLCONN   PRIVATE
SECURE_DATACONN CLEAR
TLSTIMEOUT        100
a TLSRFCLEVEL       RFC4217
```

In this example, TLSRFCLEVEL (a) in TCPIPD.TCPPARMS(FTPSECD) is changed from DRAFT to RFC4217.

Next, Example 17-34 shows the change of the TLSRFCLEVEL for the z/OS client on TCPIPB in TCPIPB.TCPPARMS(FTPCB31) a to reflect the RFC 4217 level.

Example 17-34 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 **ftp** 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 use the correct client FTP.DATA file:

```
ftp 10.1.1.40 -a TLS -f "//'TCPIPB.TCPPARMS(FTPCB31)'" (TCP TCPIPB
```

Example 17-35 shows that the connection succeeds.

Example 17-35 TCPIPB FTP client view of the connection at the RFC 4217 level

```
ftp 10.1.1.40 -a TLS -f "//'TCPIPB.TCPPARMS(FTPCB31)'" (TCP TCPIPB
EZY2640I Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration parameters.
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 16:28:36 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1701I >>> AUTH TLS
234 Security environment established - ready for negotiation
EZA2895I Authentication negotiation succeeded
EZA1459I NAME (10.1.1.40:CS03):
EZA1701I >>> USER CS03
331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS
230 CS03 is logged on.  Working directory is "CS03.".
EZA1460I Command:
stat
EZA1701I >>> STAT
211-Server FTP talking to host ::ffff:10.1.1.20, port 1042
.....................................................................Lines Deleted
211-Authentication type: TLS
211-Control protection level: Private
211-Data protection level: Clear
211-TLS security is supported at the RFC4217 level
.....................................................................Lines Deleted
211 *** end of status ***
```

The server is operating under the protocols of RFC 4217 (a). If you miss the server's 211 response about the TLS support level, you can query for the level with the command **stat (tlsrfclevel** as in Figure 17-11.

```
 EZA1460I Command:
stat (tlsrfclevel
 EZA1701I >>> XSTA (tlsrfclevel
 211-TLS security is supported at the RFC4217 level a
 211 *** end of status ***
```

Figure 17-11 Querying server TLSRFCLevel support

Likewise, you can investigate the RFC level at the client with the **locstat tlsrfclevel** command as seen in Figure 17-12.

```
 EZA1460I Command:
locstat tlsrfclevel
 EZA2916I local site variable TLSRFClevel is set to RFC4217 a
```

Figure 17-12 Querying client 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 17-13 shows how to set the local client TLSRFCLEVEL back to DRAFT mode.

```
 EZA1460I Command:
locsite tlsrfclevel=draft
 EZA1460I Command:
locstat tlsrfclevel
 EZA2916I local site variable TLSRFClevel is set to DRAFT
 EZA1460I Command:
```

Figure 17-13 Using LOCSITE client command to set the TLSRFCLEVEL

The **status** command output in Figure 17-14 shows that the server is also operating under RFC 4217. 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
211-Server site variable READVB is set to LE
211-Port of Entry resource class for IPv4 clients is: TERMINAL
211-Record format FB, Lrecl: 128, Blocksize: 6144
211-Server site variable EATTR is set to SYSTEM
211-Server site variable DSNTYPE is set to SYSTEM
211-Server site variable LISTSUBDIR is set to TRUE
211-Server site variable LISTLEVEL is set to 0
211 *** end of status ***
EZA1460I Command:
```

Figure 17-14 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 17-15 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. As shown in Figure 17-15, we connect to the server with the **-a TLS** specification as before and use **ccc** command (1) to set control connection protection as clear (2).

```
 EZA1460I Command:
ccc
 EZA1701I >>> CCC                    1
 200 CCC command successful
 EZA2905I Control connection protection is clear
 EZA1460I Command:
status
 EZA1701I >>> STAT
 211-Server FTP talking to host ::ffff:10.1.1.20, port 1042
 ...............................................................Lines Deleted
 211-Authentication type: TLS
 211-Control protection level: Clear 2
 211-Data protection level: Clear
 211-TLS security is supported at the RFC4217 level
 ...............................................................Lines Deleted
 211 *** end of status ***
 EZA1460I Command:
```

Figure 17-15 Clearing the control connection: ccc

After you clear the control connection, you cannot alter the state of the data connection. Therefore, if you want a protected data connection and a clear control connection, the data connection state must be protected *prior* to issuing the **ccc** command.

Next, as shown in Figure 17-16, we reset the control connection to private with the **auth tls** command (1). When we attempt the **ccc** command again (2), we are told that we must enter user ID and password again (3). 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.

```
 EZA1460I Command:
auth tls                                                    1
 EZA1701I >>> AUTH TLS
 234 Security environment established - ready for negotiation
 EZA2895I Authentication negotiation succeeded
 EZA1460I Command:
status
 EZA1701I >>> STAT
.............................................................Lines Deleted
 211-Authentication type: TLS
 211-Control protection level: Private
 211-Data protection level: Clear
 211-TLS security is supported at the RFC4217 level
.............................................................Lines Deleted
 211 *** end of status ***
EZA1460I Command:
ccc                                                         2
 EZA1701I >>> CCC
 530 You must first login with USER and PASS.               3
 EZA2904I Cannot set protection level to clear
 EZA2905I Control connection protection is private
 EZA1460I Command:
user cs03
 EZA1701I >>> USER cs03
 331 Send password please.
 EZA1789I PASSWORD:
 EZA1701I >>> PASS
 230 CS03 is logged on.  Working directory is "CS03.".
 EZA1460I Command:
ccc
 EZA1701I >>> CCC
 200 CCC command successful
 EZA2905I Control connection protection is clear
 EZA1460I Command:
```

Figure 17-16 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 use FTP to connect to the server with the following command:

```
ftp 10.1.1.40 -f "//'TCPIPB.TCPPARMS(FTPCB31)'"  (TCP TCPIPB
```

```
.....................................................................Lines Deleted
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 21.
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 16:44:41 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.40:CS03):
EZA1701I >>> USER CS03                                          a
331 Send password please.
EZA1701I >>> PASS
230 CS03 is logged on.  Working directory is "CS03.".
EZA1460I Command:
stat (tlsrfclevel
 EZA1701I >>> XSTA (tlsrfclevel
 211-TLS security is supported at the RFC4217 level             b
 211 *** end of status ***
 EZA1460I Command:
status
 EZA1701I >>> STAT
 211-Server FTP talking to host ::ffff:10.1.1.20, port 1135
 211-User: CS03  Working directory: CS03.
 211-The control connection has transferred 209 bytes
.....................................................................Lines Deleted
 211-Authentication type: None                                  c
 211-TLS security is supported at the RFC4217 level
.....................................................................Lines Deleted
```

Figure 17-17 An FTP connection without protection

In Figure 17-17 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 17-18, 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.

```
 EZA1460I Command:
auth tls
 EZA1701I >>> AUTH TLS
 234 Security environment established - ready for negotiation
 EZA2897I Authentication negotiation failed
 EZA1460I Command:
```

Figure 17-18 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 749. 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 747.

We connected the FTP client to the server at TCPIPD successfully. Click **Server** → **Enter custom command**. We entered **`stat`** to confirm the use of the TLS protocols as shown in Figure 17-19.

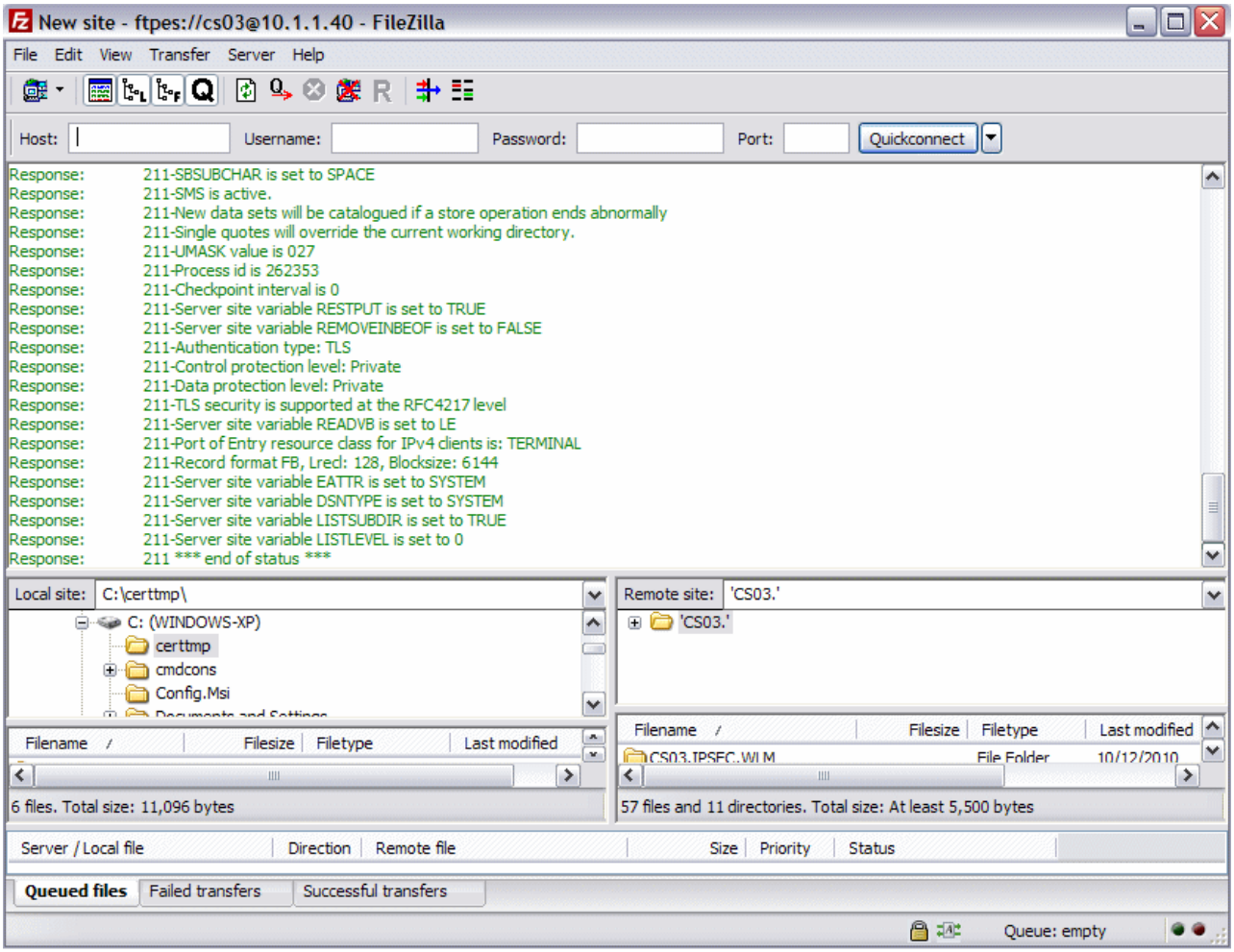


Figure 17-19 FileZilla stat command output

We then used the Server menu to enter the command **`prot c`** to obtain a clear data connection. The results are shown in Figure 17-20.

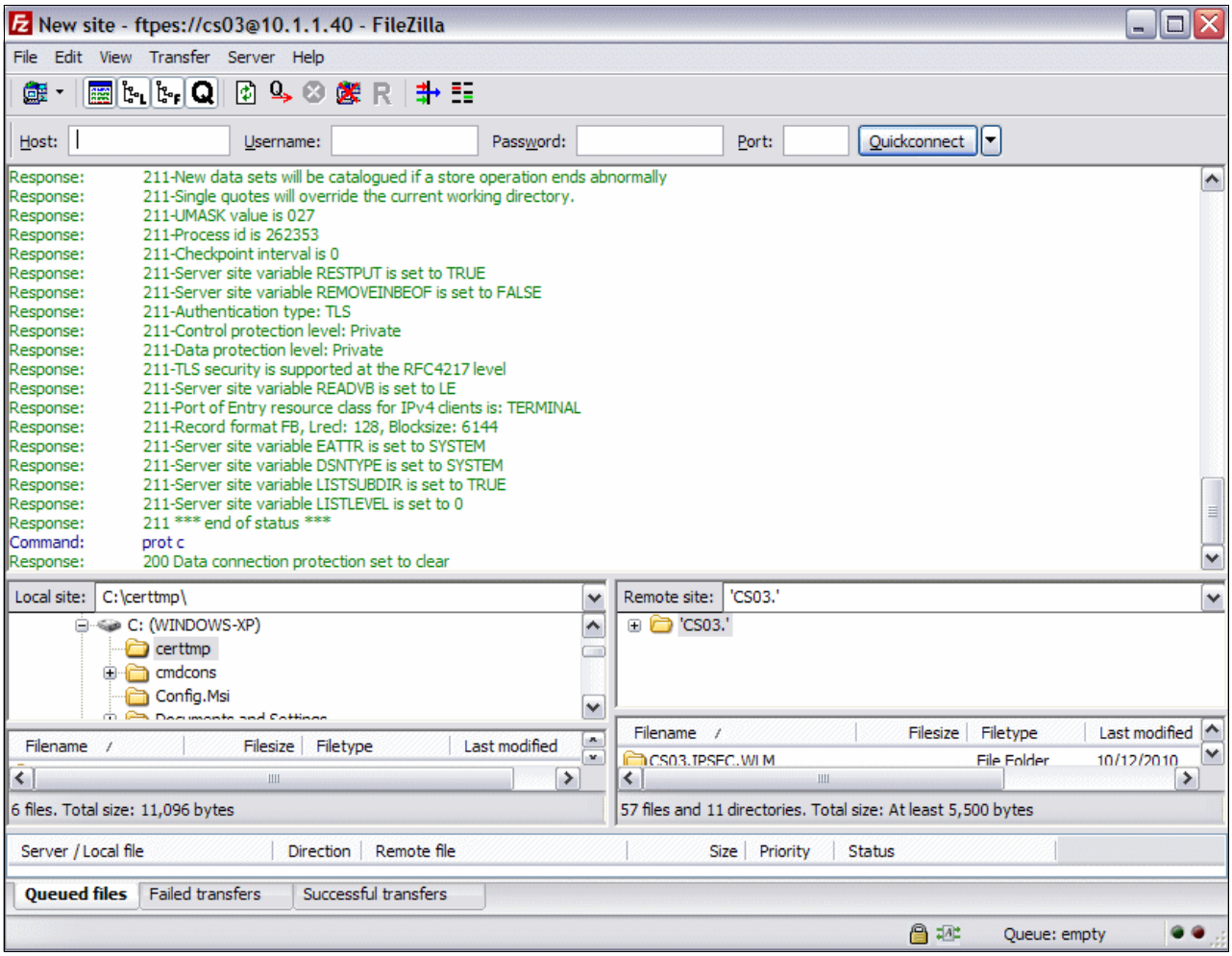


Figure 17-20 Results of client entering port c to clear the data connection

Although FileZilla sends the **ccc** command, it cannot support a clear control connection. Therefore the connection is terminated. Figure 17-21 shows the connection failure.

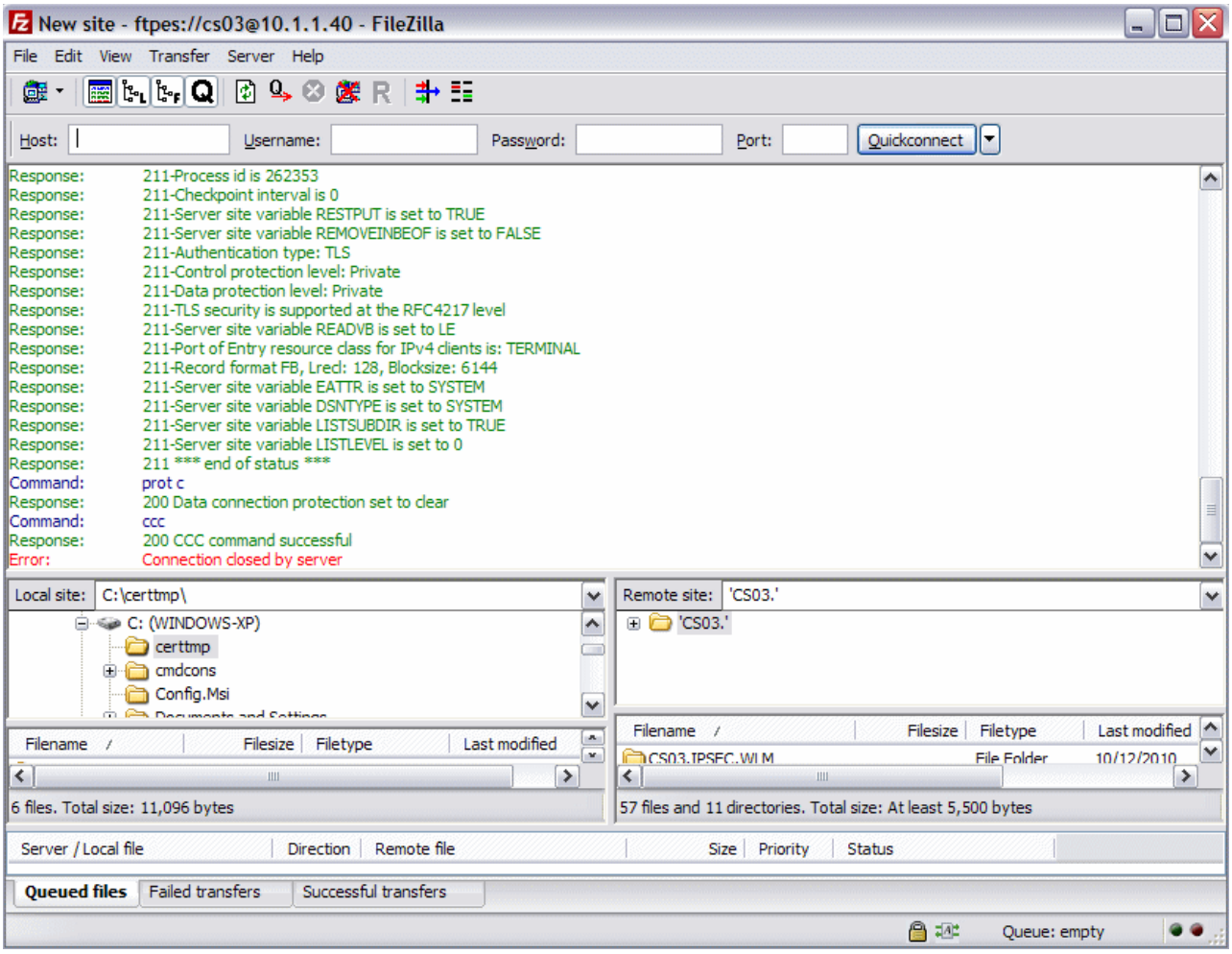


Figure 17-21 Entry of ccc command by the FileZilla client

Suggestion: 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 other unpredictable occurrence.

Although a z/OS server that is running with RFC 4217 can still communicate with downlevel clients using standard TLS protocols, 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.

17.3.6 Implicit secure TLS login

z/OS FTP supports TLS security for FTP sessions based on Revision 5 of *On Securing FTP with TLS*. This document describes the implicit method for securing an FTP session with TLS. Connections to port 990 are assumed to be secure, and no AUTH command is needed.

The requirement to implicitly secure the session with FTP port 990 was removed from later drafts and from RFC4217. Nonetheless, FTP platforms use secure FTP sessions with TLS,

so certain FTP platforms implement implicit TLS security. In this section, we discuss implicitly secure TLS FTP sessions.

Note: Starting the FTP server on port 990 is not recommended because not all secure FTP clients will be able to connect to it. Instead, the FTP server can provide equivalent support on a different port by specifying the following statements in the `FTP.DATA` file:

```
EXTENSIONS AUTH_TLS
SECURE_FTP REQUIRED
SECURE_CTRLCONN PRIVATE
SECURE_DATACONN PRIVATE
```

See APAR II13516 for more information.

Security handshake sequence

The current z/OS FTP server supports implicit TLS logins, both for z/OS FTP clients and non-z/OS client. The z/OS FTP server can drive the security handshake before or after sending the first 220 reply to client. This is controlled by the `SECUREIMPLICITZOS` operand in the FTP server `FTP.DATA` file.

When `SECUREIMPLICITZOS` is set to `TRUE`, the z/OS FTP server default, the server expects the security handshake *after* it sends reply 220 to the client. Figure 17-22 shows the z/OS FTP client to z/OS FTP server implicit login process.

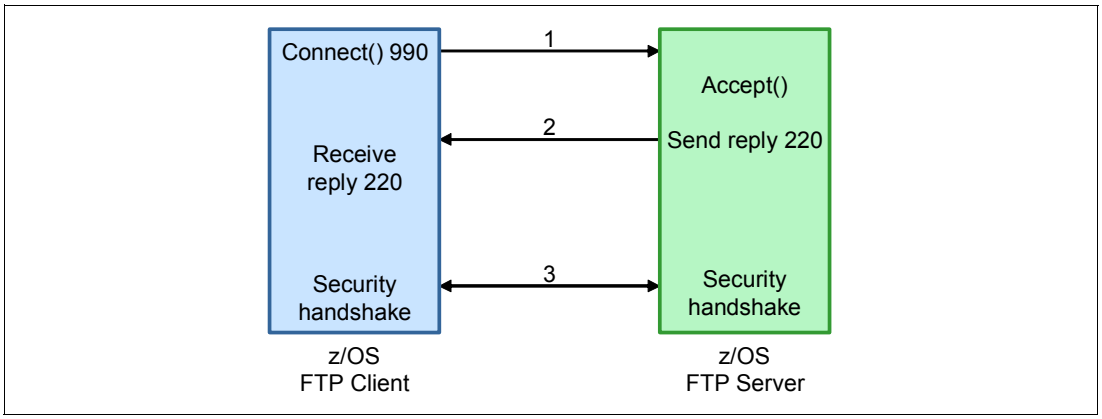


Figure 17-22 z/OS FTP client implicit login with SECUREIMPLICITZOS TRUE

The process is as follows:

1. The client connects to the TLSPORT. This example uses the default TLSPORT 990.
2. The z/OS FTP daemon accepts the incoming connection and sends the first reply to the client. Then the client receives this reply.
3. The FTP client and server perform the security handshake. When the security handshake is complete, the connection is secured.

The details of the TLS handshake are handled by the System SSL element of z/OS. The handshake is a sequence of records that are exchanged between the client and server to negotiate exactly how the connection is encrypted from among the different TLS encoding schemes that are available to the client and server.

When SECUREIMPLICITZOS is set to FALSE, the z/OS FTP server uses the same method as other non-z/OS servers and expects the security handshake before it sends reply 220 to the non-z/OS client. Figure 17-23 shows how non-z/OS FTP clients implicitly log in to the z/OS FTP server.

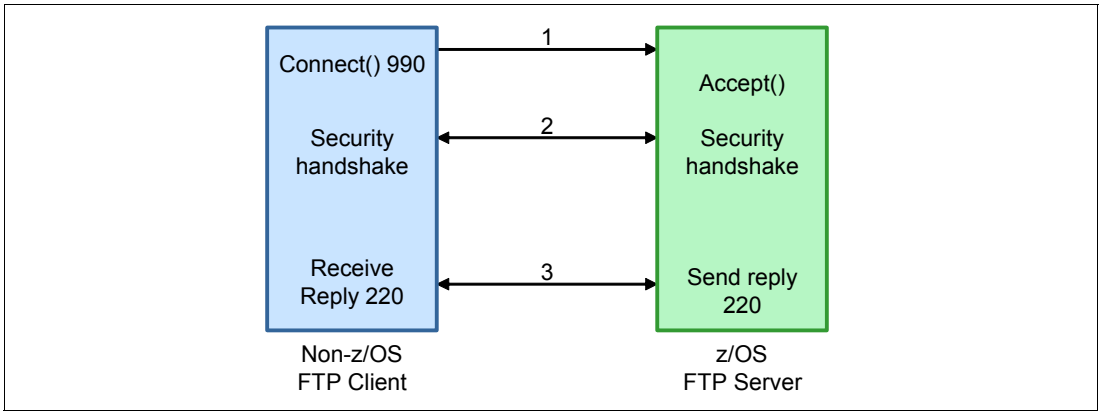


Figure 17-23 Non-z/OS FTP client implicit login with SECUREIMPLICITZOS FALSE

This process is as follows:

1. The client connects to the TLSPORT. This example uses the default TLSPORT 990.
2. The FTP server accepts the incoming connection, and then FTP client and server drive the security handshake,
3. After the security handshake is complete, the FTP server sends reply 220 to the client. The client receives this reply, and the session is established.

Configure z/OS FTP server for implicit TLS login

Next, we enable the z/OS FTP server to listen to port 990 and set up related parameters for an implicit TLS login. See *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997 for detailed FTP server implementation steps.

We use TCPIPD in system SC33 to demonstrate the configuration and verification for the scenario. The first step is to add PORT 990 to the TCPIP stack profile as shown in Example 17-36.

Example 17-36 PORT statement in TCPIPD.TCPPARMS(PROFD33)

```
PORT
   989 TCP *  NOAUTOLOG        ; FTP data port
   990 TCP FTPDD1              ; FTP control port
```

The /etc/services definition needs to be changed for the FTP server as shown in Example 17-37.

Example 17-37 /etc/services definition

```
ftp             990/tcp
```

Then, FTP.DATA in the FTP server needs to be updated as shown in Example 17-38.

Example 17-38 Server FTP.DATA to define implicit TLS login

```
; ------------------------------------------------------------------
; 7. Security options
; ------------------------------------------------------------------
;EXTENSIONS AUTH_TLS                                    1
TLSMECHANISM FTP                                        2
 KEYRING TCPIP/SharedRing1                              3
;SECURE_LOGIN      NO_CLIENT_AUTH
;SECURE_PASSWORD   REQUIRED
;SECURE_PASSWORD_KERBEROS  REQUIRED
;SECURE_FTP ALLOWED                                     1
;SECURE_CTRLCONN PRIVATE                                1
;SECURE_DATACONN CLEAR                                  1
;SECURE_PBSZ       16384
 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
;CIPHERSUITE       SSL_RC4_SHA       ; 05
;CIPHERSUITE       SSL_RC2_MD5_EX    ; 06
;CIPHERSUITE       SSL_AES_128_SHA   ; 2F
;CIPHERSUITE       SSL_AES_256_SHA   ; 35
 TLSTIMEOUT 100
 TLSRFCLEVEL RFC4217                                    4
 TLSPORT           990                                  5
 SECUREIMPLICITZOS FALSE                                6
```

In this example, the numbers correspond to the following information:

1. All parameters that are used by explicit TLS login are commented out.
2. Indicates native FTP TLS support (not AT-TLS).
3. Adds a KEYRING statement to identify the owner and name of the key ring file.
4. Adds `TLSRECLEVEL` for RFC 4217.
5. Takes the default of 990 on TLSPORT to set the secure port on which the FTP server implicitly protects the FTP session with TLS. You can select a different secure port for implicit secure FTP sessions using this statement.
6. Adds `SECUREIMPLICITZOS FALSE` to specify that server drives the security handshake before sending the first reply. This supports implicitly secured logins for non-z/OS clients and z/OS FTP client which have the same `SECUREIMPLICITZOS FALSE` configuration.

Tip: You must specify the same TLSPORT in FTP client and server. Problems with implicit TLS logins to the z/OS FTP server are usually because of a failure to synchronize the client and server configured values for TLSPORT and SECUREIMPLICITZOS.

In the client's FTP.DATA, the same parameters as the FTP server should be specified as shown in Example 17-39.

Example 17-39 Client FTP.DATA to define implicit TLS login

```
; ---------------------------------------------------------------------
; 7. Security options
; ---------------------------------------------------------------------
 TLSRFCLEVEL       RFC4217
 KEYRING           TCPIP/SharedRing1
 SECUREIMPLICITZOS FALSE
 TLSPORT           990
```

Implicit TLS login verification scenario

Now, we explain how to test the implicit TLS login using two scenarios:

- z/OS FTP client implicit TLS login
- Non-z/OS FTP client implicit TLS login

z/OS FTP client implicit TLS login

Based on our configuration, we used port 990 to implicitly log in to the z/OS FTP server as shown in Figure 17-24.

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" 10.1.1.40 990        1
.....................................................................Lines Deleted
EZY2640I Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration paramete
         rs.
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 990.
EZA2895I Authentication negotiation succeeded                        2
EZA2919I Session starts with protection on the data connection       2
220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 17:43:36 on 2011-08-02.
220 Connection will close if idle for more than 5 minutes.
EZA1459I NAME (10.1.1.40:CSO3):
EZA1701I >>> USER CSO3
331 Send password please.
EZA1789I PASSWORD:
EZA1701I >>> PASS
230 CSO3 is logged on.  Working directory is "CSO3.".
Command:
locstat tlsrfclevel                                                  3
local site variable TLSRFClevel is set to RFC4217
Command:
stat                                                                 4
.....................................................................Lines Deleted
211-Authentication type: TLS
211-Control protection level: Private
211-Data protection level: Private
211-TLS security is supported at the RFC4217 level
.....................................................................Lines Deleted
```

Figure 17-24 z/OS client implicit TLS login scenario

In this figure, the numbers correspond to the following information:

1. The command that we used to log in to the z/OS FTP server. Note that we specify port 990 and leave out the **-a TLS** parameter to request the implicit TLS login.
2. Authentication negotiation is accomplished automatically, and the session is secure.
3. The **locstat tlsrfclevel** command shows that the TLS RFC level is set to RFC4217 as we defined in the configuration.
4. The **stat** command shows all the secure options for this connection. We used TLS authentication and both control and data connections with the private level.

The **netstat** command output shows that the FTP daemon is listening on port 990 (1) as shown in Figure 17-25.

```
D TCPIP,TCPIPD,N,CONN,PORT=990
EZD0101I NETSTAT CS V1R13 TCPIPD 650
USER ID  CONN     STATE
FTPDD1   00000973 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..990
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1053
FTPDD1   0000097D LISTEN
  LOCAL SOCKET:   ::..990                          1
  FOREIGN SOCKET: ::..0
2 OF 2 RECORDS DISPLAYED
```

Figure 17-25 The netstat command output for implicit TLS connection

In addition, the **netstat** command also shows the connections when data transfers. By default, z/OS FTP server uses port 989 for data connection 2, as shown in Figure 17-26.

```
D TCPIP,TCPIPD,N,CONN
EZD0101I NETSTAT CS V1R13 TCPIPD 678
USER ID  CONN     STATE
FTPDDD3 000009B3 FINWT2
  LOCAL SOCKET:   ::FFFF:10.1.1.40..989    2
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1055
FTPDD1   0000099A LISTEN
  LOCAL SOCKET:   ::..990
  FOREIGN SOCKET: ::..0
FTPDD1   000009AB ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.1.40..990
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1053
```

Figure 17-26 netstat command output for data connection with implicit TLS connection

Now, we change the SECUREIMPLICITZOS statement to TRUE in the FTP client FTP.DATA file, and re-test as shown in Figure 17-27.

```
ftp -p tcpipb -f "//'TCPIPB.TCPPARMS(FTPCB31)'" 10.1.1.40 990
EZY2640I Using 'TCPIPB.TCPPARMS(FTPCB31)' for local site configuration paramete
         rs.
EZA1450I IBM FTP CS V1R13
EZA1466I FTP: using TCPIPB
EZA1554I Connecting to:   10.1.1.40 port: 990.
....
EZA2589E Connection to server interrupted or timed out. Waiting for reply
EZA1721W Server not responding, closing connection.
EZA1460I Command:
```

Figure 17-27 z/OS client implicit TLS login with mismatching SECUREIMPLICITZOS

In this case the FTP connection times out. When using SECUREIMPLICITZOS FALSE in the z/OS FTP server, FTP server expects the security handshake before it sends the first reply to the client. However, in the client site, the z/OS FTP client expects the security handshake to occur after it receives the first reply from the server, based on the SECUREIMPLICITZOS TRUE parameter. So, in our scenario, the connection timed out because of the operand mismatch in the FTP client and server.

Non-z/OS FTP client implicit TLS login

To support non-z/OS FTP client implicit TLS login to z/OS FTP server, we need to define SECUREIMPLICITZOS FALSE in the FTP server FTP.DATA file. With this parameter, the z/OS FTP server expects the security handshake before it sends the first reply to the client, which is the same method that non-z/OS clients use for security handshake.

We continued to use FileZilla for the testing, and configure FileZilla for implicit TLS support. We tested with the same certificate authority certificate and did not need to use MMC again to establish the key ring that FileZilla needs.

Figure 17-28 shows the FileZilla configuration for implicit TLS. Observe that in the Servertype menu, we selected **FTPS - FTP over implicit TLS/SSL**. Because we used the standard implicit port (990), we did not need to specify the port number. You need to specify the port number if you are not using the default 990 port for implicit login.

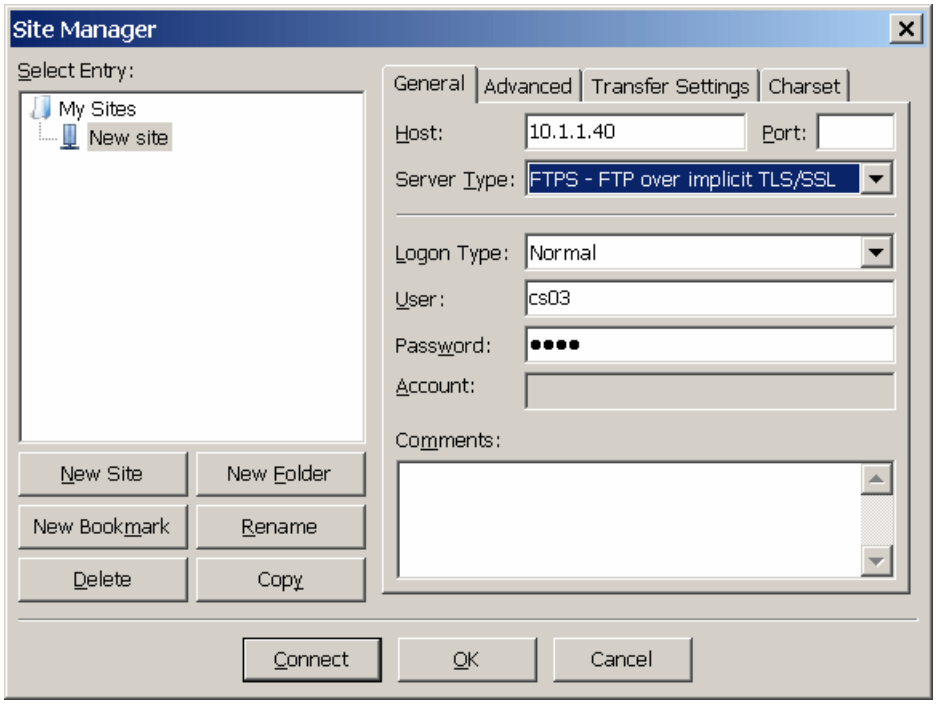


Figure 17-28 FileZilla configuration for implicit TLS

Click **Connect**. The connecting messages shown in Figure 17-29 display. FileZilla uses the default port 990 to establish the implicit TLS connection as shown at a. The connection is established with TLS security (b), and the data connection protection is set to private (c).

```
Status:.Connecting to 10.1.1.40:990...                                    a
Status:.Connection established, initializing TLS...                       b
Status:.Verifying certificate...
Status:.TLS/SSL connection established, waiting for welcome message...
Response:.220-FTPDD1 IBM FTP CS V1R13 at wtsc33.itso.ibm.com, 18:03:34 on
2011-08-02.
Response:.220 Connection will close if idle for more than 5 minutes.
Command:.USER cs03
Response:.331 Send password please.
Command:.PASS ****
Response:.230 CS03 is logged on.  Working directory is "CS03.".
Command:.SYST
Response:.215 MVS is the operating system of this server. FTP Server is running
Command:.FEAT
Response:.211- Extensions supported
Response:. AUTH TLS
Response:. PBSZ
Response:. PROT
Response:.211 End
Command:.PBSZ 0
Response:.200 Protection buffer size accepted
Command:.PROT P
Response:.200 Data connection protection set to private                   c
Status:.Connected
```

Figure 17-29 FileZilla implicit TLS login to z/OS FTP server

Now, we change SECUREIMPLICITZOS from FALSE to TRUE. After restarting the z/OS FTP server, we tried to use FileZilla for an implicit TLS login to the server and received the message shown in Figure 17-30. We received the error messages because the FTP server and client did not use the same convention for securing the connection implicitly. The client and server deadlocked and, eventually, the connection timed out.

```
Status:Connecting to 10.1.1.40:990...
Status:Connection established, initializing TLS...
Error:Connection timed out
Error:Could not connect to server
Status:Waiting to retry...
```

Figure 17-30 FileZilla implicit TLS login error message

Tips

Consider the following tips for using the implicit TLS login method:

- If the client and server are on z/OS FTP, use the TLSPORT statement in FTP.DATA to configure the same TLSPORT for the client and server.
- If the client and server are on z/OS FTP, use the SECUREIMPLICITZOS statement in FTP.DATA to set the same value in the client and server.

- You can obtain the most flexibility at the server by configuring `SECUREIMPLICITZOS` as `FALSE`, because it supports both non-z/OS FTP and z/OS FTP clients that have the same SECUREIMPLICITZOS configured.
- You might need to configure `SECUREIMPLICITZOS` as `TRUE` in the FTP server for compatibility with other z/OS images prior to V1R12.

17.4 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 see it for additional detail or for clarification of the individual implementation steps.

17.4.1 Description of FTP AT-TLS support

Tip: For information about AT-TLS in general, consult Chapter 12, “Application Transparent Transport Layer Security” on page 551.

The FTP client and server can be configured to use AT-TLS to support TLS connections.

Three types of AT-TLS applications exist:

- 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 17.3, “FTP with native TLS security support” on page 724 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, because 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 might be necessary.

With AT-TLS, SSLv2 is possible, but it is disabled by default in the AT-TLS policy.

Suggestion: 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. AT-TLS has considerable advantages over standard TLS/SSL for the FTP environment. AT-TLS can exploit the following items:

- 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 key ring.
- 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 5, “Central Policy Server” on page 161.

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 key ring, 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.

17.4.2 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 *IBM z/OS Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997.

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:

- z/OS client to z/OS server

  We verified that the standard z/OS TLS client requesting TLS worked as expected when AT-TLS was enabled at the server. We also verified that a z/OS AT-TLS client at TCPIPB could establish connections to the AT-TLS FTP server at TCPIPD.

- Workstation client to z/OS server

  We verified that the workstation client invoking standard TLS could operate with the FTP server configured for AT-TLS on the TCPIPD stack.

Complete the following tasks to set up FTP server TLS support:

1. Create key ring and certificates for AT-TLS.
2. Update the FTP.DATA for the server for AT-TLS.
3. Create the policy agent procedure and build the FTP AT-TLS policy.
4. Update the TCP/IP stack for AT-TLS.
5. TCP/IP stack changes.
6. Optional: Create certificates for client authentication.
7. 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 726, 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).

See Example 17-40 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 17-40 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 17-40 on page 771 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 17-40 on page 771:

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 that identifies the location of our key ring file is commented out. The key ring 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 suggest 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 might 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 can 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. See “Considerations” on page 725 and 17.3.5, “Activation, verification of FTP server with TLS security: RFC4217 protocols” on page 751 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 IBM Configuration Assistant to create an FTP server AT-TLS policy

Policy agent procedure

There are many ways to code the procedure explained in the sample file in `hlq.SEZAINST(PAGENT)`. See Chapter 4, “Policy agent” on page 103 for instructions on setting up the PAGENT procedure for policy agent. We structured our PAGENT procedure as you see in Example 17-41 on page 773, ensuring that the STDENV data set for the Language Environment was allocated with a Variable format, as is required by Language Environment.

Example 17-41 Sample PAGENT procedure

```
//PAGENT   PROC
//PAGENT   EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/
//             -l SYSLOGD'
//STDENV   DD PATH='/etc/pagent.sc&SYSCLONE..env',PATHOPTS=(ORDONLY)
//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 data set with Record Format V, and which is referenced in the procedure, is depicted in Example 17-42.

Example 17-42 Policy agent environment variable file

```
LIBPATH=/usr/lib:.
PAGENT_CONFIG_FILE=/SC33/etc/pagent.sc33.conf
PAGENT_LOG_FILE=SYSLOGD
TZ=EST5EDT
```

Tip: The specification of the configuration file in the JCL overrides the PAGENT_CONFIG_FILE definition in the Language Environment file.

Policy agent configuration files

Example 17-43 shows the main configuration file that we built in the zFS.

Example 17-43 Main PAGENT configuration file

```
#  ******************************************************************
#  /etc/pagent.sc33.conf in SC33
#  ******************************************************************
  LogLevel 15
  TcpImage TCPIPD /etc/pagent.sc33.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 17-44.

Example 17-44 Image configuration file for stack TCPIPD

```
# This file /etc/pagent.sc33.tcpipd.conf contains                  #
# image statements for TCP/IP stack tcpipd in z/OS image sc33
####################################################################
# A policy for FTPDD1 for AT-TLS support                           #
####################################################################
TTLSConfig /etc/cfgasst/v1r13/SC33/TCPIPD/tlsPol_FTP_Server
```

Our TCP/IP AT-TLS policy file is the one we are going to create using the IBM Configuration Assistant in the next steps. See “Using the IBM Configuration Assistant to create an FTP server AT-TLS policy” on page 776 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/pagent.sc33.tcpipd.ttls.ftp.policy).

AT-TLS policy for z/OS FTP server and z/OS FTP client

The following points are the most important 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:

- 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.

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 shown in the excerpt from pagent_TTLS.conf depicted in Example 17-45 on page 775.

Example 17-45 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 17-45, 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, and 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 might have introduced into the policy.

Using the IBM Configuration Assistant to create an FTP server AT-TLS policy

For details about using z/OSMF Configuration Assistant, see 4.4, “Configuration Assistant for z/OS Communications Server” on page 127.

To create an FTP server AT-TLS policy, complete the following steps:

1. In the Main Perspective panel (Figure 17-31), click the small symbol to the right of Action and select **Open** → **Create a New Backing Store** and type a file name for the backing store file. We named ours `FTP_TTLS`.

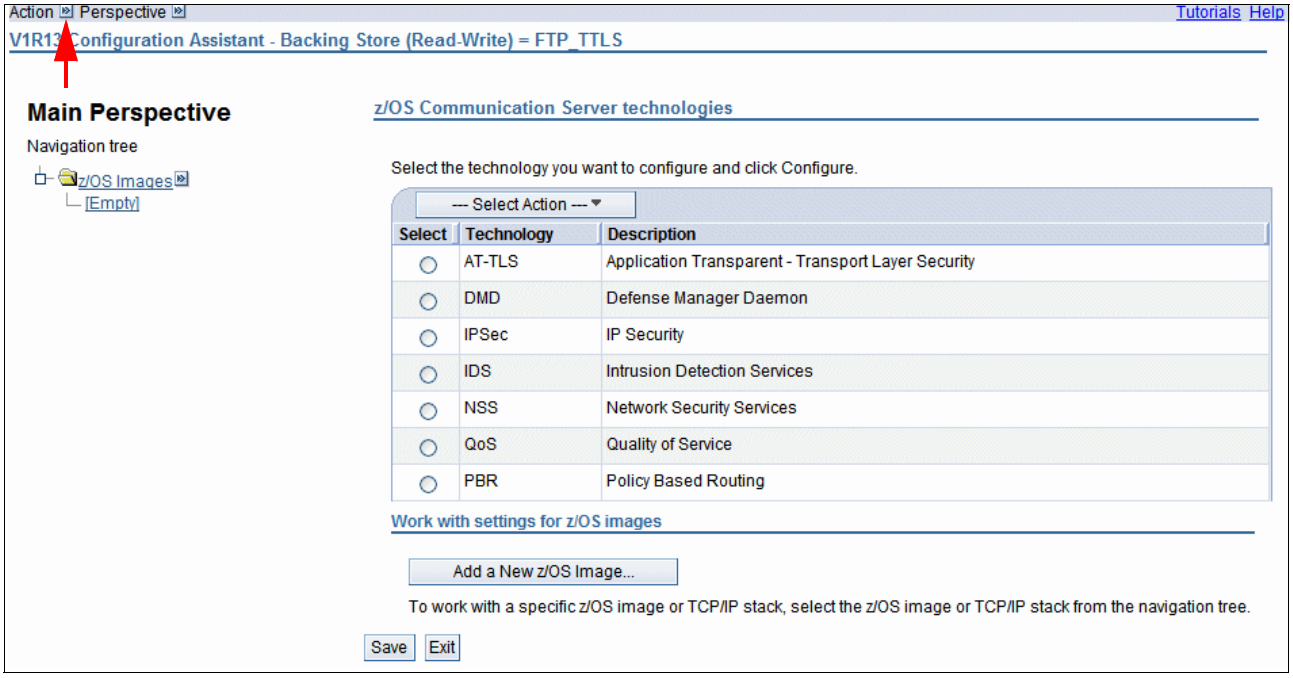


Figure 17-31 Main Perspective panel

2. Next, create the definition for the z/OS system. Click **Add a new z/OS Image**. Enter the z/OS image information and click **OK** as shown in Figure 17-32.

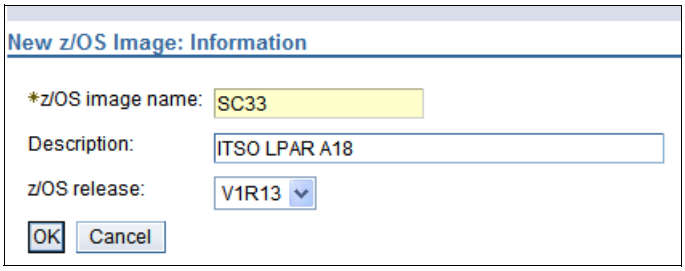


Figure 17-32 New z/OS image information

3. In the Proceed to the Next Step message panel, click **Yes**. Then, in the New TCP/IP Stack: Information panel, type the TCP/IP stack information as shown in Figure 17-33, and click **OK**.

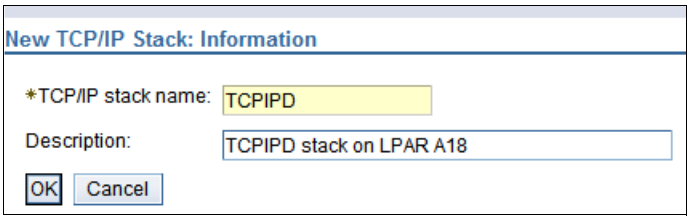


Figure 17-33 Providing the name and description of the stack

4. The Main Perspective panel (Figure 17-34) shows the z/OS image and TCPIP stack in the navigation tree. To the right, it shows the TCP/IP stack information and technologies that can be configured. Select the technology named **AT-TLS** and click **Select Action** → **Enable**.

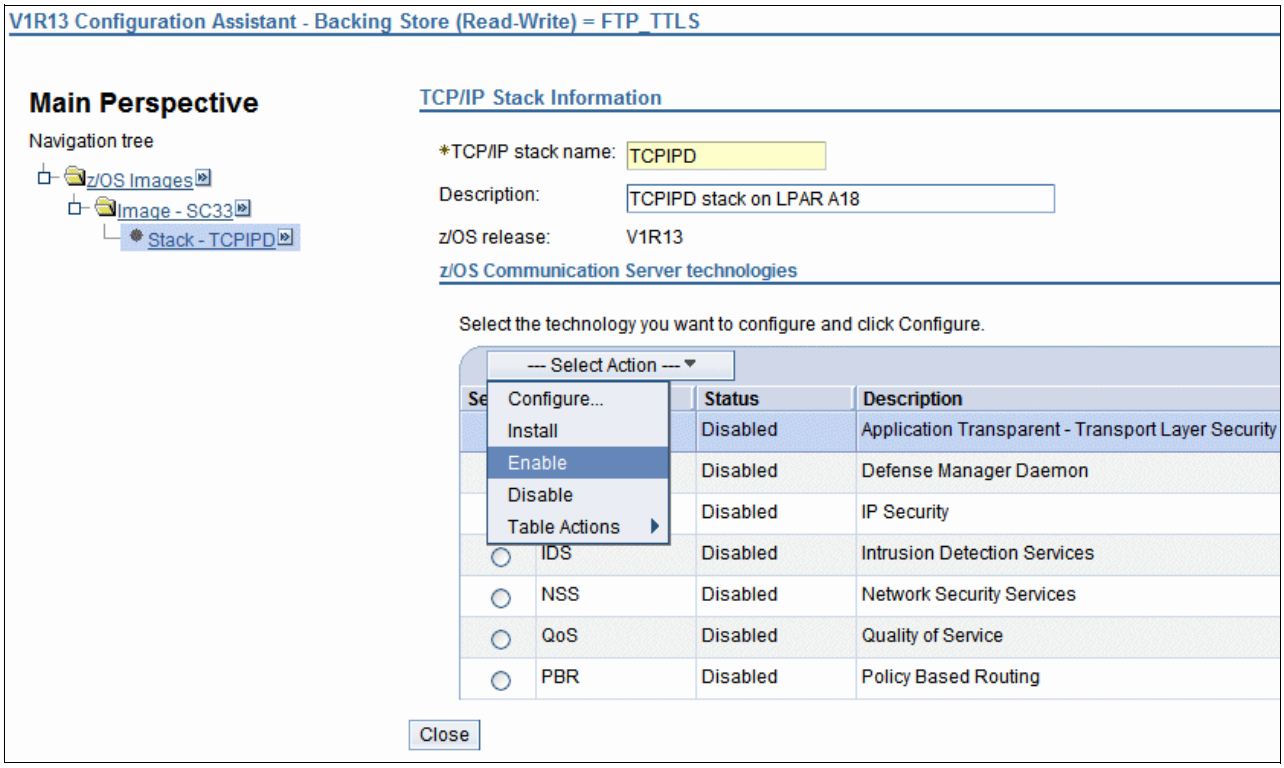


Figure 17-34 Enable AT-TLS

5. The status of the AT-TLS technology changes from `Disabled` to `Incomplete`, as shown in Figure 17-35. A status of incomplete indicates that you need to create the traffic descriptors to build the Policy Rule and Policy Action. Click **Select Action** → **Configure**.

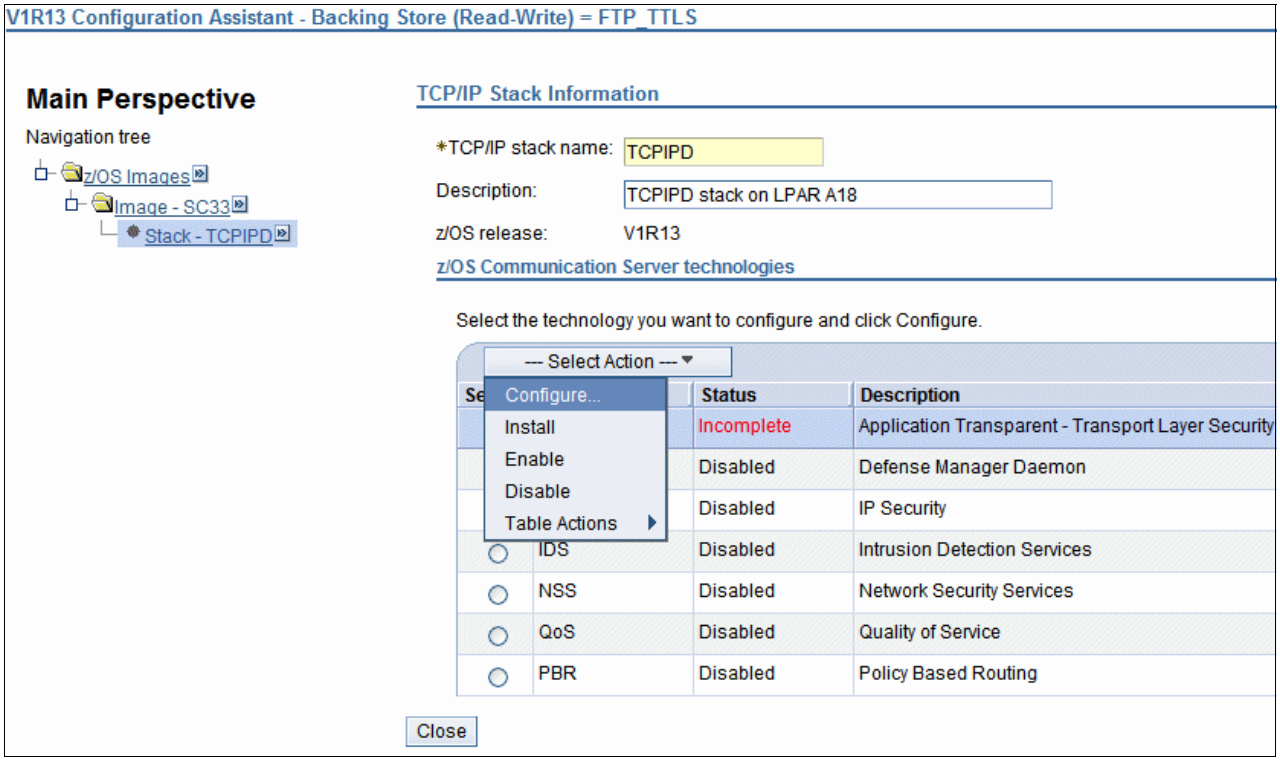


Figure 17-35 Incomplete AT-TLS policy

6. Note the perspective has changed to AT-TLS. Scroll through the available Connectivity Rules until you find `Default_FTP_Server`. Select this rule and click **Select Action** → **Enable Rule** as shown in Figure 17-36. Click **Apply Changes**.

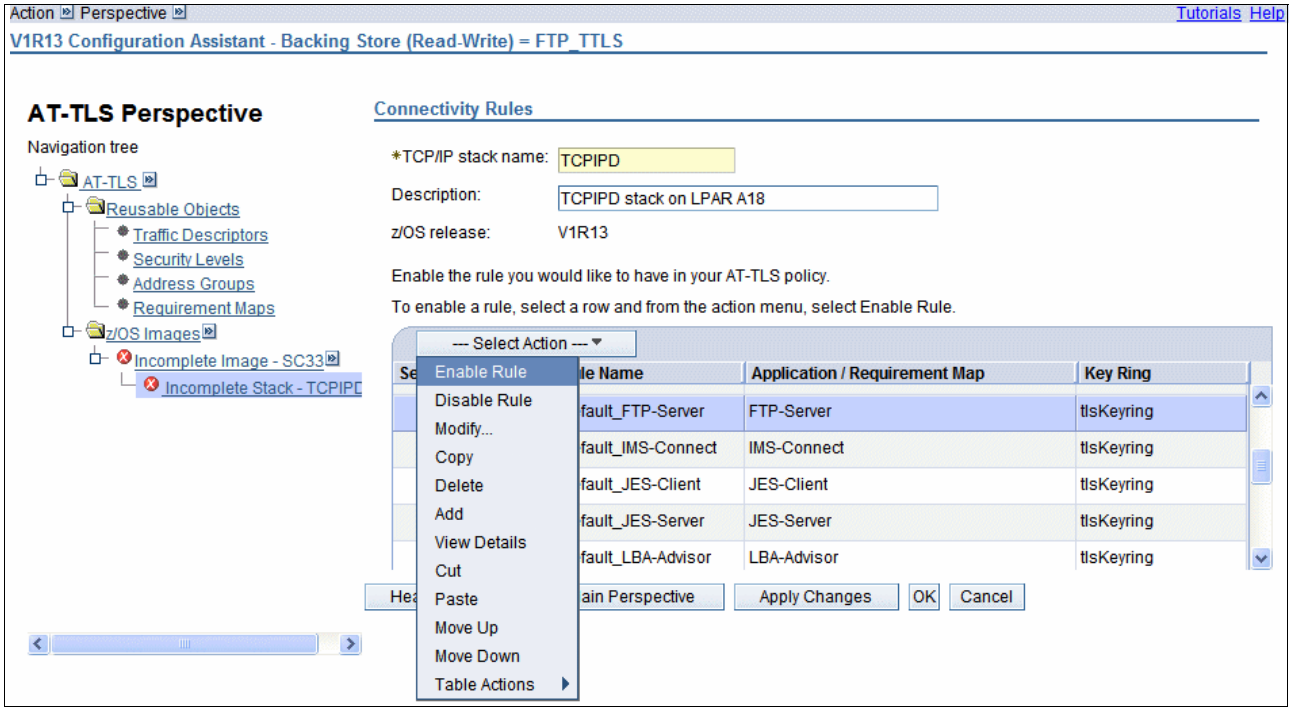


Figure 17-36 Enable AT-TLS FTP Server rule

7. To modify this rule for our environment, select `Default_FTP_Server` and click **Select Action** → **Modify**. We take the defaults for the Traffic tab. The Role tab values cannot be changed. Select the Key Ring tab and enter the key ring name as shown in Figure 17-37. Note we could specify a certificate label from the key ring here, rather than use the default, if required.

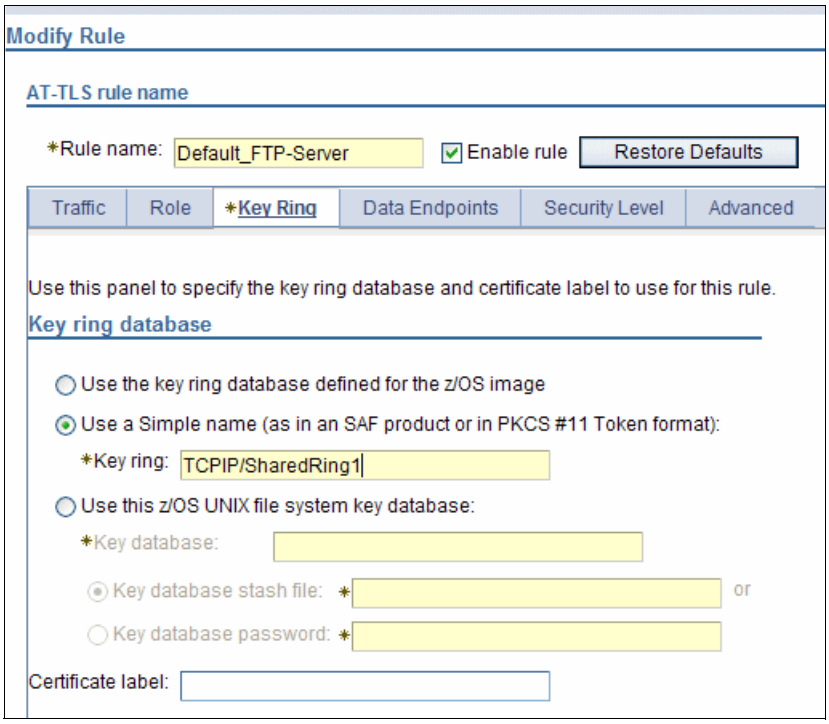


Figure 17-37 Enter key ring name

8. We take the defaults of All_IP_Addresses for the Data Endpoints. Click the Security Level tab and select **AT-TLS_Silver** as shown in Figure 17-38 on page 781. For our example, we were satisfied with the Silver Level of encryption. However, you might consider creating security levels to reflect what is realistic for your environment.

Suggestion: You can build your own Security Level with the IBM Configuration Assistant. Keep in mind that certain encryption algorithms have export restrictions. Furthermore, encryption algorithms can create performance issues, depending on the type of cryptographic hardware that is available on the platform with which you are working.

Therefore, you might want to resequence the Security Level contents, and add or subtract 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.

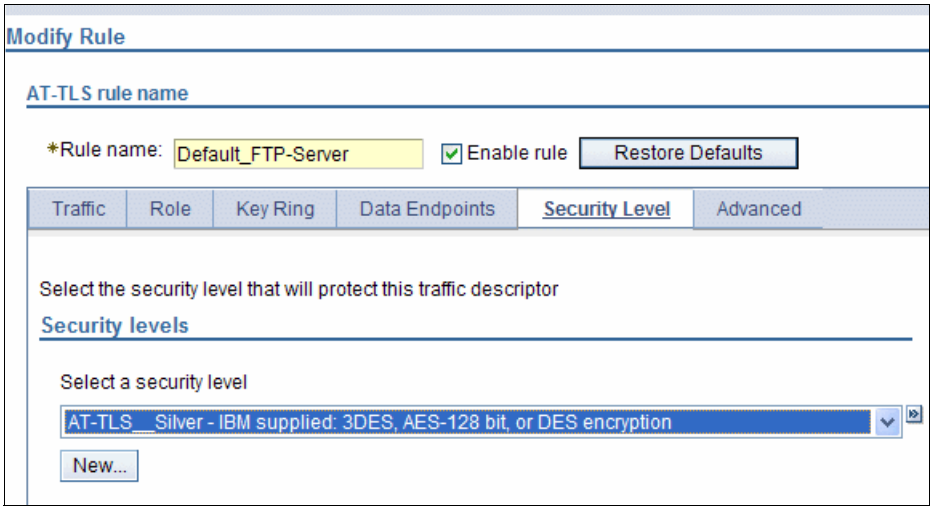


Figure 17-38 Select a security level of AT-TLS_Silver

9. Click the Advanced tab and click **Advanced**. Select the Tuning tab and change the AT-TLS handshake timeout value to 60 as shown in Figure 17-39. This value is not as high as we allowed for native TLS for FTP (100 seconds), but it is still a sizeable number. Click **OK**.

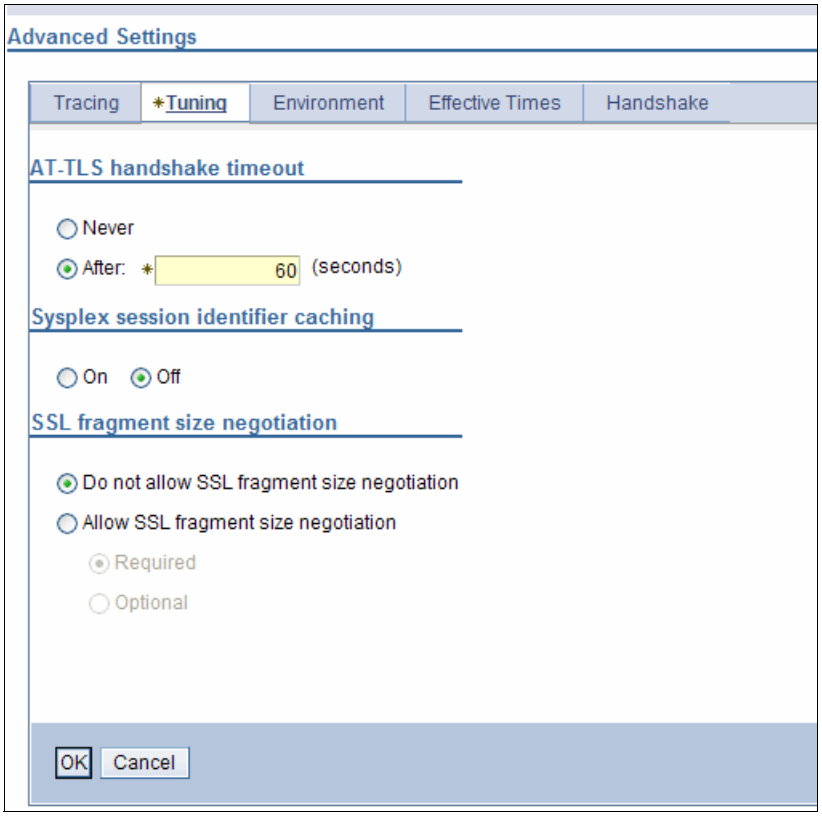


Figure 17-39 Increase handshake timeout

10. Click **OK** to return to the AT-TLS Perspective panel. Note that the Image-SC33 is still marked as `Incomplete` (Figure 17-40). Click **Incomplete Image-SC33** and click **Apply Changes** in the pop-up panel.

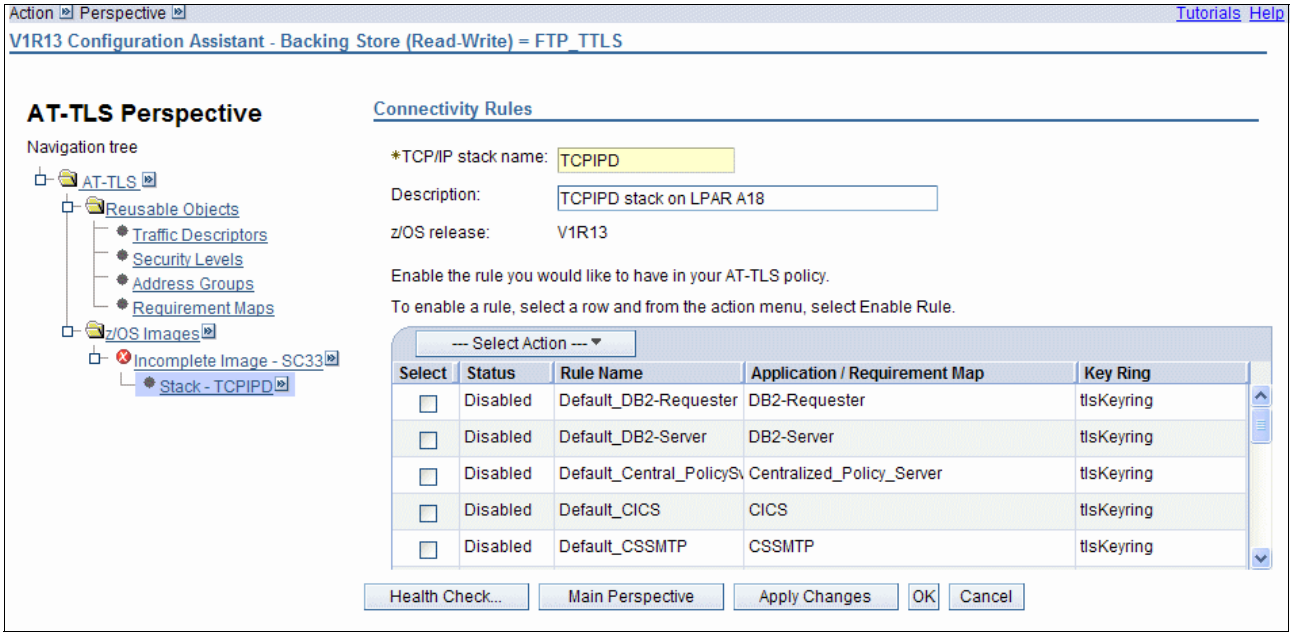


Figure 17-40 Incomplete Image-SC33

11. The image is marked incomplete as z/OSMF Configuration Assistant wants confirmation of the default key ring database, Because we have already defined our key ring name in the `Default_FTP-Server` AT-TLS policy, we can just leave the Settings tab defaults and click **Apply Changes**.

12. In the AT-TLS Perspective panel (Figure 17-41), the navigation tree shows that the z/OS image is now complete; you can install the configuration files. Click the small symbol to the right of Image-SC33 and select **Install Configuration Files**.

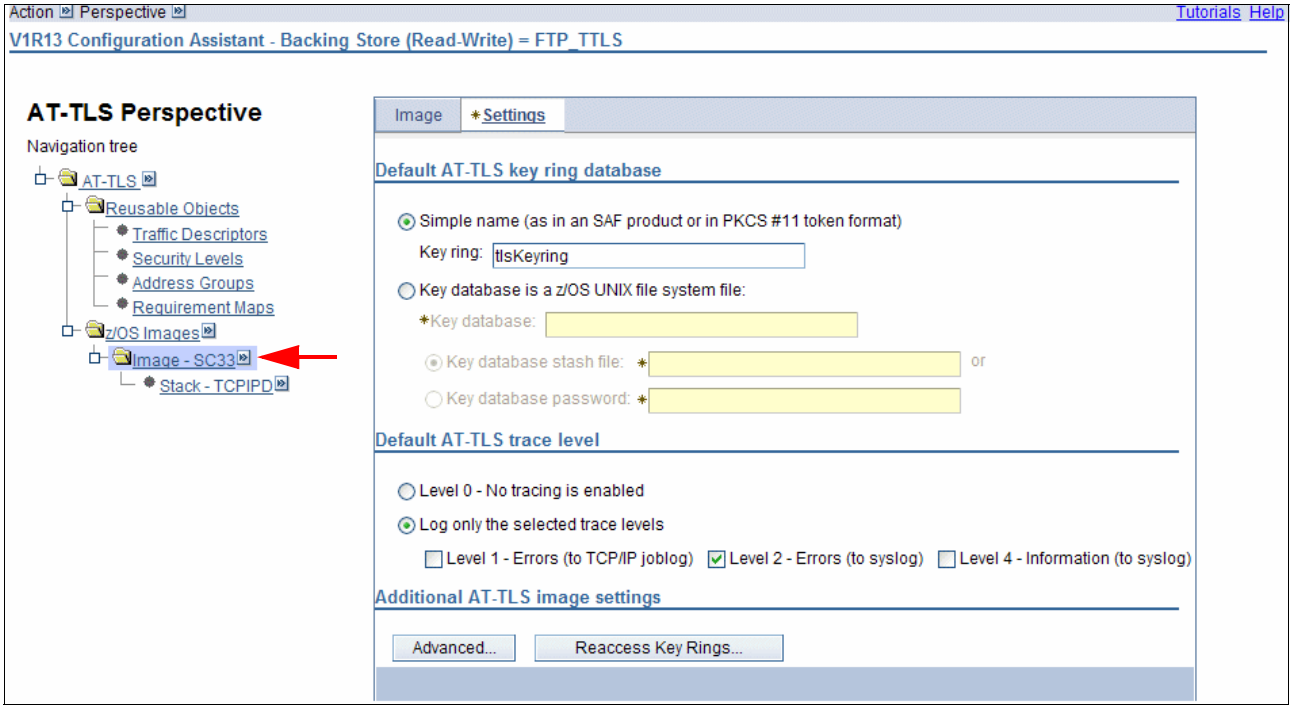


Figure 17-41 AT-TLS perspective image complete

13. Select the AT-TLS Policy and click **Select Action** → **Show Configuration File** to see the policy details that have been created by z/OSMF Configuration Assistant. Ours is shown in Example 17-46.

Example 17-46 FTP-TLS policy details

```
##
## 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 13
## Backing Store = FTP_TTLS
## FTP History:
##
## TLS default rules: Default_FTP-Server
## End TLS default rules
##
## End of Configuration Assistant information
TTLSRule Default_FTP-Server~1
{
LocalAddr ALL
RemoteAddr ALL
LocalPortRangeRef portR1
RemotePortRangeRef portR2
Direction Inbound
Priority 255
```

```
TTLSGroupActionRef gAct1~FTP-Server
TTLSEnvironmentActionRef eAct1~FTP-Server
TTLSConnectionActionRef cAct1~FTP-Server
}
TTLSGroupAction gAct1~FTP-Server
{
TTLSEnabled On
}
TTLSEnvironmentAction eAct1~FTP-Server
{
HandshakeRole Server
EnvironmentUserInstance 0
TTLSKeyringParmsRef keyR1
}
TTLSConnectionAction cAct1~FTP-Server
{
HandshakeRole Server
TTLSCipherParmsRef cipher1~AT-TLS__Silver
TTLSConnectionAdvancedParmsRef cAdv1~FTP-Server
CtraceClearText Off
Trace 2
}
TTLSConnectionAdvancedParms cAdv1~FTP-Server
{
ApplicationControlled On
HandshakeTimeout 60
SecondaryMap On
}
TTLSKeyringParms keyR1
{
Keyring TCPIP/SharedRing1
}
TTLSCipherParms cipher1~AT-TLS__Silver
{
V3CipherSuites TLS_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_RSA_WITH_AES_128_CBC_SHA
}
PortRange portR1
{
Port 21
}
PortRange portR2
{
Port 1024-65535
}
```

14. Click **Close**, then **Select Action** → **Install**. Because z/OSMF Configuration Assistant runs on image SC33, we select **Save to disk** (Figure 17-42 on page 785). In your environment, you might need to use FTP to send the AT-TLS policy to the correct image. We add a meaningful suffix to the default Install file name field, and then click **Go**:

    `/etc/cfgasst/v1r13/SC33/TCPIPD/tlsPol`**`_FTP_Server`**

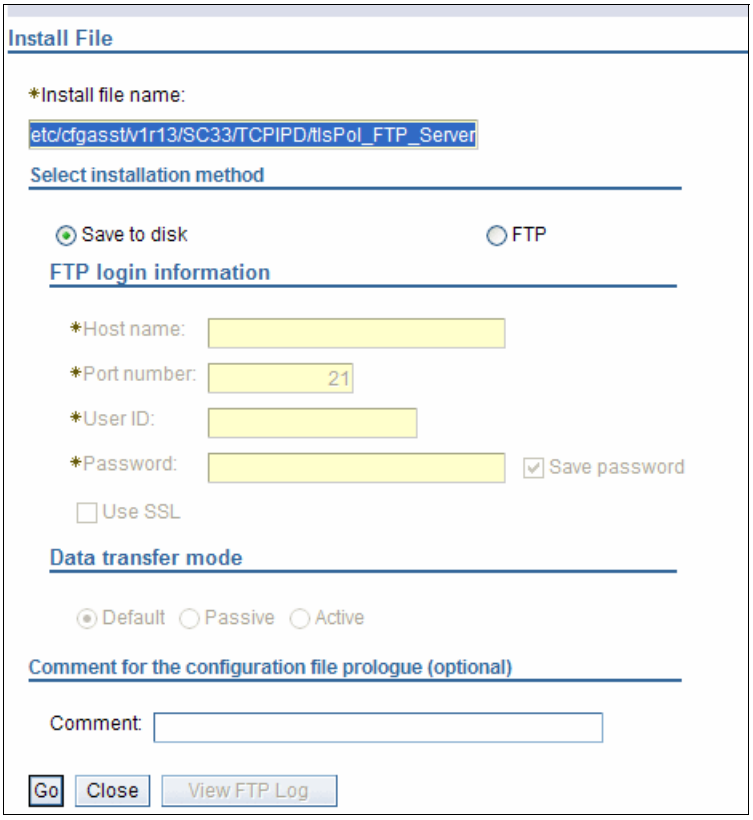


Figure 17-42 Install File

15. When the save is complete, click **Close**. You can optionally enter a comment for the policy file's history log. Click **OK** → **Close**.

Before continuing, remember that you can use more than one method to create an FTP policy rule. Behind the GUI panels, however, important details are inserted into the traffic descriptor for the FTP server. Therefore, select **Traffic Descriptors** as shown in Figure 17-43 and highlight the FTP-Server traffic descriptor object. Then click **Select Action → View Details**.

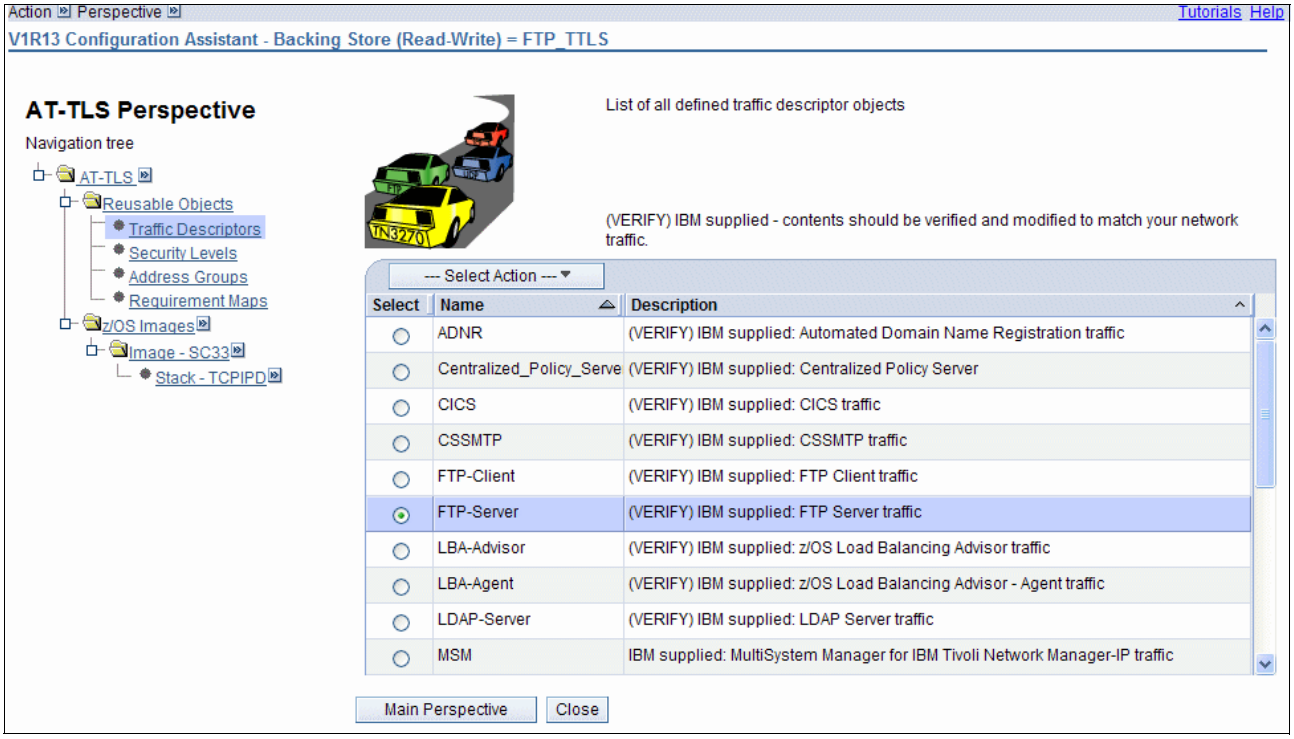


Figure 17-43 Traffic descriptors

The details are displayed (Figure 17-44).

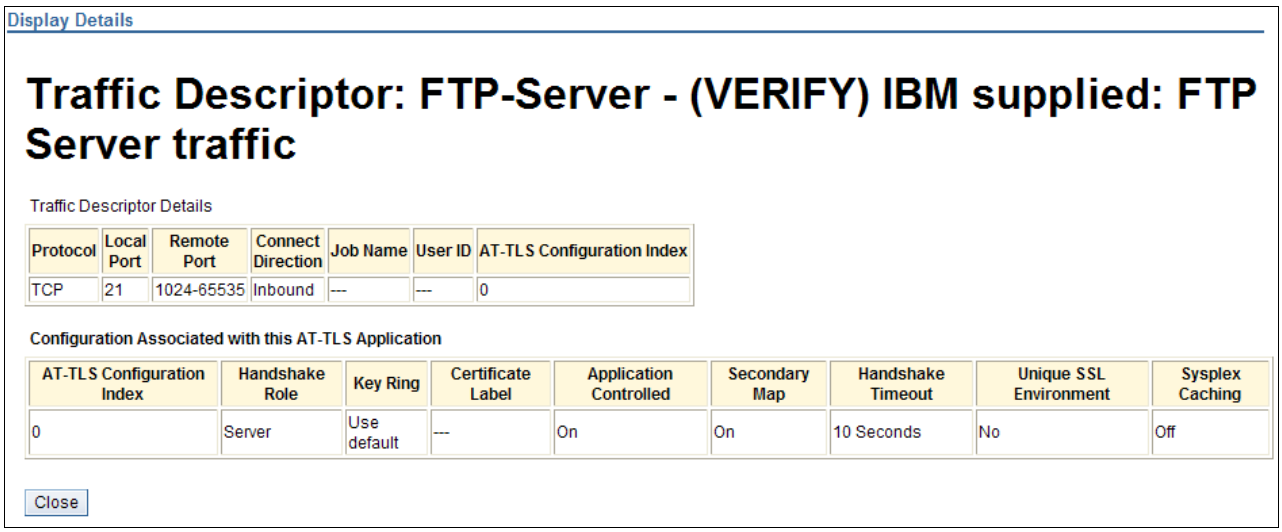


Figure 17-44 Viewing a traffic descriptor for the FTP server

Note that our configuration specifies that we use the default certificate in the key ring file. If we wanted to exploit the full functionality of AT-TLS, we might choose to populate a key ring with many certificates and then to provide a certificate label for a certificate that is not the default.

In our example, the FTP server is also set to *Application Controlled*. You might recall from a description of AT-TLS for FTP that this server is indeed an application-controlled server for AT-TLS.

Secondary Map is enabled for the FTP server, which 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 timeout is set to 10 seconds. This value differs greatly from the SSL time-out default of 100 seconds for standard FTP TLS. We changed this value in our AT-TLS rule's advanced settings to 60 seconds.

Important: If you are migrating from native TLS to AT-TLS, you probably want to consider changing the handshake timeout value to approximate what it was under native TLS to avoid any surprises. Be aware of the situation to establish appropriate TLSTIMEOUT values for your installation.

Update the TCP/IP stack for AT-TLS

This section describes 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 can 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 17-47.

Example 17-47 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 17-48 you see the details for adding authorization for two of the TCP/IP stacks in the installation: for TCPIPD (a) in LPAR A18 (SC33), and for TCPIPB (b) in LPAR A13 (SC31).

Example 17-48 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 12, “Application Transparent Transport Layer Security” on page 551.

Initialization access control for other IP services

It is helpful to permit the `EZB.INITSTACK.sysname.tcpname` RACF profile to other IP services, such as OMPROUTE and SNMP. In a normal installation, these daemons are started by the TCP/IP stack when it starts. However, when AT-TLS is enabled and before the policy agent itself initializes, these applications cannot open sockets, and you will see RACF error messages as shown in Example 17-49.

Example 17-49 Error message before policy agent initialized

```
*EZZ4248E TCPIPD WAITING FOR PAGENT TTLS POLICY 1
...
S OMPD
...
ICH408I USER(OMVSKERN) GROUP(OMVSGRP ) NAME(####################) 581
  EZB.INITSTACK.SC33.TCPIPD CL(SERVAUTH) 2
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
...
EZZ7805I OMPD EXITING ABNORMALLY - RC(11) 2
$HASP395 OMPD     ENDED
```

In this example, the numbers correspond to the following information:

1. A window of time when there is no AT-TLS policy in effect that can represent a security exposure.
2. OMPRoute cannot run because it was not authorized to `EZB.INITSTACK.SC33.TCPIPD`.

Not being able to open sockets is not necessarily a problem. When PAGENT with the AT-TLS policy starts, the messages will stop, and the failed processes that are waiting will continue to initialize. However, if you want to avoid these messages, consider adding RACF definitions, at least for OMPROUTE and the SNMP subagent.

If you decide you want to authorize the other programs, you can do so by authorizing their user IDs to the SERVAUTH profile as shown in Example 17-50.

Example 17-50 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) -
          ACCESS(READ)) a
PERMIT EZB.INITSTACK.SC31.TCPIPB CLASS(SERVAUTH) ID(OMVSKERN,IBMUSER,TCPIP) -
          ACCESS(READ)) a
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

In our example, the processes about which we are concerned are associated with the user IDs OMVSKERN, IBMUSER, and TCPIP. Note how we omitted the `WHENPROGRAM` specification (a) on the RACF PERMIT command.

TCP/IP stack changes

To activate AT-TLS, the TTLS parameter must be added to the TCPCONFIG profile config statement as shown in Example 17-51.

Example 17-51 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.

17.4.3 Activation and verification of FTP AT-TLS support

This section discusses the activation and startup validation of the FTP AT-TLS environment and includes the following topics:

- 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 A18 and received the messages shown in Example 17-52.

Example 17-52 Sample symptoms when SAF INITSTACK permissions are not set

```
S TCPIPD
..
*EZZ4248E TCPIPD WAITING FOR PAGENT TTLS POLICY                    1
 EZZ4202I Z/OS UNIX - TCP/IP CONNECTION ESTABLISHED FOR TCPIPD
 EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
 EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPD ARE AVAILABLE.
 ..
 EZZ8100I OMPD SUBAGENT STARTING
 EZZ7898I OMPD INITIALIZATION COMPLETE                            3
...
 EZZ4250I AT-TLS SERVICES ARE AVAILABLE FOR TCPIPD                2
 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
 IAZ0511I NETSRV1 Server Port number could not be resolved, DEFAULT
 assumed: 175
 EZZ8452I PAGENT READY FOR REMOTE CLIENT CONNECTIONS ON POLICY SERVER
 EZD1576I PAGENT IS READY FOR SERVICES CONNECTION REQUESTS
....
```

In this example, the numbers correspond to the following information:

1. TCP/IP is waiting for PAGENT start. There is a window of time when there is no AT-TLS policy in effect that can represent a security exposure.
2. PAGENT starts, and AT-TLS is enabled.
3. Because we permit the `EZB.INITSTACK.sysname.tcpname` RACF profile to the user who OMPD, our OMPROUTE procedure, is running under, OMPD can start prior to the initialization of PAGENT.

Initializing PAGENT with AT-TLS policies

The PAGENT startup messages in Example 17-53 show that additional policies are loaded, and not just our AT-TLS policies.

Example 17-53 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
EZD1576I PAGENT IS READY FOR SERVICES CONNECTION REQUESTS
```

We want to examine whether the AT-TLS policies that loaded are really ours. We move to the UNIX System Services environment from TSO by using 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 to examine only the AT-TLS policies. We redirected the output to a file in our user directory:

```
pasearch -t > myttlspolicies
```

Example 17-54 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 17-55.

Example 17-55 pasearch output from UNIX System Services

```
TCP/IP pasearch CS V1R13                  Image Name: TCPIPD
  Date:                 08/03/2011        Time:  08:26:39
  TTLS Instance Id:     1312374380

policyRule:             Default_FTP-Server~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:         gAct1~FTP-Server                                  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)
   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:           All
    ToAddr:             All
   Remote Address:
```

```
   FromAddr:           All
   ToAddr:             All
  LocalPortFrom:       21              LocalPortTo:       21
  RemotePortFrom:      1024            RemotePortTo:      65535
  JobName:                             UserId:
  ServiceDirection:    Inbound
 Policy created: Wed Aug  3 08:26:20 2011
 Policy updated: Wed Aug  3 08:26:20 2011

 TTLS Action:                  gAct1~FTP-Server                    c
   Version:                    3
   Status:                     Active
  Scope:                      Group                               c1
  TTLSEnabled:                On
  CtraceClearText:            Off
  Trace:                      2
  FIPS140:                    Off
  TTLSGroupAdvancedParms:
   SecondaryMap:              Off
   SyslogFacility:            Daemon
  Policy created: Wed Aug  3 08:26:20 2011
  Policy updated: Wed Aug  3 08:26:20 2011

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
   TLSv1.1:                   On
   ApplicationControlled:     Off
   HandshakeTimeout:          10
   ClientAuthType:            Required
   ResetCipherTimer:          0
   TruncatedHMAC:             Off
   CertValidationMode:        Any
   ServerMaxSSLFragment:      Off
   ClientMaxSSLFragment:      Off
   ServerHandshakeSNI:        Off
   ClientHandshakeSNI:        Off
  EnvironmentUserInstance:    0
  Policy created: Wed Aug  3 08:26:20 2011
  Policy updated: Wed Aug  3 08:26:20 2011

TTLS Action:                  cAct1~FTP-Server                    e
  Version:                    3
  Status:                     Active
  Scope:                      Connection                          e1
  HandshakeRole:              Server
  CtraceClearText:            Off
```

```
Trace:                        2
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
Policy created: Wed Aug  3 08:26:20 2011
Policy updated: Wed Aug  3 08:26:20 2011
```

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 17-55 on page 791.

However, note that there is only one Group Action (**c1**), although the other actions (Environment **d1** and Connection **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 are other ways to see the results of **pasearch** without entering the UNIX shell:

- Use the EZACMD System REXX command to issue **pasearch** from the z/OS console, TSO or NetView. For more information about using EZACMD, see 2.6.5, “EZACMD console command security” on page 29 and also the following resources:
  - Section 8.7, “MVS console support for selected TCP/IP commands” in *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7996
  - *z/OS Communications Server: IP System Administrator’s Commands*, SC31-8781
- A REXX exec that can be invoked from TSO or NetView to execute the **pasearch** command or any other UNIX command for you.

Example 17-56 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 17-57 we have executed the **netstat** command with the **ttls** option.

Example 17-57 Sample NETSTAT TTLS command output

```
D TCPIP,TCPIPD,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPD 451
TTLSGRPACTION                          GROUP ID           CONNS
GACT1~FTP-SERVER                       00000002               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

After starting the FTP server and having FTP connections in place, view what the syslog daemon log indicates about the FTP startup and the client's login by using TLS.

The syslog daemon isolation that we set up for the FTP procedure named FTPDD collects messages in the `/var/syslog/%Y/%m/%d/ftpd.log` file. Browsing the log file covering the current date from the UNIX shell, as shown in Example 17-58 on page 795, 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 17-58 Syslogd output for the FTP server and the client login

```
DM1133 main: Initialization parameter values:
DM1136 ..localsite values ------------------------
..
DM1317 ..security mechanism values ---------------   a
DM1319 ....SECURE_FTP.............A
DM1321 ....SECURE_LOGIN............N
DM1323 ....SECURE_PASSWORD..........R
DM1325 ....SECURE_PASSWORD_KERBEROS..R
DM1327 ....SECURE_CTRLCONN..........P
DM1329 ....SECURE_DATACONN..........C
DM1331 ....SECURE_PBSZ.............16384
DM1333 ....KEYRING..................
DM1335 ....CIPHERSUITE..............
DM1337 ....TLSPORT..................990
DM1339 ....TLSTIMEOUT..............100
DM1345 ....TLSRFCLEVEL.............DRAFT
DM1349 ....TLSMECHANISM............ATTLS
DM1352 ....securePort..............FALSE
DM1355 ....secureImplictiZos.........TRUE
..
DM1549 main: daemon's code page is: IBM-1047
DM1552 main: daemon's locale is: C
DM1564 main: environment variable: TZ=EST5EDT4
DM1564 main: environment variable: _BPXK_SETIBMOPT_TRANSPORT=TCPIPD  b
DM1564 main: environment variable: _BPX_JOBNAME=FTPDDDD
DM1564 main: environment variable: _EDC_ADD_ERRNO2=1
EZY2700I Using port FTP control (21)
..
SD0438 accept_client: accept()
SD0505 accept_client: accepted client on socket 8
SD1677 handle_client_socket: entered for socket 8
SD1899 handle_client_socket: new session for ::ffff:10.1.1.20 port 1057    c
SD1160 spawn_ftps: entered
SD0372 accept_client: prepare to accept another client
SD0391 accept_client: calling selectex for socket 7
SD1180 spawn_ftps: my pid is 33816668 and my parent's is 67371091
GU4900 initADcontainer: entered
GU4921 initSIOCSAPPLDATAInput: entered
GU4945 ftpSetApplData: entered
BU0936 good_morning: entered
EZYFS50I ID=FTPDD100000 CONN   starts Client IPaddr=::ffff:10.1.1.20
hostname=wtsc31.itso.ibm.com
SD0781 setup_new_pgm: entered
SD0927 setup_new_pgm: msgcat->_name = /usr/lib/nls/msg/C/ftpdmsg.cat
SD1112 setup_new_pgm: issuing execv
GU1137 chkVerRel: system information for SC33: z/OS version 1 release 13 (2817)
PR0308 parse_cmd: entered
SR1232 get_command: UTF8_on = 0
PR0490 parse_cmd: >>> AUTH TLS
FR0186 auth: entered with mechname TLS
FR2827 ftpAuthAttls: Query AT-TLS state for socket
FR2878 ftpAuthAttls: AT-TLS policy set as application controlled.
FU1367 TTLSRule: Default_FTP-Server~1
```

```
FU1373 TTLSGroupAction: gAct1~FTP-Server
FU1379 TTLSEnvironmentAction: eAct1~FTP-Server
FU1386 TTLSConnectionACtion: cAct1~FTP-Server
SR3397 reply: --> 234 Security environment established - ready for negotiation
FR2557 authClientAttls: Start TLS handshake
FR2588 authClientAttls: FIPS140 not enabled
FR2606 authClientAttls: Using TLSv1 protocol
FR2620 authClientAttls: SSL cipher: 09
FR2683 getUseridAttls: no client certificate available
EZYFS54I ID=FTPDD100000 SECURE OK     Mechanism=TLS
GU4945 ftpSetApplData: entered
SR1232 get_command: UTF8_on = 0
PR0490 parse_cmd: >>> USER cs03
RA2225 user: username = CS03, authusername =
SR3397 reply: --> 331 Send password please.
GU4945 ftpSetApplData: entered
SR1232 get_command: UTF8_on = 0
PR0495 parse_cmd: >>> PASS ******
```

Note that a in Example 17-58 on page 795 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 c, the client IP address is that of the TCPIPB TSO client at 10.1.3.21. Examine the output from the **netstat ttls** command in Example 17-59; d shows that we have one connection active to AT-TLS.

Example 17-59 Netstat TTLS connections

```
D TCPIP,TCPIPD,N,TTLS
EZD0101I NETSTAT CS V1R13 TCPIPD 488
TTLSGRPACTION                          GROUP ID           CONNS
GACT1~FTP-SERVER                       00000002               1   d
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

Example 17-60 shows the detail on the Action Group that is active.

Example 17-60 Netstat TTLS Action Group information

```
D TCPIP,TCPIPD,N,TTLS,GROUP,DETAIL
EZD0101I NETSTAT CS V1R13 TCPIPD 492
TTLSGRPACTION:   GACT1~FTP-SERVER
  GROUPID:         00000002
  TASKS:           4                   GROUPCONNS:      1
  WORKQELEMENTS:   0                   SYSLOGQELEMENTS: 0
    ENV: EACT1~FTP-SERVER                          ENVCONNS: 1
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

Example 17-61 displays the connection ID of an FTP connection using a standard **netstat conn** command. From this command we can obtain the connection ID for the FTP connection (a). Then we can use **netstat ttls** command to display the TTLS information for this particular connection (b).

Example 17-61 Specific connection information for an ATTLS connection

```
D TCPIP,TCPIPD,N,CONN,PORT=21
EZD0101I NETSTAT CS V1R13 TCPIPD 494
USER ID  CONN     STATE
FTPDD1   00000125 ESTBLSH                    a
  LOCAL SOCKET:   ::FFFF:10.1.1.40..21
  FOREIGN SOCKET: ::FFFF:10.1.1.20..1059
FTPDD1   00000020 LISTEN
  LOCAL SOCKET:   ::..21
  FOREIGN SOCKET: ::..0
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT

D TCPIP,TCPIPD,N,TTLS,CONN=125,DETAIL            b
EZD0101I NETSTAT CS V1R13 TCPIPD 498
CONNID: 00000125
  JOBNAME:      FTPDD1
  LOCALSOCKET:  ::FFFF:10.1.1.40..21
  REMOTESOCKET: ::FFFF:10.1.1.20..1059
  SECLEVEL:     TLS VERSION 1
  CIPHER:       09 TLS_RSA_WITH_DES_CBC_SHA
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
TTLSRULE: DEFAULT_FTP-SERVER~1
  PRIORITY:       255
  LOCALADDR:      ALL
  LOCALPORT:      21
  REMOTEADDR:     ALL
  REMOTEPORTFROM: 1024              REMOTEPORTTO: 65535
  DIRECTION:      INBOUND
  TTLSGRPACTION:  GACT1~FTP-SERVER
    GROUPID:                    00000002
    TTLSENABLED:                ON
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    SYSLOGFACILITY:             DAEMON
    SECONDARYMAP:               OFF
    FIPS140:                    OFF
  TTLSENVACTION:  EACT1~FTP-SERVER
    ENVIRONMENTUSERINSTANCE:    0
    HANDSHAKEROLE:              SERVER
    KEYRING:                    TCPIP/SHAREDRING1
    SSLV2:                      OFF
    SSLV3:                      ON
    TLSV1:                      ON
    TLSV1.1:                    ON
    RESETCIPHERTIMER:           0
    APPLICATIONCONTROLLED:      OFF
    HANDSHAKETIMEOUT:           10
```

```
    TRUNCATEDHMAC:              OFF
    CLIENTMAXSSLFRAGMENT:       OFF
    SERVERMAXSSLFRAGMENT:       OFF
    CLIENTHANDSHAKESNI:         OFF
    SERVERHANDSHAKESNI:         OFF
    CLIENTAUTHTYPE:             REQUIRED
    CERTVALIDATIONMODE:         ANY
  TTLSCONNACTION: CACT1~FTP-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
    CTRACECLEARTEXT:            OFF
    TRACE:                      2
    APPLICATIONCONTROLLED:      ON
    HANDSHAKETIMEOUT:           60
    SECONDARYMAP:               ON
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

17.5 Migrating from native FTP TLS to FTP AT-TLS

This section discusses migrating from the native FTP TLS environment to the FTP ATTLS environment.

17.5.1 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 z/OSMF Configuration Assistant.

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 migrate it to the current IBM Configuration Assistant for z/OS Communications Server. For details, see 4.6.2, “Migrating backing store files to z/OSMF Configuration Assistant” on page 153. Then, simply add the new AT-TLS policy to the suite of policies already defined.

17.5.2 Details on migrating from TLS to AT-TLS

Migrating from native FTP TLS to FTP with AT-TLS is fairly 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
- Defining AT-TLS instead of native TLS
- New procedures that are necessary for migration
- Changes to security that are required for migration
- Changes to the TCP/IP stack that 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 17-1 shows which statements must migrate to the policy agent file.

Table 17-1 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

Table 17-2 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 17-2 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

Defining 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.

New procedures that are necessary for migration

For AT-TLS, you need a functioning policy agent. See Chapter 4, “Policy agent” on page 103 for information about how to accomplish this task.

Changes to security that 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 726 if you need more information.

Policy agent requires additional RACF authorizations, as do the users who need access to the PAGENT commands like **pasearch**. See Chapter 4, “Policy agent” on page 103 for information about how to accomplish this task.

Special authorizations in the SERVAUTH facility class are required 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 12, “Application Transparent Transport Layer Security” on page 551, or consult “TCP/IP stack initialization access control” on page 787 for more information about this topic.

Changes to the TCP/IP stack that 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 789.

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.

17.6 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 issue, first check the console and `syslogd` for error messages. For more details and for references to alternate publications within the z/OS family, consult the following resources:

- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786

Look for the error codes and return codes and consult the IP messages volumes mentioned here. You might also have to consult UNIX System Services publications and z/OS Integrated Security Services publications for specialized error codes, such as 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.

17.7 Additional information

For additional information about FTP security, see the following resources:

- *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, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Part 5

Appendixes

Because security technologies can be complex, this part of the book includes tutorial information and includes the following appendixes.

Appendix	Topic
Appendix A, “Basic cryptography” on page 805	Security issues; secret keys and their algorithms; public and private keys, and digital certificates; crypto systems and performance; and message integrity
Appendix B, “Telnet security advanced settings” on page 823	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 859	Steps used for our IPSec scenario between z/OS and Windows: setting up the IKE daemon, certificates, the z/OS IPSec policy, and a Windows IPSec policy, and verification that things are working
Appendix D, “zIIP Assisted IPSec” on page 903	Configuration of zIIP Assisted IPSec and an example of zIIP Assisted IPSec implementation
Appendix E, “z/OS Communications Server IPSec RFC currency” on page 919	Lists RFCs that are relevant to z/OS Communications Server
Appendix F, “Our implementation environment” on page 921	The complete environment used for the four *IBM z/OS Communications Server TCP/IP Implementation* books and the environment that we used for this book

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 other untrusted network.

We discusses the following topics in this appendix.

Section	Topic
“Potential problems with electronic message exchange” on page 806	An example to illustrate the security issues with electronic message exchanges.
“Secret key cryptography” on page 810	The basic concepts surrounding secret keys and the algorithms used for those keys.
“Public key cryptography” on page 812	The basic concepts surrounding public keys where public and private keys are used, and also digital certificates and their role in the secure use of public key cryptography.
“Performance issues of cryptosystems” on page 818	Performance concerns when using cryptography
“Message integrity” on page 818	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 a 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 following items:

- 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).

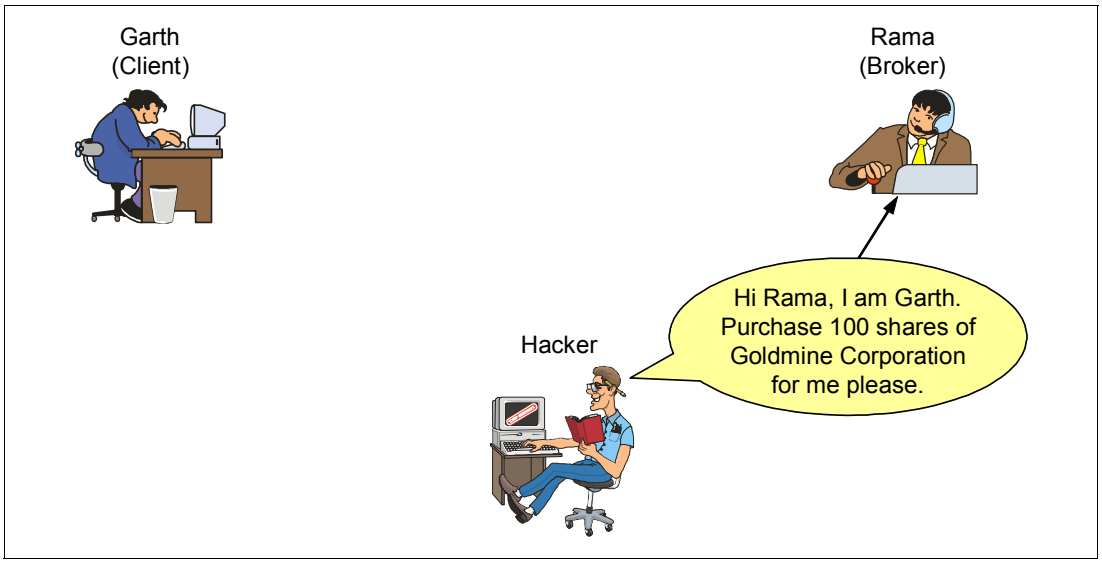


Figure A-1 Hacker posing as a genuine client

What is needed here is a way to ensure that the client is who he says he is. In some cases, this way must involve a sort of shared secret, such as a password, which is called *user authentication*. “Authentication” on page 813 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.

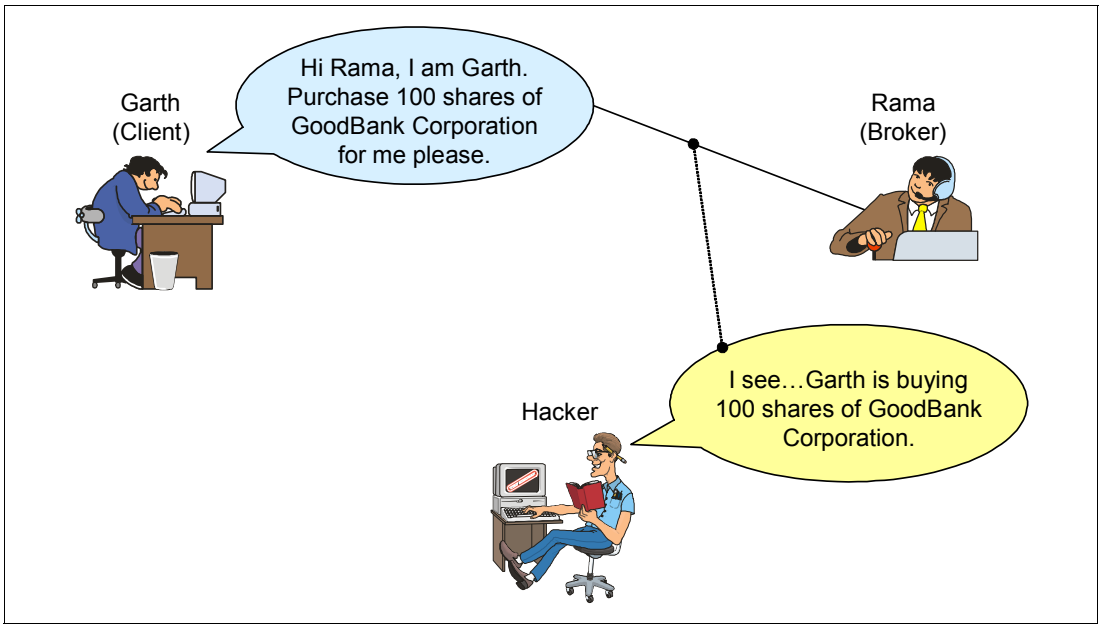


Figure A-2 Hacker intercepting the order

Assume that you have a 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 812 explains this technology.

The order could have been intercepted and altered

Figure A-3 shows the hacker altering the original message.

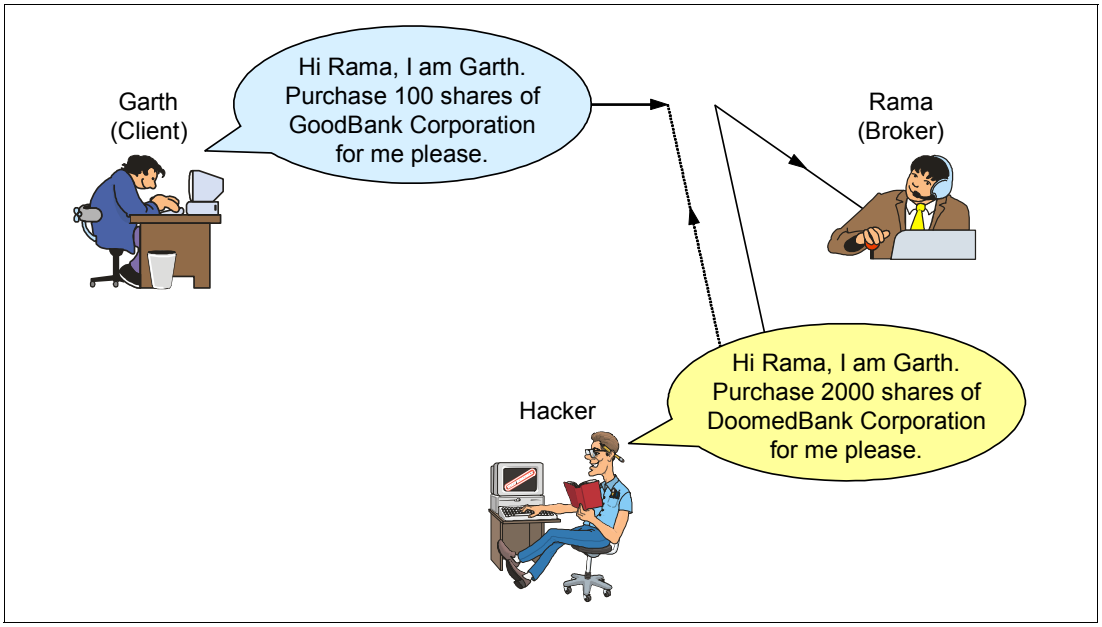


Figure A-3 Hacker intercepting and altering the order

How do you know, when you receive a message, that the contents of the message have not been modified? What is needed is a 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 820, and “Digital signatures” on page 821.

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.

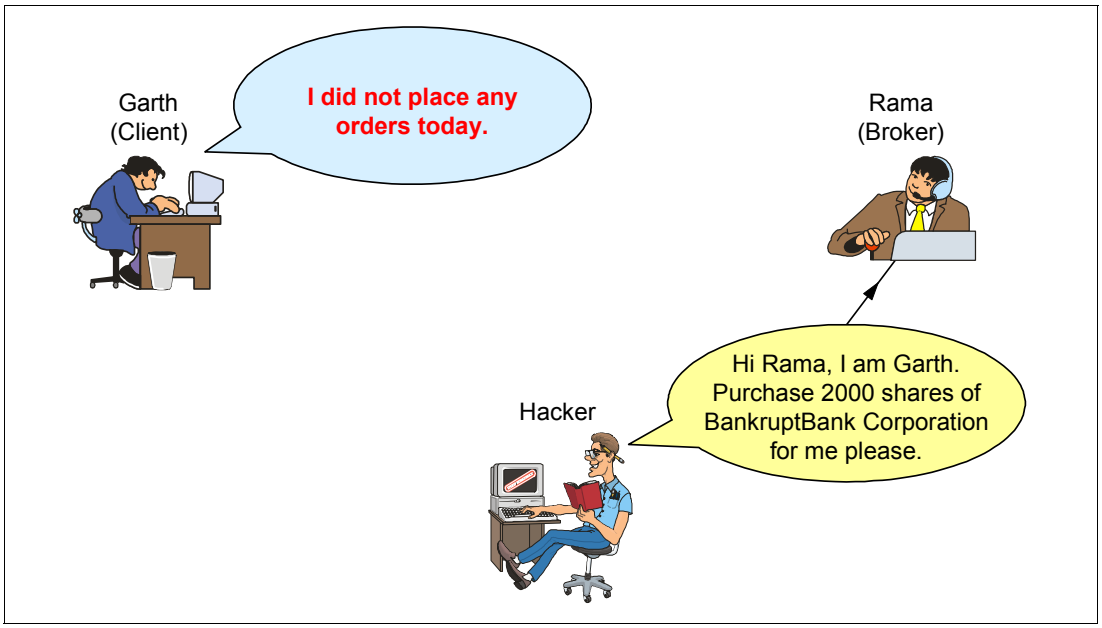


Figure A-4 Hacker placing an order

What is needed is a 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 821.

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 812. 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.

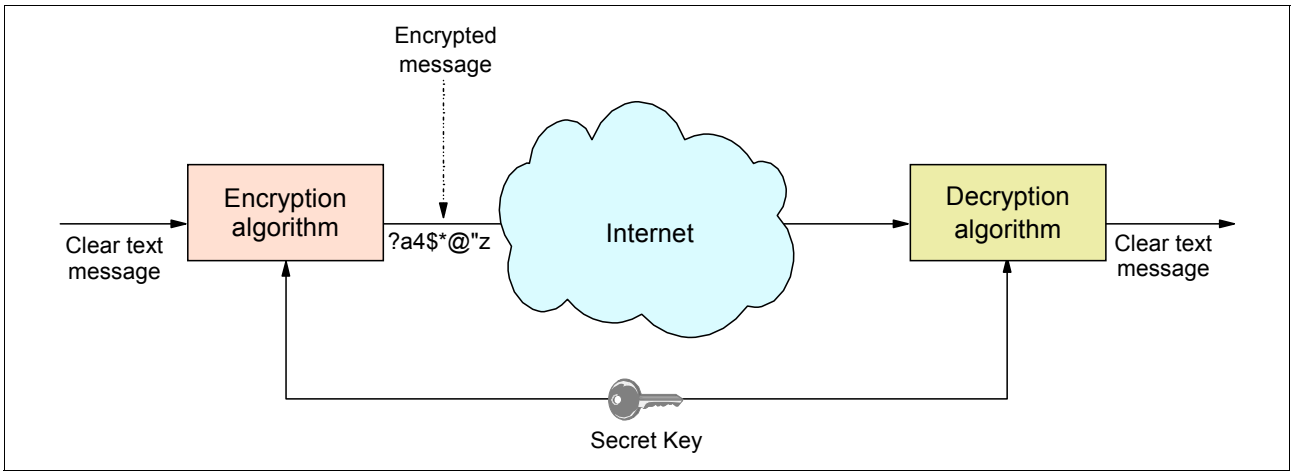


Figure A-5 Symmetric encryption and decryption: Using the same key

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 encrypting 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 need to communicate securely, it is extremely difficult to manage the distribution of keys so as not to compromise the integrity of any of them. Public key encryption, described in “Public key cryptography” on page 812, can be used to exchange secret keys securely, and from then onward, the conversation can use the faster secret key encryption.

The following items are frequently-used symmetric algorithms:

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. The *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. Also, 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 more are seen as unbreakable for many years to come because the work factor rises exponentially with the size of the key.

Tip: 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 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), have the following 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.

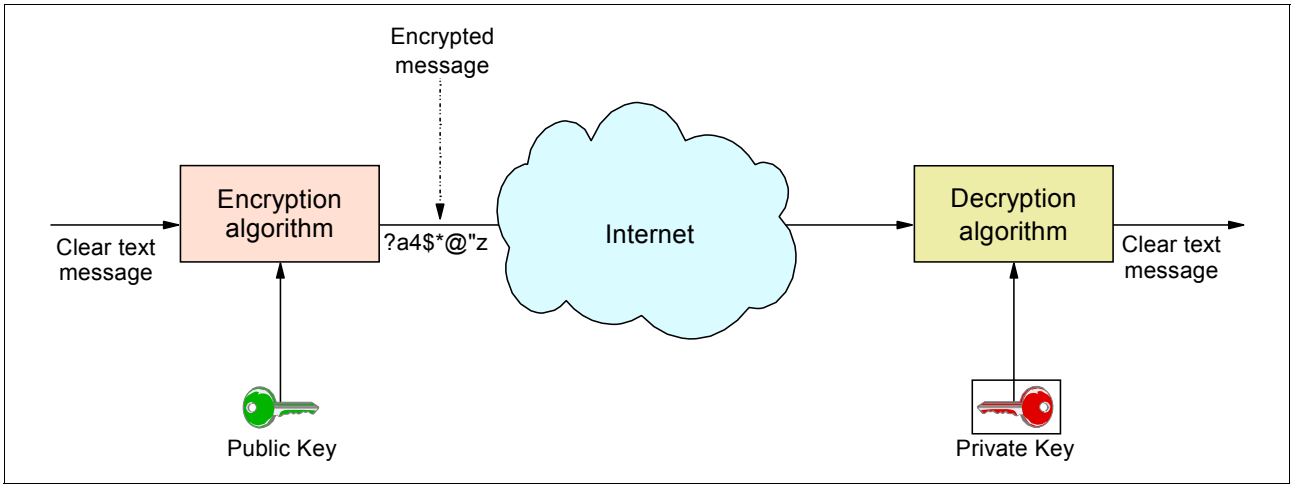


Figure A-6 Public-key cryptography: encryption using a public key

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 useful for authentication. What happens if you encrypt a message using your own private key is shown in Figure A-7.

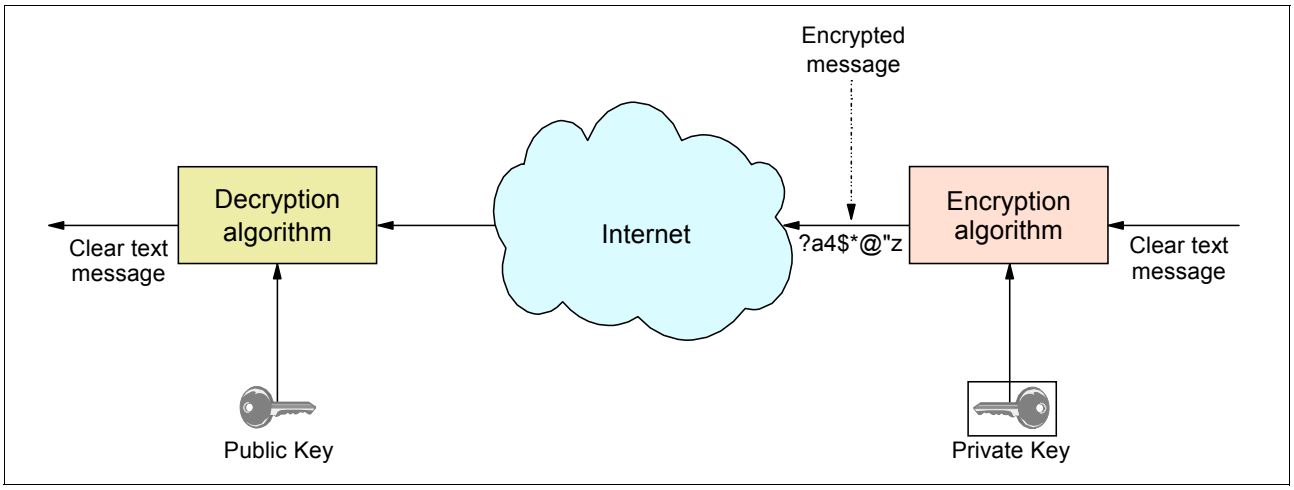


Figure A-7 Public-key cryptography: encryption using a private key results in authentication

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 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.

Diffie-Hellman W. Diffie and M.E. Hellman, the inventors of public key cryptography, published this algorithm in 1976. 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 sometimes used for authentication of certificates during an TLS/SSL handshake such as DSA and RSA. Also, 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 (Figure A-8).

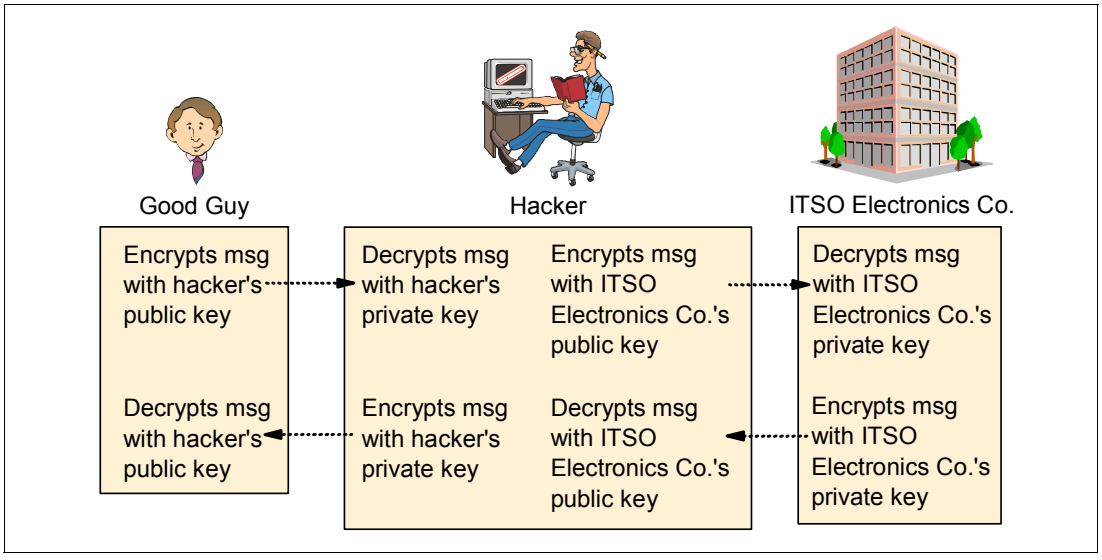


Figure A-8 Scenario where public key being used by good guy is really a hacker's key

In Figure A-8 on page 814, 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 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 are 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. Finally, 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, see *IMS in the Parallel Sysplex Volume II: Planning the IMSplex*, SG24-6928.

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 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 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 an important concept in passports, and 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 contents of the certificate, including the certificate owner's identity, is digitally signed by the CA. That is, a message digest is calculated from the entire contents of the certificate. Then this message digest is encrypted with the private key of the CA to form the digital signature.

Figure A-9 shows a simplified layout of a digital certificate.

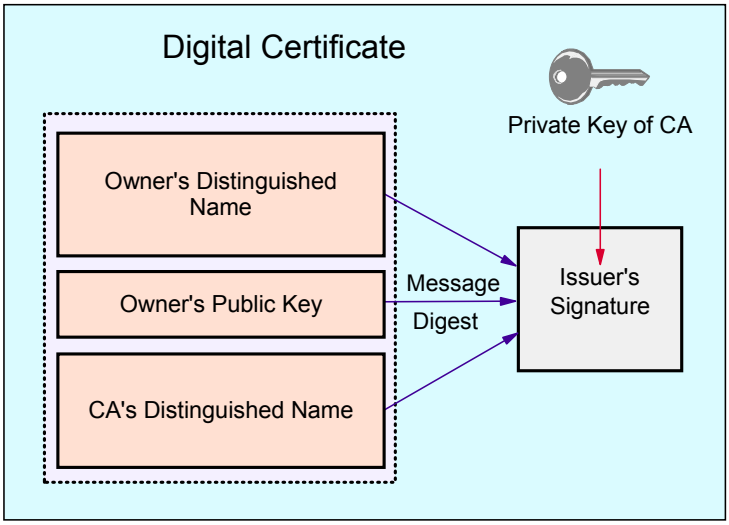


Figure A-9 Simplified layout of a digital certificate

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 can 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 to obtain that public key. In general, a chain of multiple certificates might 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 website. After verifying the validity of the request, the CA sends back the certificate in an email message or allows it to be downloaded.

This is obviously an oversimplification. For more complete details about certificates, see Chapter 3, "Certificate management in z/OS" on page 39.

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 39.

Performance issues of cryptosystems

Elliptic curve cryptosystems are said to have performance advantages over RSA and non-elliptical DSA in decryption and signing. Although 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, whereas asymmetric cryptosystems are used for electronic s 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 might or might 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 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.

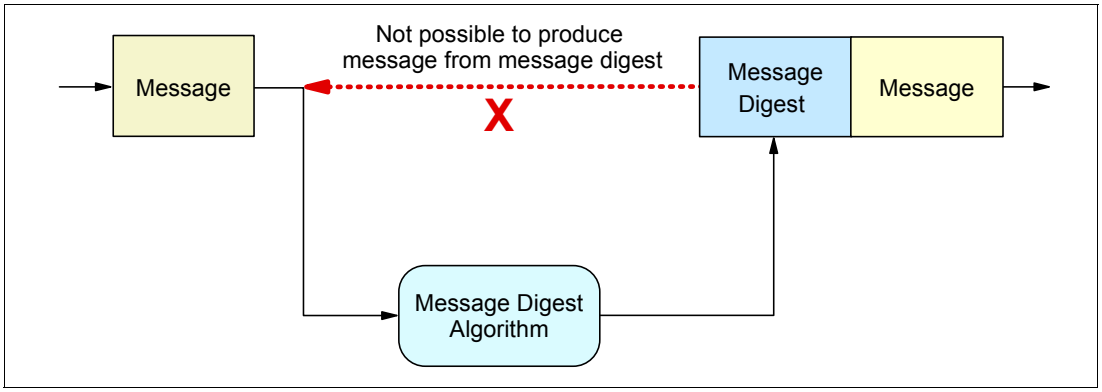


Figure A-10 Message digest

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 820.

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. Because 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, whereas 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 can recompute the message digest and encrypt it with the shared key. This result should match the MAC sent on the message.

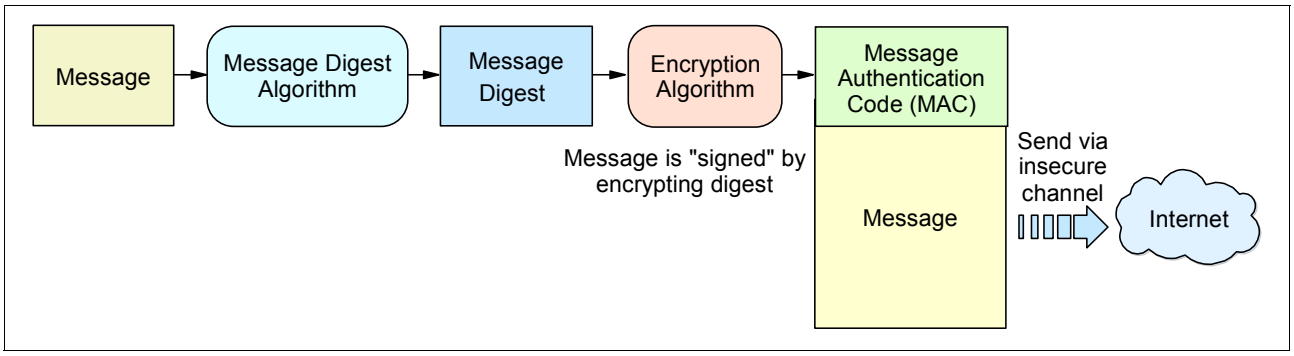


Figure A-11 Message digest for data integrity

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 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 821.

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, a few 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. Although 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 a signature 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 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.

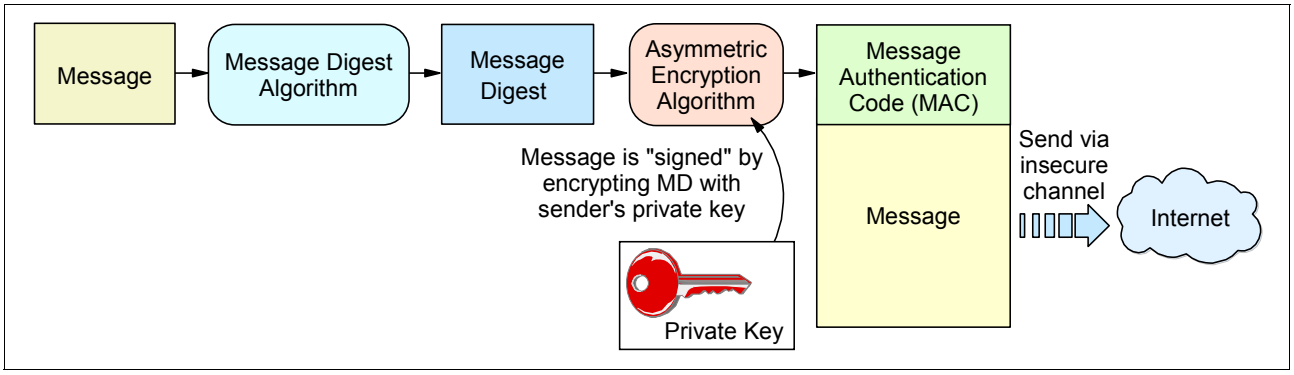


Figure A-12 Digital creation

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.

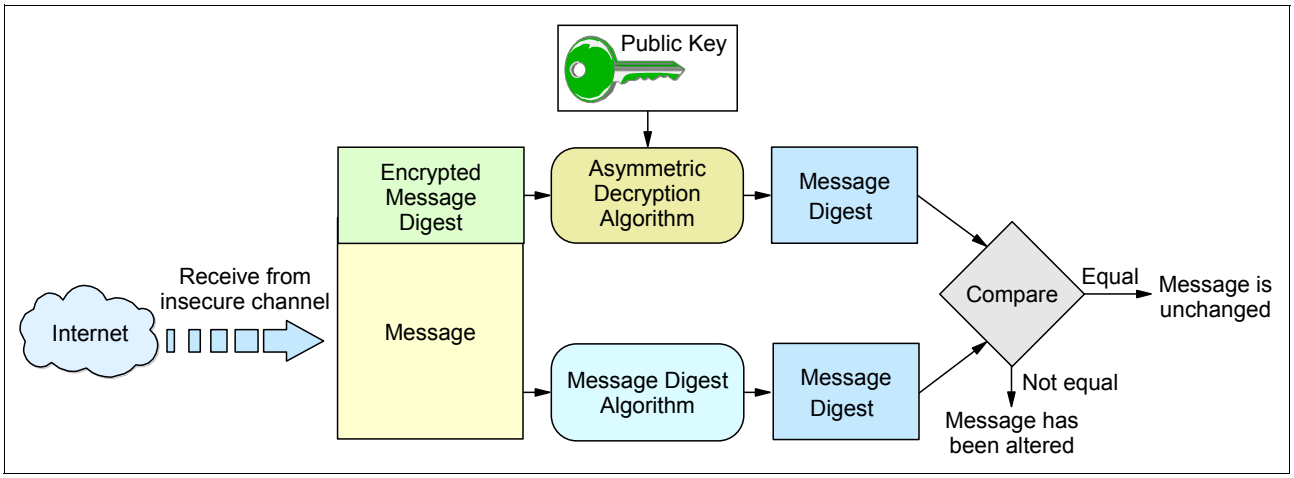


Figure A-13 Digital verification

With digital signatures, only public-key cryptosystems can be used. If secret-key cryptosystems are used to encrypt the signature, it will be 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. Thus, anybody can send encrypted data to the receiver that only the receiver can decrypt. See Figure A-14 for a graphical representation.

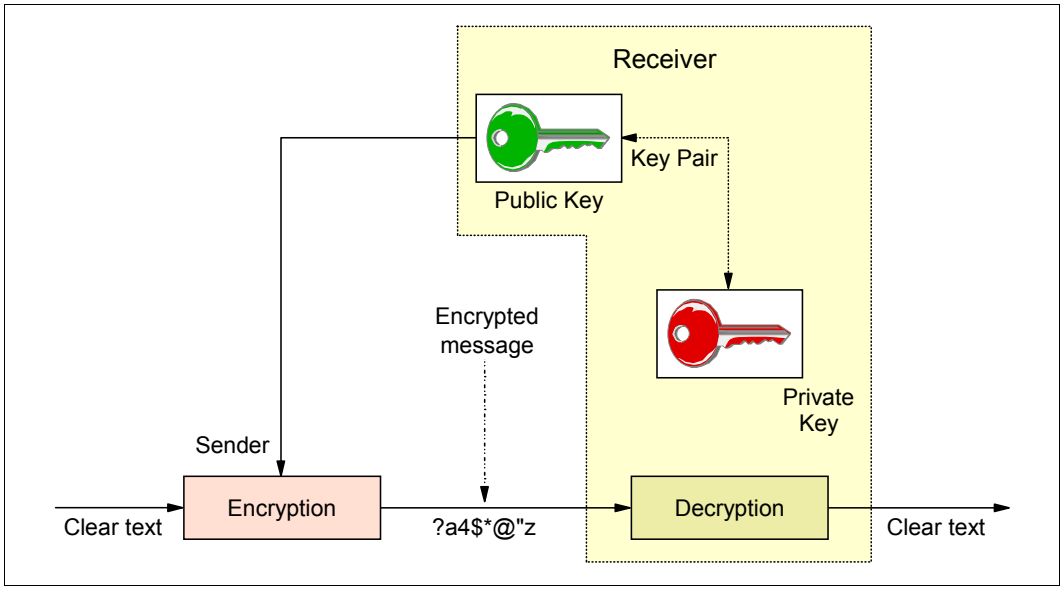


Figure A-14 Encrypting data with the receiver's public key

- 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. Thus, only the sender can encrypt the signature, but anybody 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 814.

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 scenario configured with native Transport Layer Security (TLS) function
- One scenario configured with Application Transparent Transport Layer Security (AT-TLS) policy

As discussed in Chapter 16, “Telnet security” on page 677, generally use AT-TLS instead of the native TLS to implement the secure connections.

We discuss the following topics in this appendix.

Section	Topic
“Advanced native TLS configuration” on page 824	Implementation and verification of the advanced TN3270 server configuration using the native TLS function
“Advanced AT-TLS configuration using client ID groups” on page 836	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 departments (or user groups) with various security requirements.

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-1 depicts the environment for this scenario.

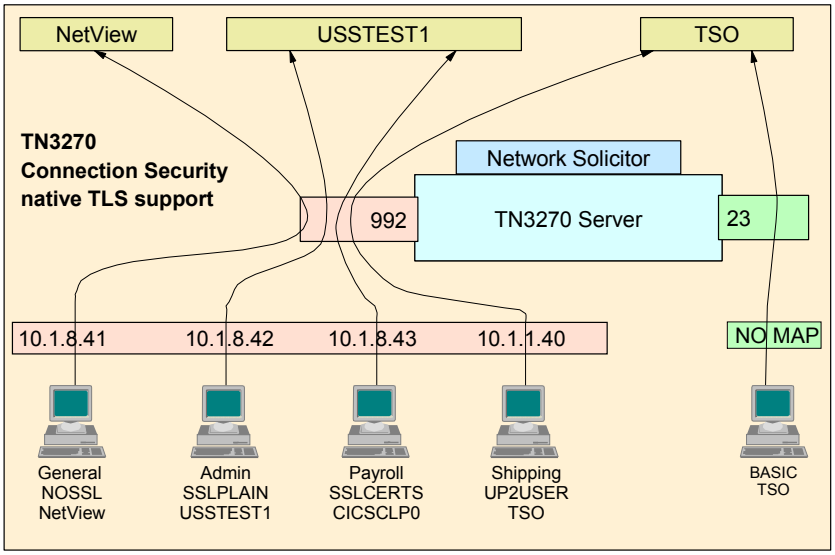


Figure B-1 Diagram of the advanced native TLS configuration

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
- Start the TN3270 server

Generate the key ring and certificates

Tip: In Chapter 3, “Certificate management in z/OS” on page 39, we include detailed examples and explanations for the steps that are involved in preparing the key ring and certificates that we use in this scenario. See that chapter for a complete discussion about using a shared key ring and a SITE certificate.

Example B-1 shows the RACF statements that are necessary to create the shared key ring, the certificate authority (CA) certificate, and the SITE certificate used by this scenario.

Example B-1 Sample RACF for key ring and certificates

```
racdcert certauth gencert                          -                    1
   subjectsdn( o('IBM Corporation')                 -
               ou('ITSO Certificate Authority')     -
                C('US'))                            -
    NOTBEFORE(DATE(2011-09-11))                     -
    NOTAFTER(DATE(2028-09-11))                      -
    keyusage(certsign)                              -
    withlabel('CS ITSO CA1')
  setropts raclist(facility) refresh
  racdcert certauth list

  racdcert site gencert subjectsdn(cn('ITSO.IBM.COM')   - 2
     o('IBM Corporation')                             -
     ou('ITSO CS Shared SITE')                       -
     C('US'))                                         -
     withlabel('CS ITSO SharedSite1')                -
     signwith(certauth label('CS ITSO CA1')
  racdcert site list

  racdcert ID(TCPIP) ADDRING(SharedRing1) 3

  racdcert ID(TCPIP) CONNECT(CERTAUTH                   - 4
                     LABEL('CS ITSO CA1')              -
                     RING(SharedRing1)                  -
                     USAGE(CERTAUTH)

  racdcert ID(TCPIP) CONNECT(SITE                       -
                     LABEL('CS ITSO SharedSite1')      -              5
                     RING(SharedRing1)                  -
                     DEFAULT                            -
                     USAGE(PERSONAL)

  setropts raclist(DIGTRING) refresh
  setropts raclist(DIGTCERT) refresh
  racdcert listring(*) id(TCPIP)
```

1. Creates a self-signed CA 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 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 window and be prompted for a user ID, password, and desired application. No SSL security will be implemented for that type of connection.

We use TN3270D configuration file for this test example.

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
```

In this example, the numbers correspond to the following information:

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 also be used, 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
```

In this example, the numbers correspond to the following information:

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 PROF to show profile SSL information
- Display CLIENTID to show client group SSL information
- Display OBJECT to show object SSL information

We used IBM Personal Communications V5.7 to establish the connection. Because we used self-signed certificate, we downloaded and installed the certificate of certificate authority into Personal Communications.

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 shown 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
```

In this example, the numbers correspond to the following information:

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 831 and Example B-8 on page 832.

Display CONN to show connection SSL information

Example B-5 and Example B-6 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 831 and Example B-8 on page 832. 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 Telnet CONN DETAIL for SSL information, before application selection

```
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 4
35 OF 35 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

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). See Table 16-1 on page 682 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 833.

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 on page 832.

Example B-7 Connection summary information for SSL port 992 after application selection

```
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 application selection

```
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: 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
    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 4
35 OF 35 RECORDS DISPLAYED
```

In this example, the numbers correspond to the following information:

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). See Table 16-1 on page 682 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 on page 833.

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 shown 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   CICSCLP0           --------
```

```
    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 details about the object. If you specify a TYPE that you know is related to a secure group you defined, such as DEFAPPL, PARMSGRP, or USS, you can see SSL-related information. 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.

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, see the verification steps for a Telnet stand-alone task described in *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997.

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 departments (or user groups) with various security requirements. Certain of the client ID grouping statements (PARMSGROUP, PARMSMAP) can be integrated 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 824. 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.

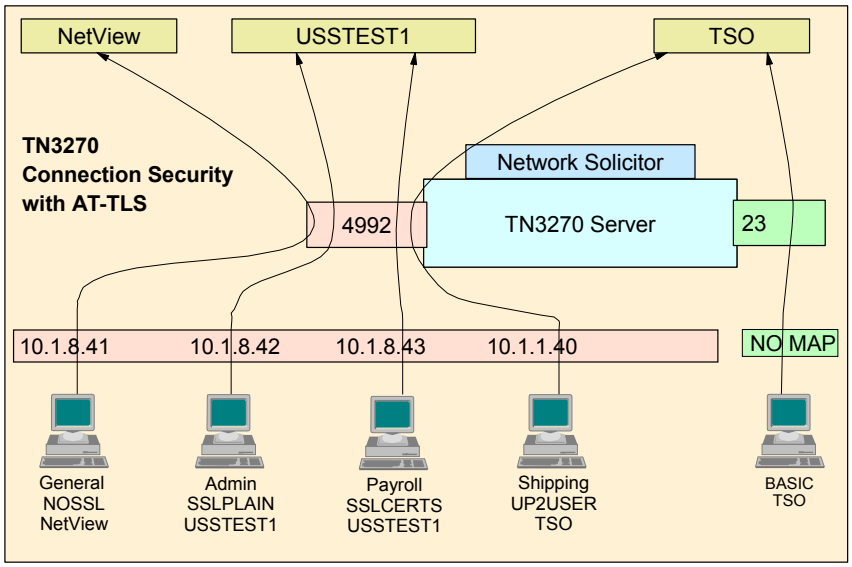


Figure B-2 TN3270 with advanced AT-TLS configuration diagram

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 TCP/IP profile
- Define AT-TLS policies
- Upload the policy to z/OS
- Modify the policy agent configuration file
- Define AT-TLS port in TN3270 configuration file

Set up the policy agent

We set up the policy agent as shown in 4.2, “Implementing PAGENT on z/OS” on page 109.

Create a certificate

We continued to use the certificate created for native TLS connection security as shown in “Advanced native TLS configuration” on page 824. 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 4.2, “Implementing PAGENT on z/OS” on page 109.

Modify TCP/IP profile

To enable AT-TLS to the TCP/IP stack, add the TCPCONFIG TTLS statement in the TCP/IP Profile as shown in Example B-16. 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 define AT-TLS connectivity rules for groups that have or might have secure connections, as shown in Table B-1. We do not need to define the policy for the basic (non-secure) connections.

Tip: 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 use the z/OSMF Configuration Assistant to define AT-TLS policies with the following steps:

1. Start the IBM Configuration Assistant for z/OS Communications Server. In the Main Perspective window, select the **Add a New z/OS image** option.
2. Type the name of the z/OS image (in our case, we use `SC33`) and click **OK**.
3. In the Main Perspective window, select the **Add New TCP/IP stack** option, enter the name of the TCP/IP stack (in our case, we use `TCPIPD`), and click **OK**.
4. In the Main Perspective window, select AT-TLS in the z/OS Communications Server technologies list. Click **Enable**, and then select the **Configure** option.
5. In the AT-TLS Perspective window, select **Reusable Objects** → **Traffic Descriptors** in the left pane.

6. Instead of using the predefined traffic descriptors, click **Add** to create a new traffic descriptor, name it `ATTLS_TN3270D_4992`, and then click **Add**, as shown on Figure B-3.

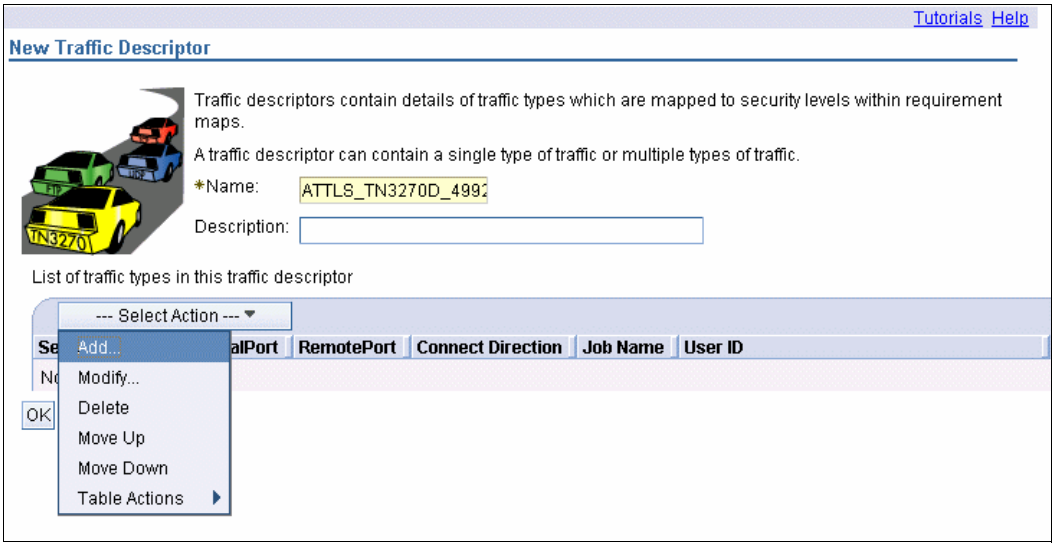


Figure B-3 New Traffic Descriptor window

7. In the New Traffic Type - TCP window, set the local port number to `4992` and the jobname to `TN3270D`. Then click the Advanced tab as shown in Figure B-4.

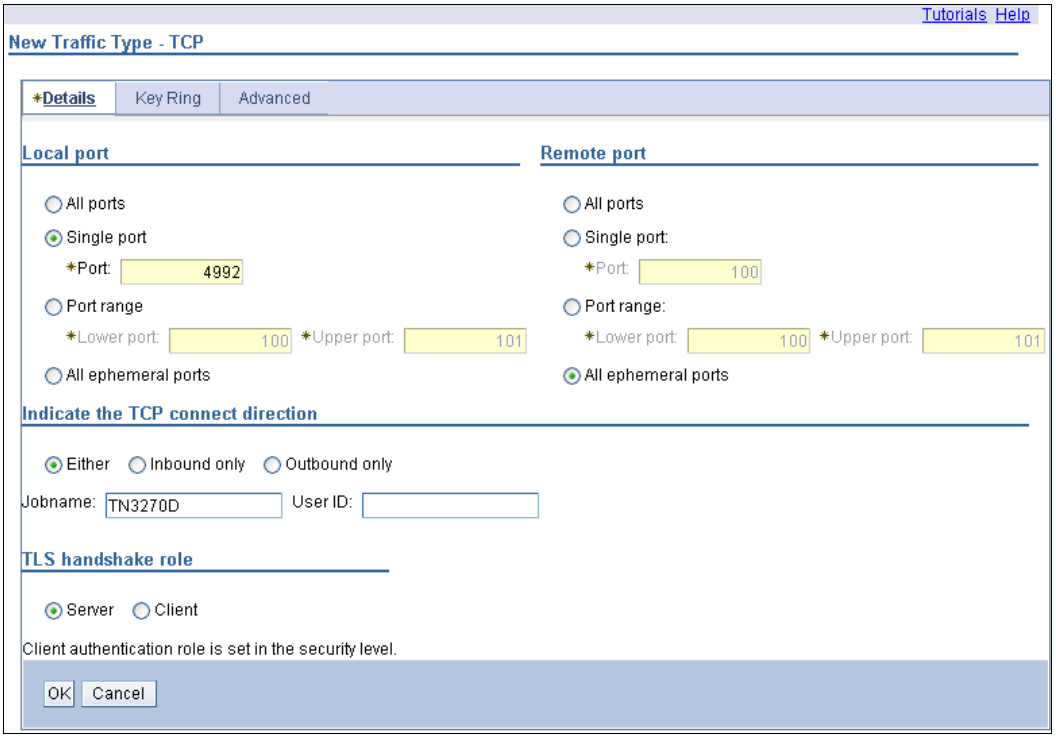


Figure B-4 New Traffic Type window

8. In the New Traffic Type - TCP, Advanced tab, toggle the **Application Controlled** option to **On**, and then click **OK**.

Tip: 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.

9. When back on the New Traffic Descriptor window, click **OK**. The traffic descriptor we've just created is now shown in the list in the New Traffic Descriptor window.
10. Select **Reusable Objects** → **Security Levels** on the left pane of the AT-TLS Perspective window. Create two new security levels:
    - No client authentication to be applied for ADMIN_SSLPLAIN and SHIPPING_UP2USER rule.
    - Requires client authentication to be applied to the PAYROLL_SSLCERT rule, which is equivalent to CLIENTAUTH SSLCERT in TN3270 profile for native TLS support.
11. Click **Add** to define a security level with no client authentication. In the New Security Level: Name and Type window, enter the name of the security level (we used Gold_NoClientAuth). Click **Next**.
12. In the New Security Level: Ciphers window, select the **Use Only Selected Ciphers** option and 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. Then click **Next**. The window configuration can be seen on Figure B-5 on page 841.

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

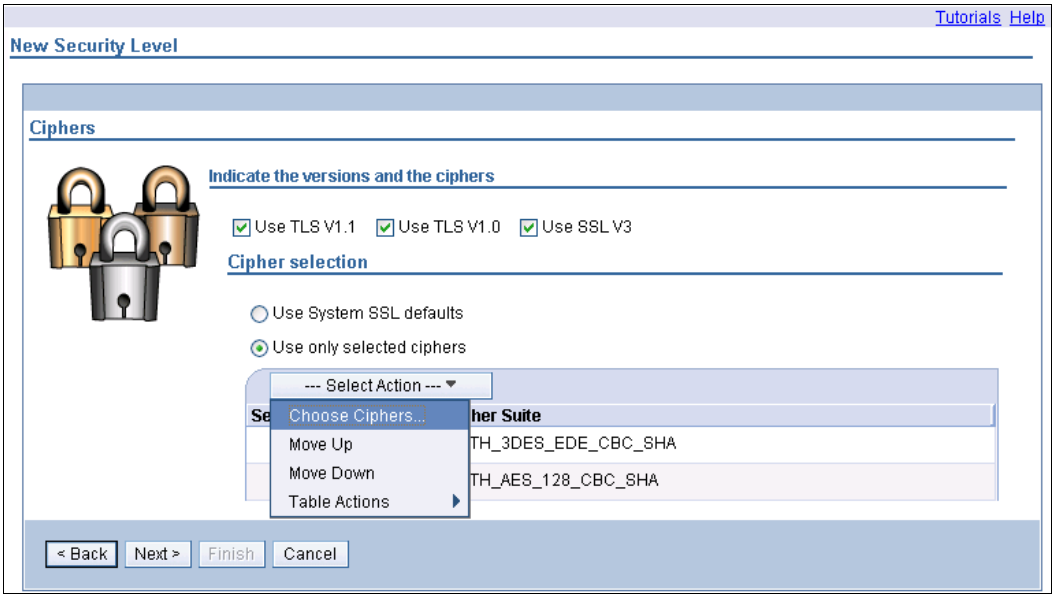


Figure B-5 New Security Level - Ciphers window

13. In the New Security Level - Advanced Settings window, click **Advanced Settings**. In the next window, click the **Client Authentication** tab. Make sure **No client authentication** is selected, click **OK** to return to the New Security Level - Advanced Settings window, and click **Finish**.

14. Define another security level with client authentication required by following the same instructions shown from step 11 to step 13. This time name the security level *Gold_ClientAuthSSLCert* (see step 11 on page 840) and select the **Use client authentication** option as **Required** in the Advanced AT-TLS Settings window (see step 13). The window configuration can be seen in Figure B-6.

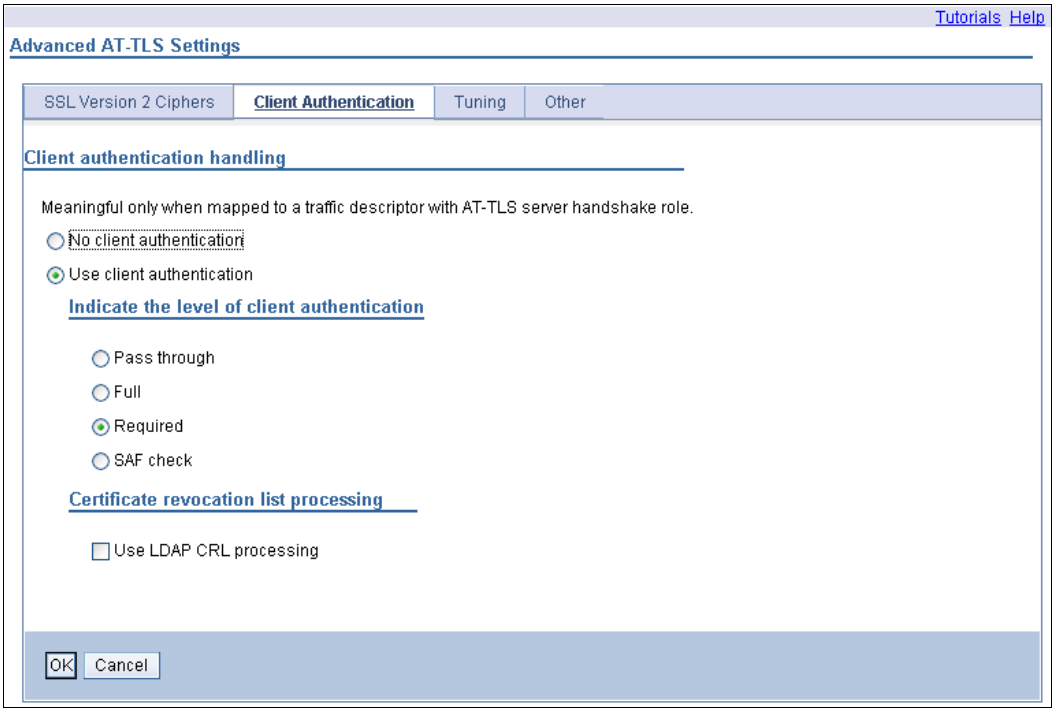


Figure B-6 Advanced AT-TLS Settings window

15. Back in the AT-TLS Perspective window, select **Reusable Objects** → **Requirement Maps** in the left pane. You will create two requirement maps that map the traffic descriptors and the security levels.
16. For the first requirement map, click the **Add** button, and type a name for the requirement map (we used *SSLPLAIN_ReqMap*). Then map the ATTLS_TN3270D_4992 traffic descriptor and the Gold_NoClientAuth security level together, and then click **OK**.
17. For the second requirement map, click the **Add** button, and type a name for the requirement map (we use *SSLCERT_ReqMap*). Then map the ATTLS_TN3270D_4992 traffic descriptor and the Gold_ClientAuthSSLCert security level together, and then click **OK**.
18. Next, put all the defined objects together into the connectivity rules. In the AT-TLS Perspective window, select the TCP/IP stack name (*TCPIPD*, in our case) in the left pane.
19. For the first connectivity rule, click **Add**. In the New Connectivity Rule: Data Endpoints window, enter the name of the rule (we use *ADMIN_SSLPLAIN*). Then define *10.1.8.42* on Local data endpoint, and select **All_IPv4_Addresses** for the Remote data endpoint. The configuration window is shown in Figure B-7. After you have completed configuration, click **Next**.

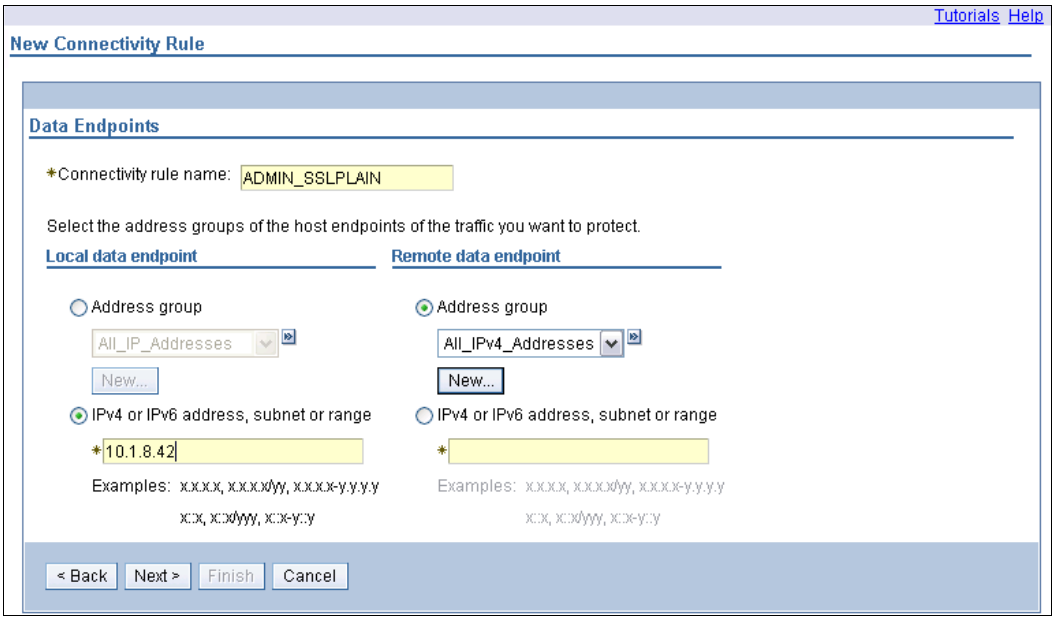


Figure B-7 New Connectivity Rule - Data Endpoints window

20. In the New Connectivity Rule - Select Requirement Map window, click **Select an existing requirement map**, select what you have defined previously (in our case it is SSLPLAIN_ReqMap), and click **Next**. In the next window, click **Finish**.
21. For the second connectivity rule, use the same procedure as described on step 19 through step 20, with the following parameter differences:
    - Connectivity rule name: PAYROLL_SSLCERT
    - Local data endpoint: 10.1.8.43
    - Remote data endpoint: All_IPv4_Addresses
    - Requirement map: SSLCERT_ReqMap

22. For the third connectivity rule, again use the same procedure as described on step 19 through step 20, with the following parameter differences:
    - Connectivity rule name: SHIPPING_UP2USER
    - Local data endpoint: 10.1.1.40
    - Remote data endpoint: All_IPv4_Addresses
    - Requirement map: SSLPLAIN_ReqMap
23. The AT-TLS Perspective window displays the defined connectivity rule, but it is still Incomplete because the Key Ring is not defined yet. Click the **Apply Changes** option to save the configuration.
24. In the left pane, select 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 837.

    We specified `TCPIP/SharedRing1` in the Key ring is in SAF product field for our example, and clicked **OK**.
25. The policy definition was now complete. The connectivity rules defined in the AT-TLS Perspective window are displayed in Figure B-8. The configuration file generated by the IBM Configuration Assistant for z/OS Communications Server is shown in Example B-17 on page 844.

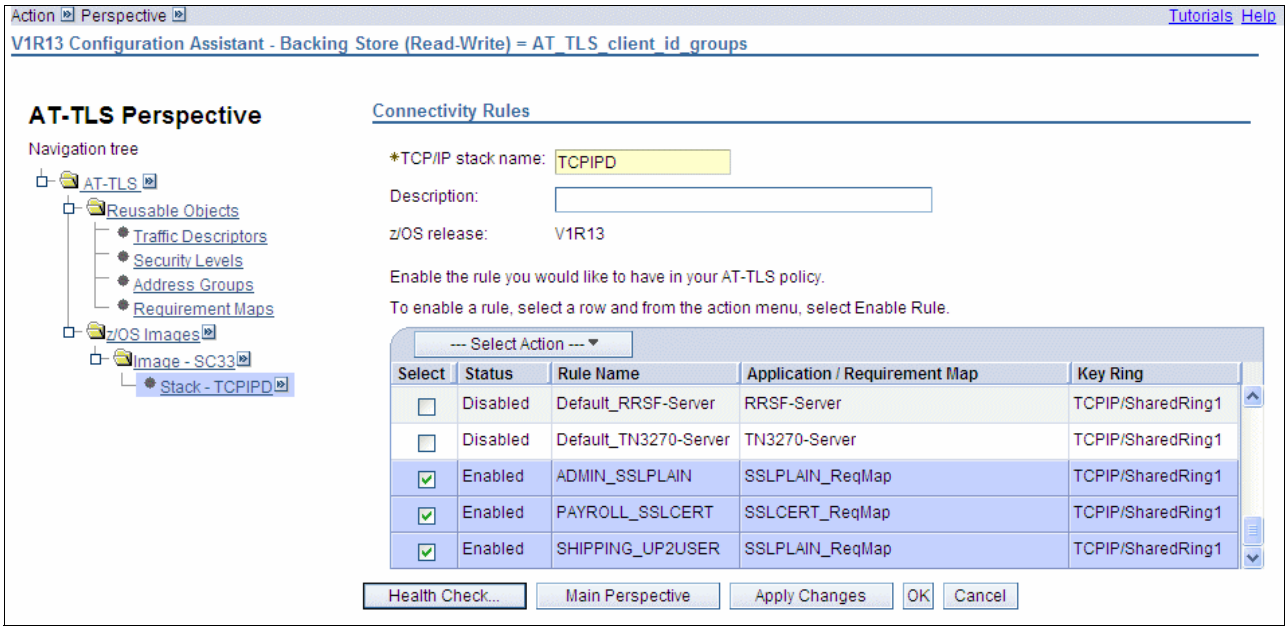


Figure B-8 AT-TLS Perspective window with newly defined connectivity rules

Example B-17 AT-TLS policy configuration file

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: SC33
##    Stack: TCPIPD
##
.............................................................. Lines deleted
## End of IBM Configuration Assistant information


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~Gold_NoClientAuth
  TTLSConnectionAdvancedParmsRef  cAdv1~ATTLS_TN3270D_4992
  CtraceClearText                 Off
  Trace                           2
}
TTLSConnectionAction              cAct2~ATTLS_TN3270D_4992
{
  HandshakeRole                   ServerWithClientAuth
  TTLSCipherParmsRef              cipher2~Gold_ClientAuthSSLCert
  TTLSConnectionAdvancedParmsRef  cAdv1~ATTLS_TN3270D_4992
  CtraceClearText                 Off
  Trace                           2
}
TTLSConnectionAdvancedParms       cAdv1~ATTLS_TN3270D_4992
{
  ApplicationControlled           On
  SecondaryMap                    Off
}
TTLSKeyringParms                  keyR~SC33
{
  Keyring                         TCPIP/SharedRing1
}
TTLSCipherParms                   cipher1~Gold_NoClientAuth
{
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
TTLSCipherParms                   cipher2~Gold_ClientAuthSSLCert
{
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_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, complete the following steps:

1. In the IBM Configuration Assistant Navigation Tree pane, AT-TLS perspective, click the small arrow in front of the Stack-TCPIPD, and click **Install Configuration Files.** Select the radio button for the TCPIPD policy, click **Select Action** → **Show Configuration File** to view the policy file created. Click **Close.** Click **Select Action** → **Install**.
2. Click **Save to disk** and specify the installation file name:

   `/etc/cfgasst/v1r13/SC33/TCPIPD/tlsPol_client_id_groups`
3. Click **Go.** Note the message: `The save to the file system was successful`

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/pagent.sc33.conf
#  ******************************************************************
  TcpImage TCPIPD /etc/pagent.sc33.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/pagent.sc33.tcpipd.conf
#  ******************************************************************
TTLSConfig TTLSConfig /etc/cfgasst/v1r13/SC33/TCPIPD/tlsPol_client_id_groups
```

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 4 ; 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 5
     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
  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 4
; DEFAULTAPPL CICSCLPO  DESTIPGRP,PAYROLL  ; omit - defined in AT-TLS policy

  PARMSMAP UP2USER         DESTIPGRP,SHIPPING 5
  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
```

In this example, the numbers correspond to the following information:

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 and PAYROLL are omitted because they are defined in the policy agent.
5. The PARMSGROUP and PARMSMAP statements for SHIPPING are specified to override the CONNTYPE to ANY.

Activation and verification

Complete 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 at 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
```

In this example, the numbers correspond to the following information:

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 as shown in Example B-25.

Example B-25 pasearch -t display

```
TCP/IP pasearch CS Image Name: TCPIPD
TTLS Instance Id:     1227199614

policyRule:             sc33_to_world~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
   ActionType:          TTLS Group
  ActionType:          TTLS Group
  Action Sequence:     0
 policyAction:         eAct1
```

```
  ActionType:          TTLS Environment
  Action Sequence:     0
..................................................................... Lines
deleted
 TTLS Action:                  gAct1
   Version:                    3
   Status:                     Active
   Scope:                      Group
   TTLSEnabled:                On
   CtraceClearText:            Off
    CtraceClearText:            Off
    Trace:                      2
    TTLSGroupAdvancedParms:
     SecondaryMap:              Off
     SyslogFacility:            Daemon
    Policy created: Thu Nov 20 11:46:54 2008
    Policy updated: Thu Nov 20 11:46:54 2008
..................................................................... Lines
deleted
  TTLS Action:                  eAct1
    Version:                    3
    Status:                     Active
    Scope:                      Environment
    HandshakeRole:              Server
    TTLSKeyringParms:
     Keyring:                   TCPIP/SharedRing1
    TTLSEnvironmentAdvancedParms:
     SSLv2:                     Off
     SSLv3:                     On
     TLSv1:                     On
     ApplicationControlled:     Off
    ApplicationControlled:     Off
    HandshakeTimeout:          10
    ClientAuthType:            Required
    ResetCipherTimer:          0
   EnvironmentUserInstance:    0
   Policy created: Thu Nov 20 11:46:54 2008
  Policy updated: Thu Nov 20 11:46:54 2008
..................................................................... Lines
deleted
policyRule:             ADMIN_SSLPLAIN~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
   ActionType:          TTLS Group
   Action Sequence:     0
  policyAction:         eAct1
   ActionType:          TTLS Environment
   Action Sequence:     0
  policyAction:         cAct2~ATTLS_TN3270D_4992
   ActionType:          TTLS Connection
   Action Sequence:     0
```

```
..................................................................... Lines
deleted
  TTLS Condition Summary:                NegativeIndicator: Off
   Local Address:
    FromAddr:           10.1.8.42
    ToAddr:             10.1.8.42
   Remote Address:
    FromAddr:           All
    ToAddr:             All
   LocalPortFrom:       4992              LocalPortTo:       4992
   RemotePortFrom:      1024              RemotePortTo:      65535
  RemotePortFrom:      1024              RemotePortTo:      65535
  JobName:             TN3270D           UserId:
  ServiceDirection:    Both
 Policy created: Thu Nov 20 11:46:54 2008
 Policy updated: Thu Nov 20 11:46:54 2008
..................................................................... Lines
deleted
policyRule:            PAYROLL_SSLCERT~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
   ActionType:         TTLS Group
   Action Sequence:    0
  policyAction:        eAct2~ATTLS_TN3270D_4992
   ActionType:         TTLS Environment
   Action Sequence:    0
  policyAction:        cAct3~ATTLS_TN3270D_4992
   ActionType:         TTLS Connection
   Action Sequence:    0
..................................................................... Lines
deleted
  TTLS Condition Summary:                NegativeIndicator: Off
   Local Address:
    FromAddr:           10.1.8.43
    ToAddr:             10.1.8.43
   Remote Address:
    FromAddr:           All
    ToAddr:             All
   LocalPortFrom:       4992              LocalPortTo:       4992
   RemotePortFrom:      1024              RemotePortTo:      65535
   JobName:             TN3270D           UserId:
   ServiceDirection:    Both
  Policy created: Thu Nov 20 11:46:54 2008
  Policy updated: Thu Nov 20 11:46:54 2008
 Policy updated: Thu Nov 20 11:46:54 2008
..................................................................... Lines
deleted
TTLS Action:                  eAct2~ATTLS_TN3270D_4992
   Version:                    3
   Status:                     Active
   Scope:                      Environment
```

```
   HandshakeRole:              ServerWithClientAuth
   HandshakeRole:              ServerWithClientAuth
   TTLSKeyringParms:
    Keyring:                   TCPIP/SharedRing1
   TTLSEnvironmentAdvancedParms:
    SSLv2:                     Off
    SSLv3:                     On
    TLSv1:                     On
    ApplicationControlled:     Off
    HandshakeTimeout:          10
    ClientAuthType:            Required
    ResetCipherTimer:          0
   EnvironmentUserInstance:    0
   Policy created: Thu Nov 20 11:46:54 2008
   Policy updated: Thu Nov 20 11:46:54 2008

 TTLS Action:                  cAct3~ATTLS_TN3270D_4992
   Version:                    3
   Status:                     Active
   Scope:                      Connection
   HandshakeRole:              ServerWithClientAuth
    HandshakeRole:              ServerWithClientAuth
    CtraceClearText:            Off
    Trace:                      2
    TTLSConnectionAdvancedParms:
     SecondaryMap:              Off
     ApplicationControlled:     On
     HandshakeTimeout:          10
     ResetCipherTimer:          0
    TTLSCipherParms:
     v3CipherSuites:
      0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
      2F  TLS_RSA_WITH_AES_128_CBC_SHA
    Policy created: Thu Nov 20 11:46:54 2008
    Policy updated: Thu Nov 20 11:46:54 2008

policyRule:              SHIPPING_UP2USER~4
  Rule Type:             TTLS
  Version:               3                 Status:            Active
  Weight:                252               ForLoadDist:       False
  Priority:              252               Sequence Actions:  Don't Care
  Priority:              252               Sequence Actions:  Don't Care
  No. Policy Action:     3
  policyAction:          gAct1
   ActionType:           TTLS Group
   Action Sequence:      0
  policyAction:          eAct1
   ActionType:           TTLS Environment
   Action Sequence:      0
  policyAction:          cAct2~ATTLS_TN3270D_4992
   ActionType:           TTLS Connection
  Action Sequence:    0
..................................................................... Lines
deleted
   Local Address:
```

```
    FromAddr:           10.1.1.40
    ToAddr:             10.1.1.40
   Remote Address:
    FromAddr:           All
    ToAddr:             All
   LocalPortFrom:       4992              LocalPortTo:       4992
   RemotePortFrom:      1024              RemotePortTo:      65535
   JobName:             TN3270D           UserId:
   ServiceDirection:    Both
  Policy created: Thu Nov 20 11:46:54 2008
  Policy updated: Thu Nov 20 11:46:54 2008
..................................................................... Lines
deleted
 TTLS Action:                  cAct2~ATTLS_TN3270D_4992
   Version:                    3
   Status:                     Active
   Scope:                      Connection
   HandshakeRole:              Server
   CtraceClearText:            Off
   Trace:                      2
   TTLSConnectionAdvancedParms:
    SecondaryMap:              Off
    SecondaryMap:              Off
    ApplicationControlled:     On
    HandshakeTimeout:          10
    ResetCipherTimer:          0
   TTLSCipherParms:
    v3CipherSuites:
     0A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
     2F  TLS_RSA_WITH_AES_128_CBC_SHA
   Policy created: Thu Nov 20 11:46:54 2008
   Policy updated: Thu Nov 20 11:46:54 2008
```

Display CONN to show connection AT-TLS information

We used IBM Personal Communications V5.7 to establish a secure session. Because we used self-signed certificate, we downloaded and installed the certificate of the certificate authority into Personal Communications. We defined a Personal Communications 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
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
```

In this example, the numbers correspond to the following information:

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 on page 856 shows a secure connection with a client requesting TLS that is destined to IP address 10.1.1.40.

Example B-27 TLS secure connection using port 4992, 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
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
```

In this example, the numbers correspond to the following information:

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 that is destined to IP address 10.1.1.40.

Example B-28 TLS non-secure connection using port 4992, 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
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
```

In this example, the numbers correspond to the following information:

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 (not applicable in this case).

C

Configuring IPSec between z/OS and Windows

This appendix contains the steps used for our IPSec scenarios between z/OS and Windows.

We discuss the following topics in this appendix.

Section	Topic
“IPSec between z/OS and Windows: Pre-shared Key Mode” on page 860	Setting up the IKE daemon, the z/OS IPSec policy, and a Windows IPSec policy. Verifying that things are working.
“IPSec between z/OS and Windows: RSA mode” on page 882	Setting up the IKE daemon, the certificates for x.509 RSA signature mode, the z/OS IPSec policy, and a Windows IPSec policy.
“Set up a Windows IPSec policy for RSA mode” on page 889	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.

For more details about our IPSec scenarios between z/OS systems, see 8.6, “Configuring IPSec between two z/OS systems: Pre-shared Key Mode using IKEv2” on page 285 and to 8.7, “Configuring IPSec between two z/OS systems: RSA signature mode using IKEv1” on page 306.

IPSec between z/OS and Windows: Pre-shared Key Mode

Figure C-1 shows the setup that we implemented in this scenario.

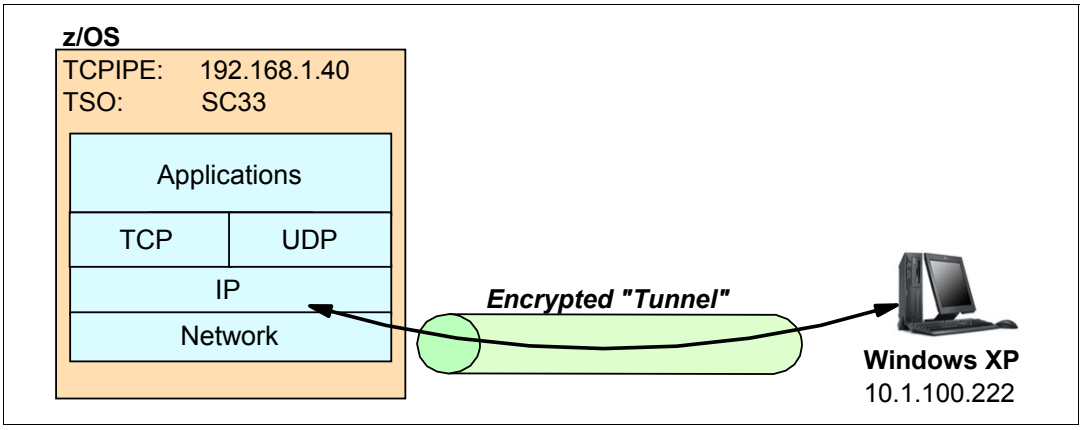


Figure C-1 VPN between z/OS and Windows

The following steps are used to configure a VPN between z/OS image and Windows XP using Dynamic Tunnels with pre-shared keys:

1. Set up the IKE daemon
2. Set up the z/OS IPSec policy
3. Set up a Windows IPSec policy for pre-shared key mode
4. 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 8.5.8, “Setting up the IKE daemon” on page 274.

Because we built our test environment using a Windows XP station, our IPSec tunnel will be implemented with IKEv1.

Tip: Internet Key Exchange version 2 (IKEv2) support was added in Windows Server 2008 R2 and Windows 7. Windows XP does not support it by itself.

Set up the z/OS IPSec policy

We used the z/OSMF Configuration Assistant to create our IPSec policy file for SC33 z/OS image.

To configure the test connection, we completed the following steps:

1. We create requirement map objects called `All_Traffic_PFS and Basic_Services`.
2. For the TCPIPE stack of the z/OS image SC33, we assign the requirement maps to the connectivity rules: `All_Traffic_PFS_Rule` and `Basic_Services_Rule`. We created these rules for the following purposes:
   - 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).
   - Basic_Services: A rule that permits Resolver, DNS, Ping, and OMPROUTE-IP_V4 traffic between all IPv4 addresses to all IPv4 addresses.
3. We verified that the IKE daemon settings are correct.
4. We updated the IPSec policy of the z/OS image.

For the All_Traffic_PFS connectivity rule, we choose the security level that we created in “Creating a security level for PFS” on page 269.

Creating requirement map objects

To create requirement map objects using IBM Configuration Assistant, follow these steps:

1. Start the IBM Configuration Assistant. In the Main Perspective panel, select **IPSec** technology then click **Select Action** and click **Configure**. In the IPSec Perspective panel, click **Requirement Maps**, then click **Select Action** and click **Add**.
2. In the New 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 **All_other_traffic** button, then selecting **PFS** in the Security Level drop-down menu as shown in Figure C-2 on page 862. Click **OK**.

> **Tip:** The PFS Security level we used in this scenario is the same one created in “Creating a security level for PFS” on page 269 in this book.

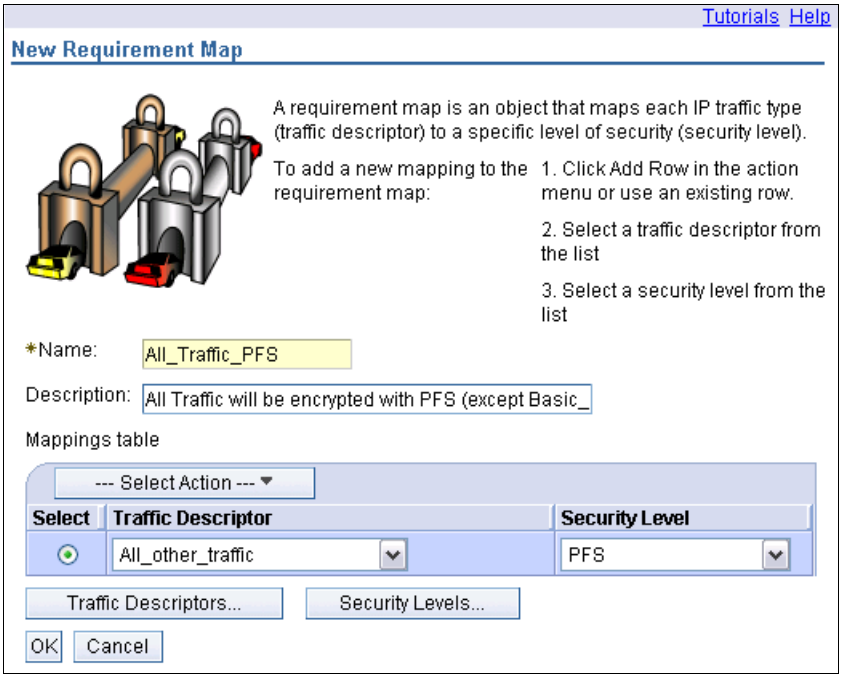


Figure C-2 New Requirement Map panel: Adding All_Traffic_PFS requirement map object

3. In the IPSec Perspective panel, click **Requirement Maps**, then click **Select Action** → **Add**.
4. In the New Requirement Map panel, complete the following steps:
   a. Enter a name and description for the object (we used `Basic_Services`).
   b. In the Mappings table, traffic descriptor list, click the radio button next to “Select a traffic descriptor” and from the drop down menu select **Ping-IP_V4** and then set the security level to **Permit**.
   c. Repeat the same procedure for the following traffic descriptor objects: Resolver, OMPROUTE-IP_V4, and DNS. To add extra rows to the Mappings table click **Select Action** then **Add Row**. The final configuration of this panel should looks like the configuration shown in Figure C-3 on page 863.
   d. Click **OK**.

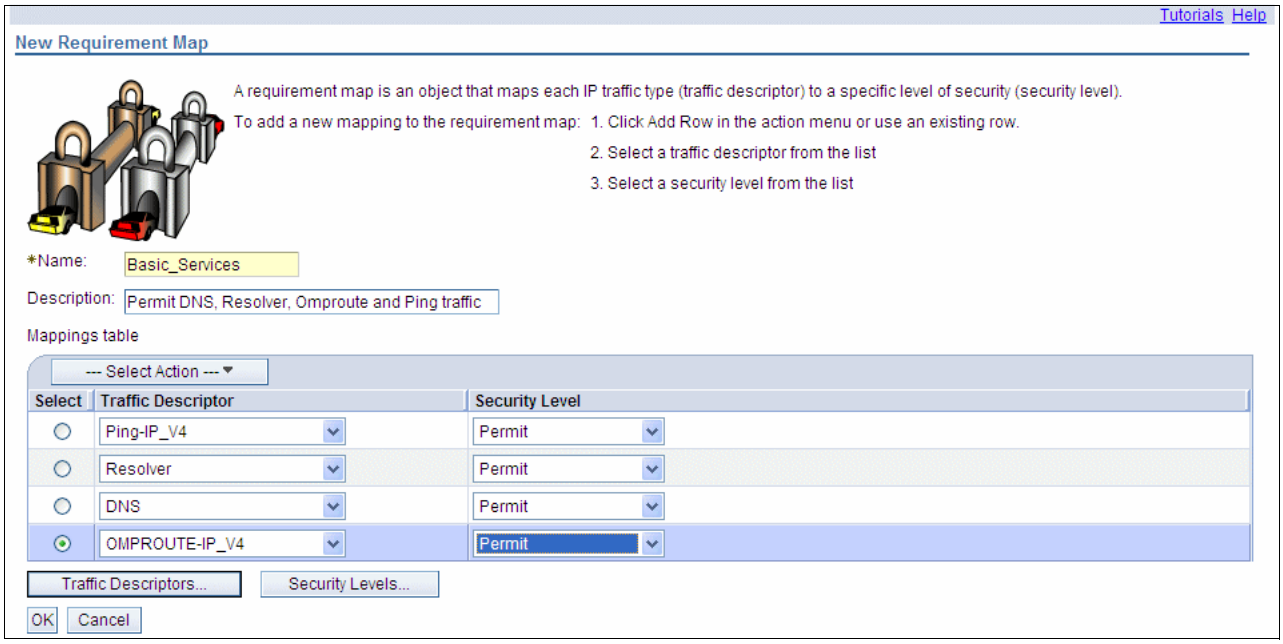


Figure C-3 Requirement Map panel: Adding Basic_Services object

The Requirement Maps panel should now include the new objects, as shown in Figure C-4.

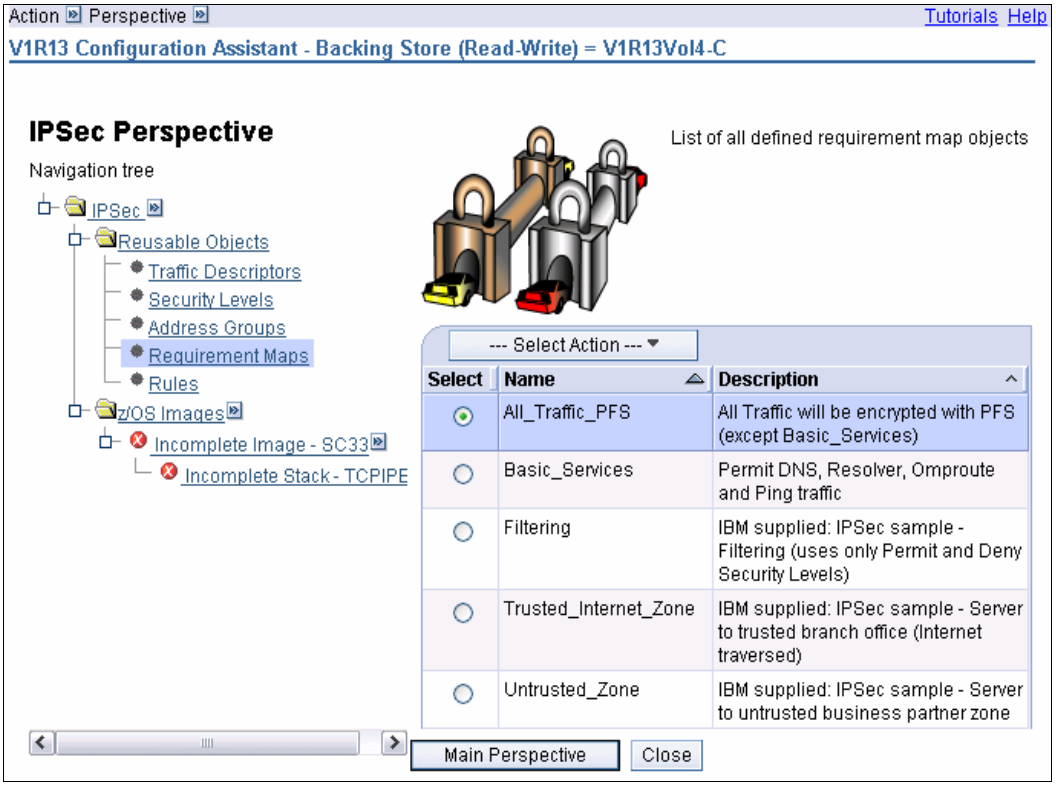


Figure C-4 Requirement Map panel: including All_Traffic_PFS and Basic_Services objects

Creating the connectivity rules

To create the connectivity rules by using IBM Configuration Assistant, complete the following steps:

1. Navigate by using the navigation tree as follows: **IPSec** → **z/OS Images** → **Image - *image_name*** → **Stack - *stack_name***.
2. After you click the stack name (ours is **TCPIPE**), add connectivity rules by clicking **Select Action** → **Add** (or click **Yes** if a pop-up window opens). The connectivity rule wizard - Welcome panel displays. Click **Next**.
3. In the Connectivity Rule: Network Topology panel, select **Filtering only**. This connectivity rule will contain only Permit and Deny security levels, and click **Next**.
4. In the Connectivity Rule: Data Endpoints panel, enter a name to the rule (in our configuration, we entered `Basic_Services`, as shown in Figure C-5). Select both **Address group** buttons. For the address group on the left, select **All IP V4 addresses** as the local data endpoint; for the address group on the right, select **All IP V4 addresses** as the remote data endpoint. Click **Next**.

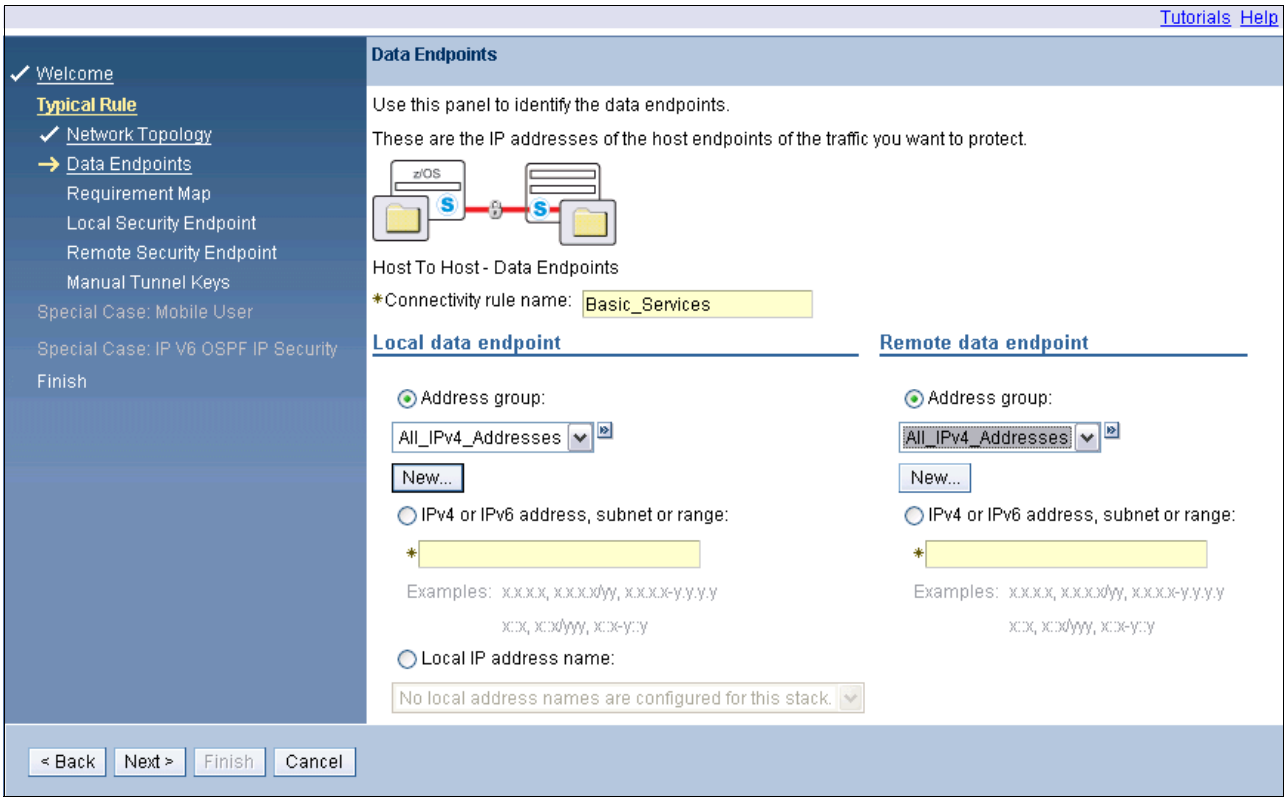


Figure C-5 New Connectivity Rule - Data Endpoints

5. In the Connectivity Rule: Requirement Map panel, click **Select an existing requirement map** radio button, select **Basic_Services** from the drop-down menu, and click **Next**.
6. In the Connectivity Rule: Finish panel, select **No, do not log filter matches**. Click **Finish**.

Follow these steps to add our next connectivity rule, All_Traffic_PFS:

1. Click **Select Action** and **Add** in Rules tab of the IPSec perspective panel. Click **Next** in the Connectivity Rules wizard - Welcome panel.
2. In the Connectivity Rule: Network Topology panel, select **This connectivity rule will contain a security level using IPSec tunnels**, and then select **Host to Host**. Click **Next**.

3. In the Connectivity Rule: Data Endpoints panel, enter a name for the rule. In our configuration, we entered `All_Traffic_PFS` as shown in Figure C-6. Then 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`). Click **Next**.

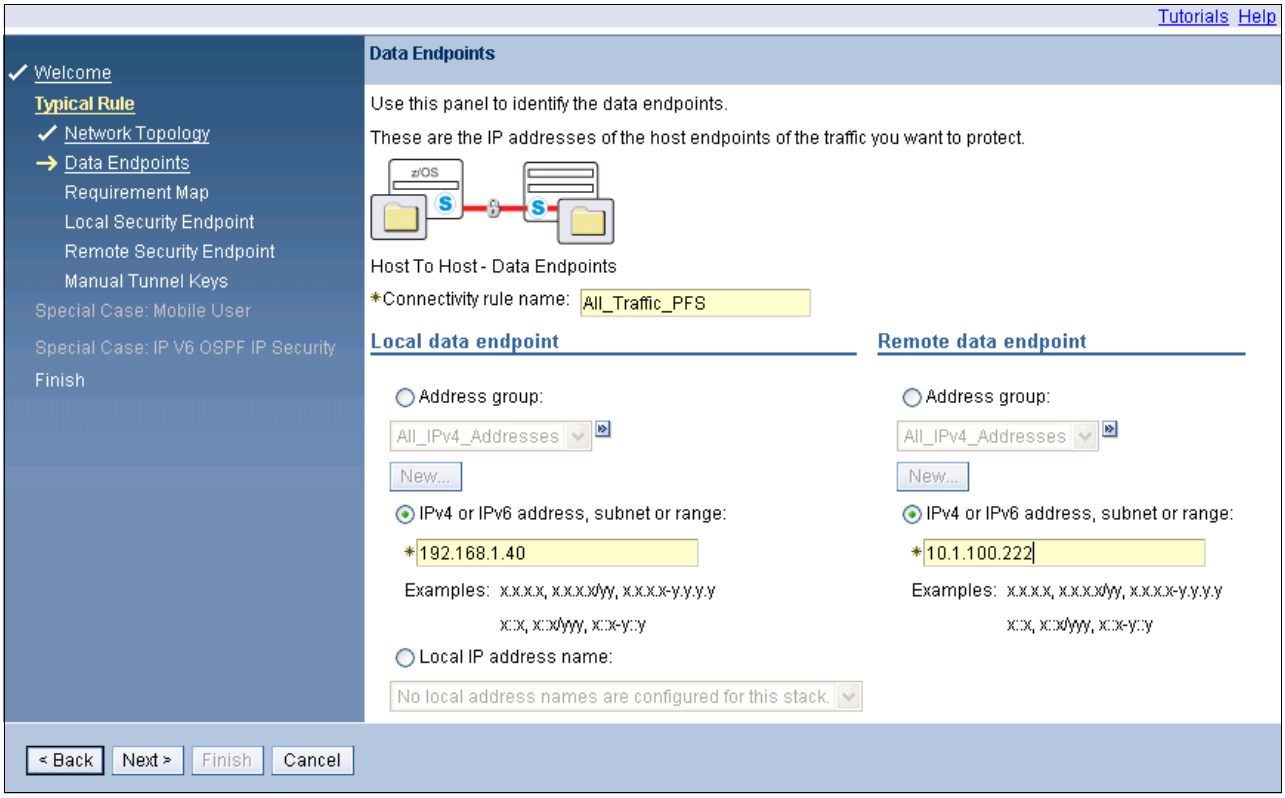


Figure C-6 Connectivity Rule - Data Endpoints

4. In the Connectivity Rule: Requirement Map panel, click **Select an existing requirement map** radio button, select **All_Traffic_PFS** from the drop-down menu, and click **Next**.
5. In the Connectivity Rule: Local Security Endpoint panel, click **Next** to accept the default value of using the local IP address as the local IKE identity.

6. In the Connectivity Rule: Remote Security Endpoint 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 (10.1.100.222). 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-7. Click **Next.**

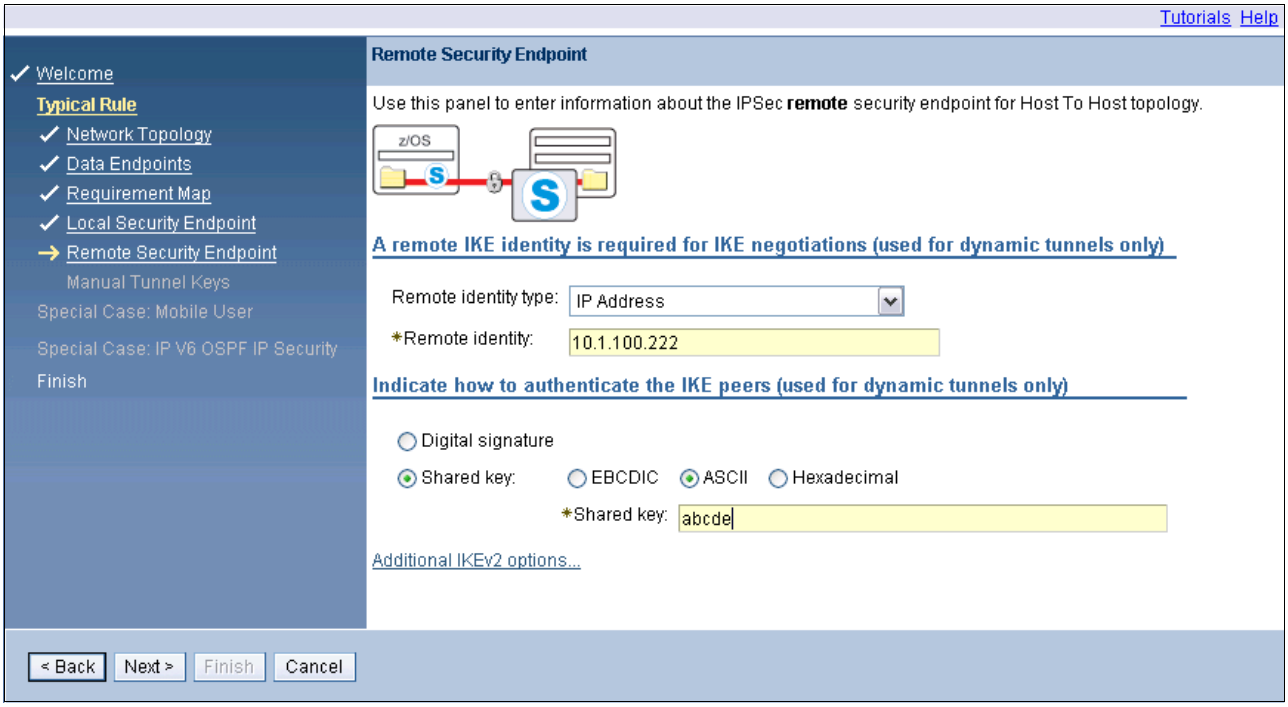


Figure C-7 Connectivity Rule - Remote Security Endpoint

7. In the Connectivity Rule: Finish panel, select **Yes, log all filter matches**. Click **Finish**.

The IPSec Perspective panel should now include the new rules in the Connectivity Rules table as shown in Figure C-8.

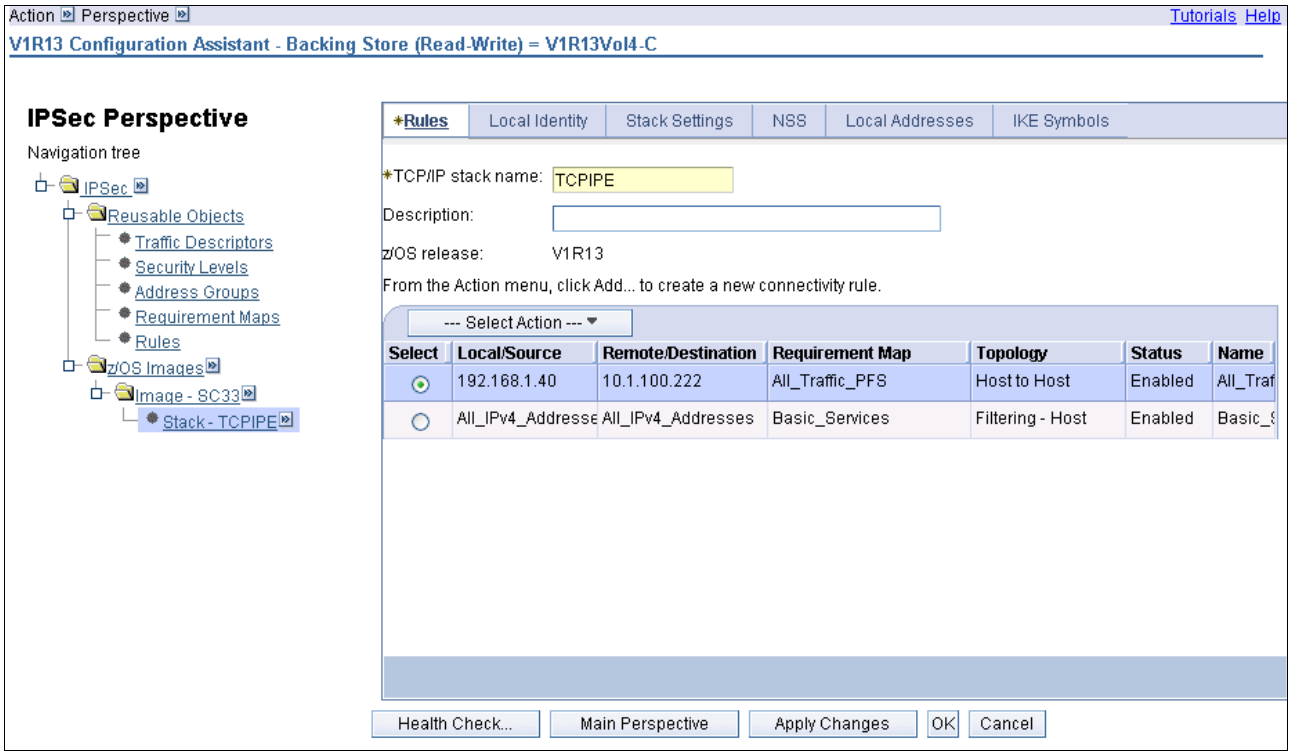


Figure C-8 IPSec perspective panel with the All_Traffic_PFS rule included

Tip: Verify that the rules are in the correct order; use **Select Action** and then **Move Up** if necessary to change the order of the rules. Packets are checked against the connectivity rules in the order that they are listed in the table. If there is a match with the definition of this first rule, it is used. If there is no match, the second rule is checked, and so on.

For performance, order the rules to ensure that the majority of the traffic will match the first rules listed, if possible.

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.

Updating the IPSec policy of the z/OS image

Install the new policy to the z/OS image as explained in 8.6.2, “Installing the configuration files” on page 297.

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 using pre-shared key mode. The next step is to ensure that an equivalent setup has been handled on the Windows XP workstation.

Set up a Windows IPSec policy for pre-shared key mode

In this section, we describe how to set up the Microsoft Management Console (MMC). Using MMC we will add one snap-in: IP Security Management (see Figure C-26 on page 890). 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. Type `mmc` in the Open field. Click **OK** to start the Microsoft Management Console.
3. In the Console 1 window, click **File** from the menu bar. From the 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 **IP Security Policy Management** and click **Add**.
6. In the Select Computer window, select **Local computer** and click **Finish**.
7. In the Add Standalone Snap-in window, click **Close**.
8. In the Add/Remove Snap-in window, verify that one snap-in has been added: IP Security Policies on Local Machine. Click **OK**.

This process completes the required settings for the MMC when implementing IPSec with pre-shared key mode.

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 menu, click **Create IP Security Policy** as shown in Figure C-9.

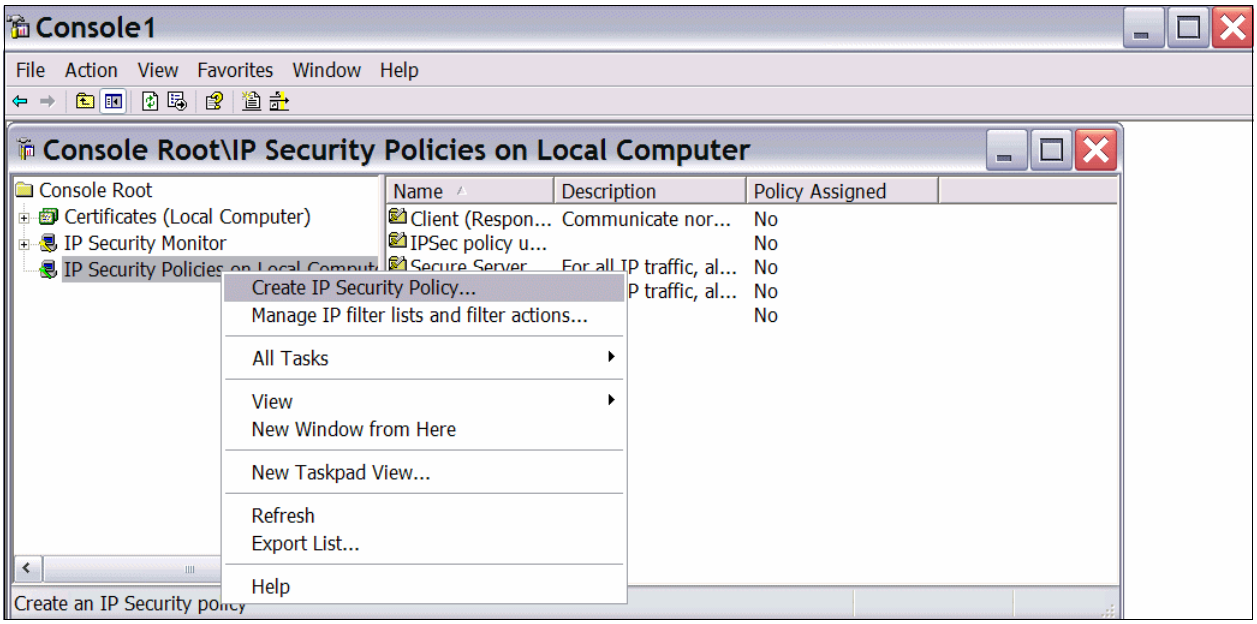


Figure C-9 MMC: IP Security Policies on Local Computer

2. In the IP Security Policy Wizard - Welcome to the IP Security Policy Wizard window, 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 check mark from the **Activate the default response rule** check box. Click **Next**.
5. In the IP Security Policy Wizard - Completing the IP Security Policy Wizard window, 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-10, and click **Add**.

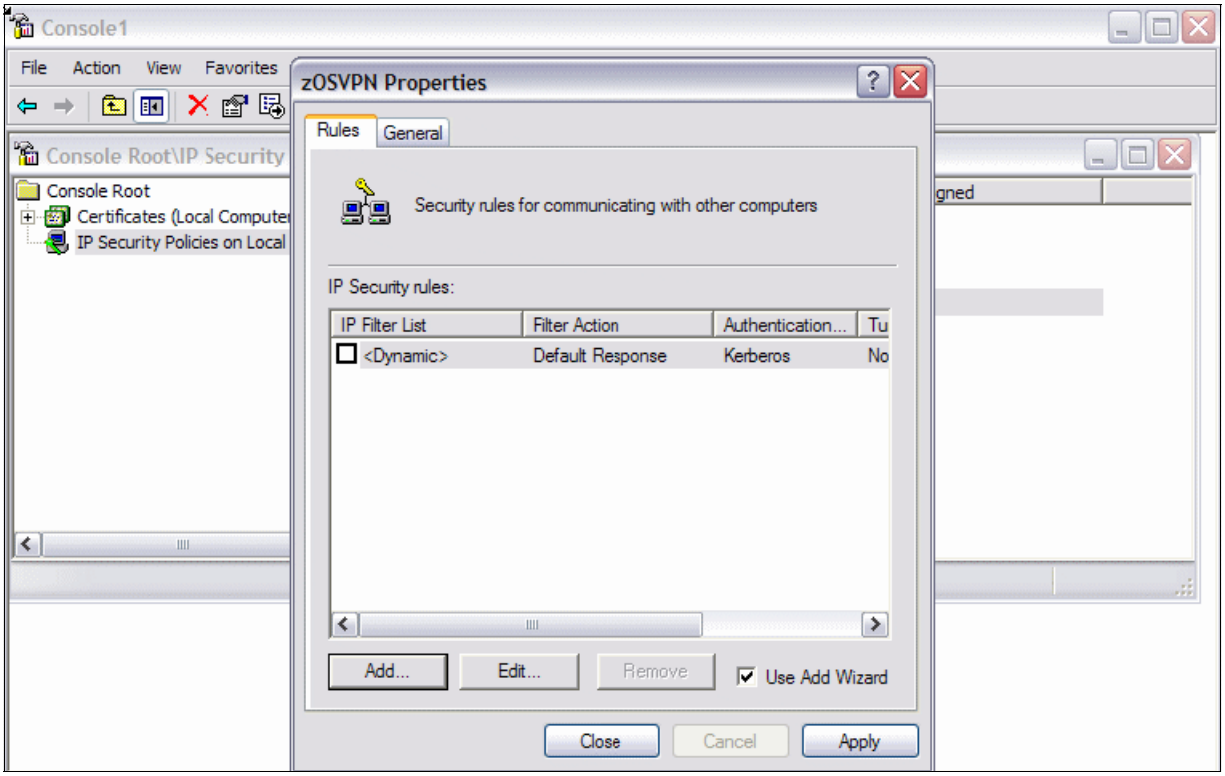


Figure C-10 IP policy properties window

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.

Tip: 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)** and enter your shared key as shown in Figure C-11. In this example, our shared key is `abcde`. Click **Next**.

Tip: Generally, 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, see *z/OS Communications Server: IP Configuration Guide*, SC31-8775. Also see the sections on RSA signature mode in Chapter 8, “IP Security” on page 245.

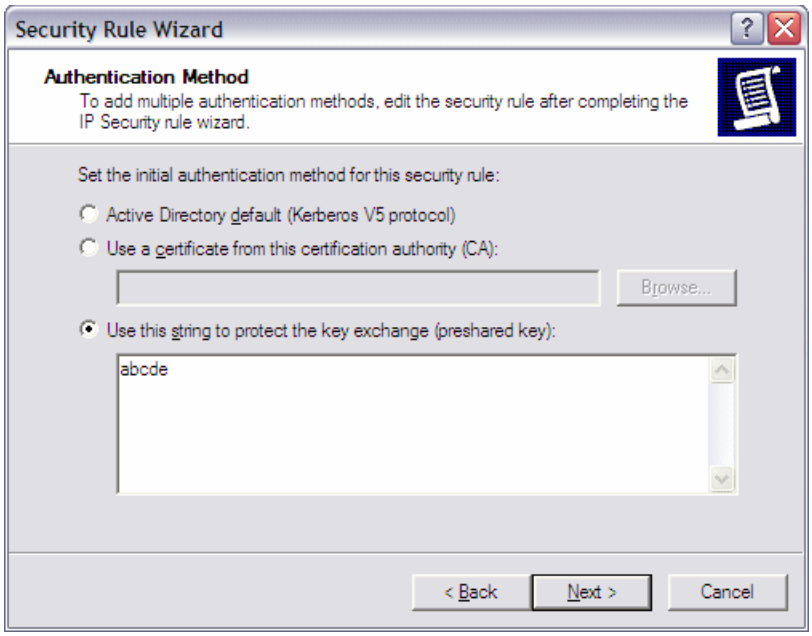


Figure C-11 Authentication Method window: Using a shared key

11. In the Security Rule Wizard - IP Filter List window, click the circle for **All IP Traffic** as shown in Figure C-12. Click **Edit**.

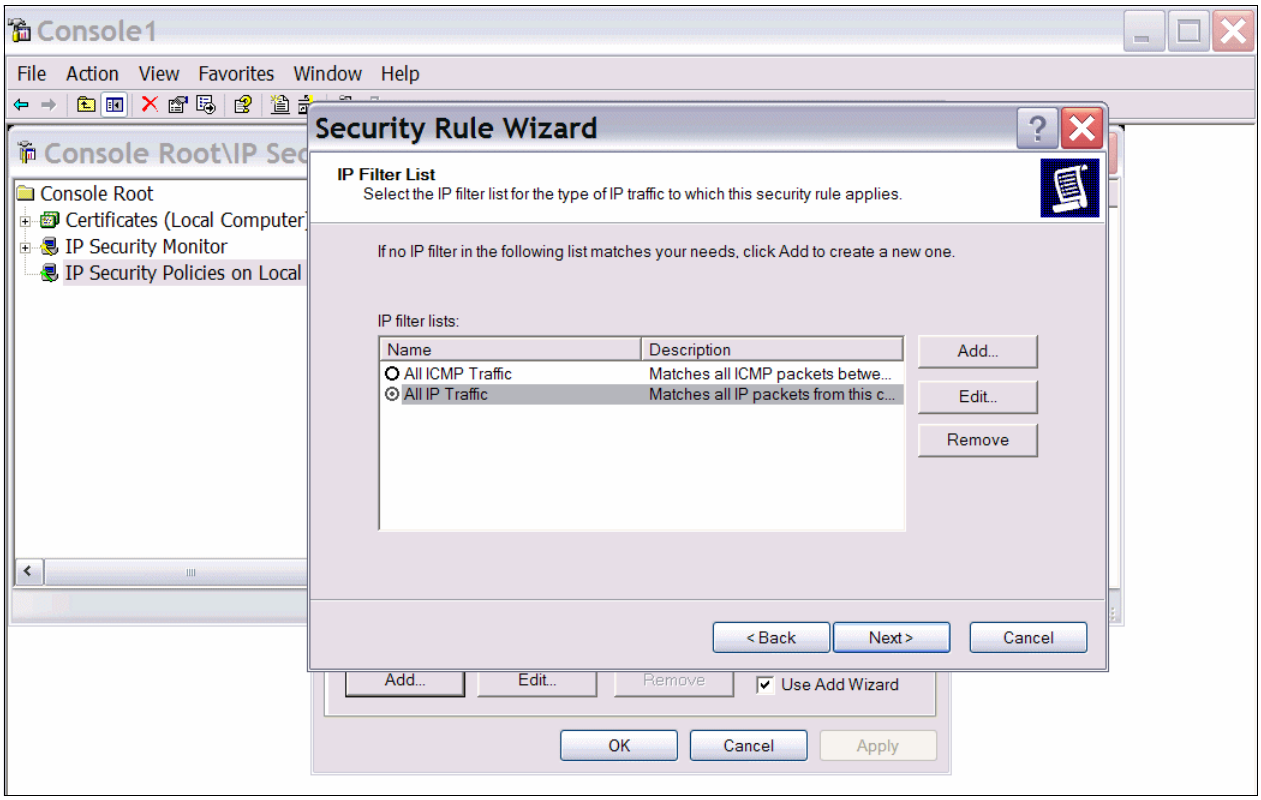


Figure C-12 Security Rule Wizard: IP filter list

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 creates 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-13.

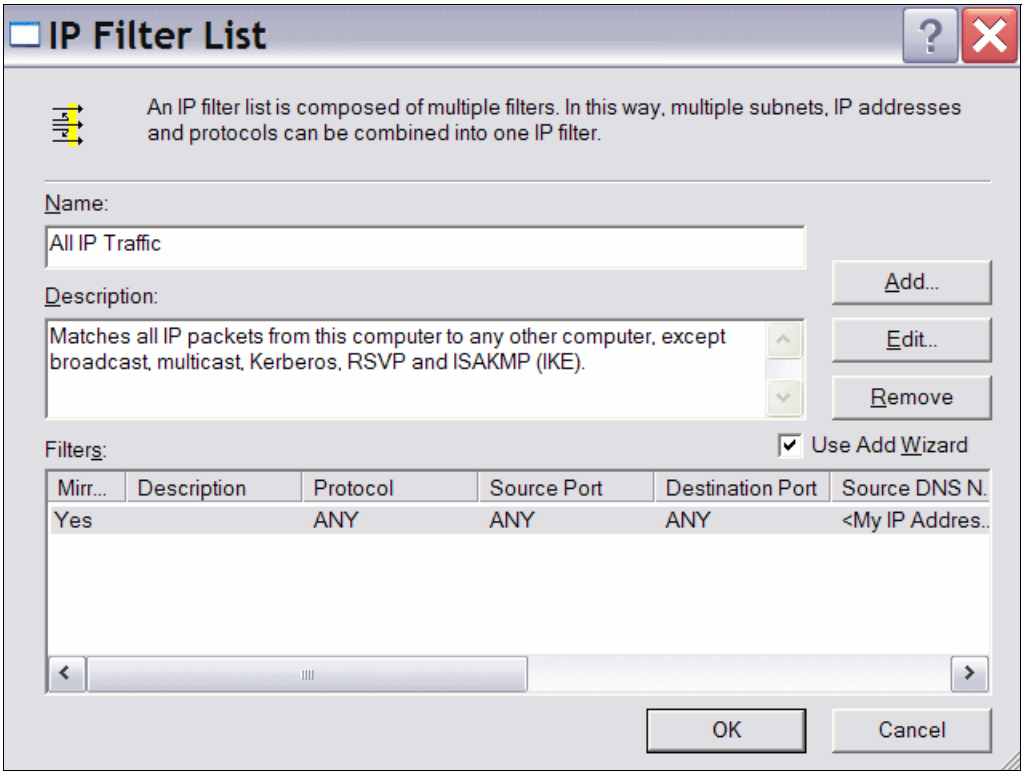


Figure C-13 IP filter list

13. In the Filter Properties window (shown in Figure C-14), select **A specific IP Address** in the Destination address column. Type the IP address of the z/OS image. This IP address also represents the VPN endpoint. Select **Mirrored. Also match packets with the exact opposite source and destination addresses**. In this example, we typed `192.168.1.40` for the Destination IP address. Click **OK**.

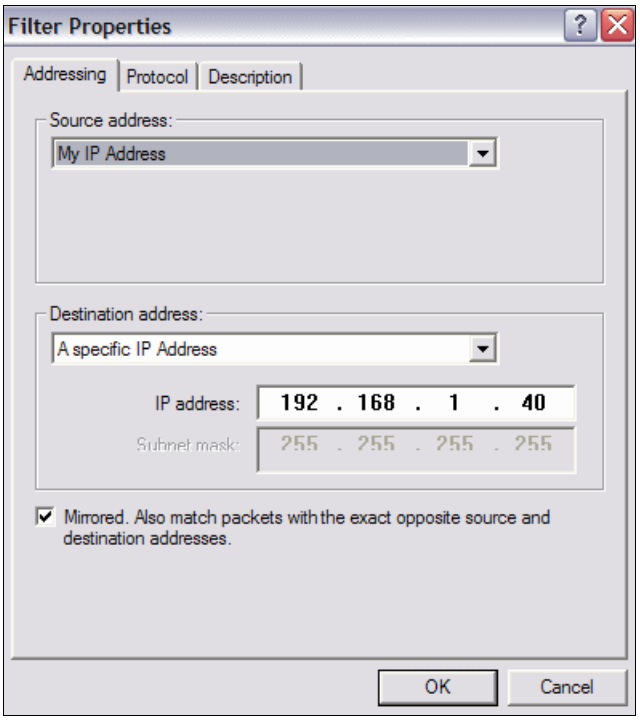


Figure C-14 Filter properties

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-15. Click **Edit**.

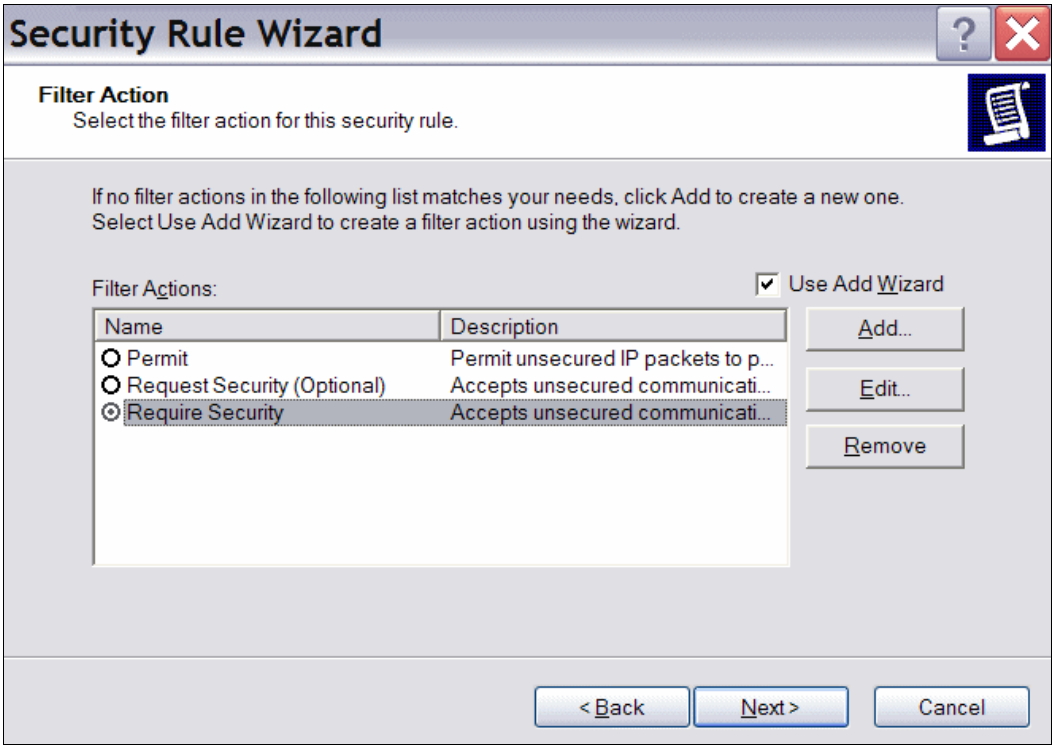


Figure C-15 Security Rule Wizard: Filter Action

17. In the Require Security Properties window, select **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-16.

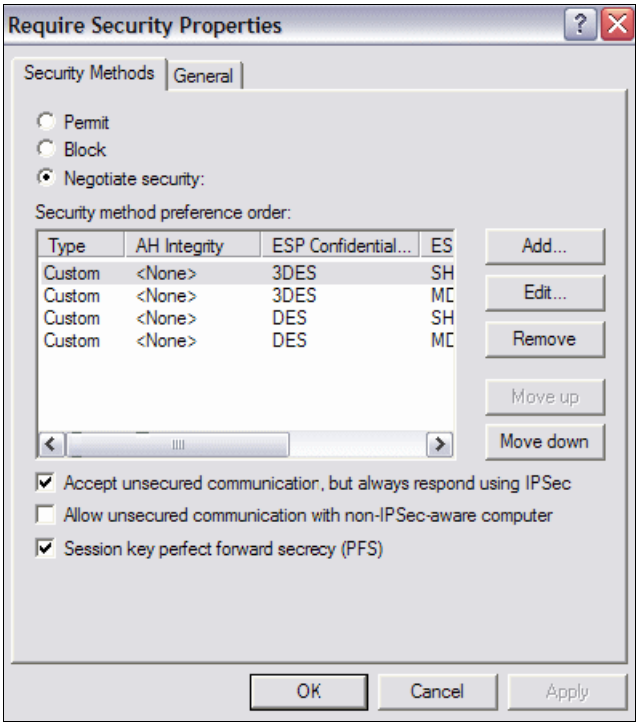


Figure C-16 Require Security Properties

18. In the Modify Security Method window, select **Custom** (for expert users). Click **Settings**, as shown in Figure C-17.

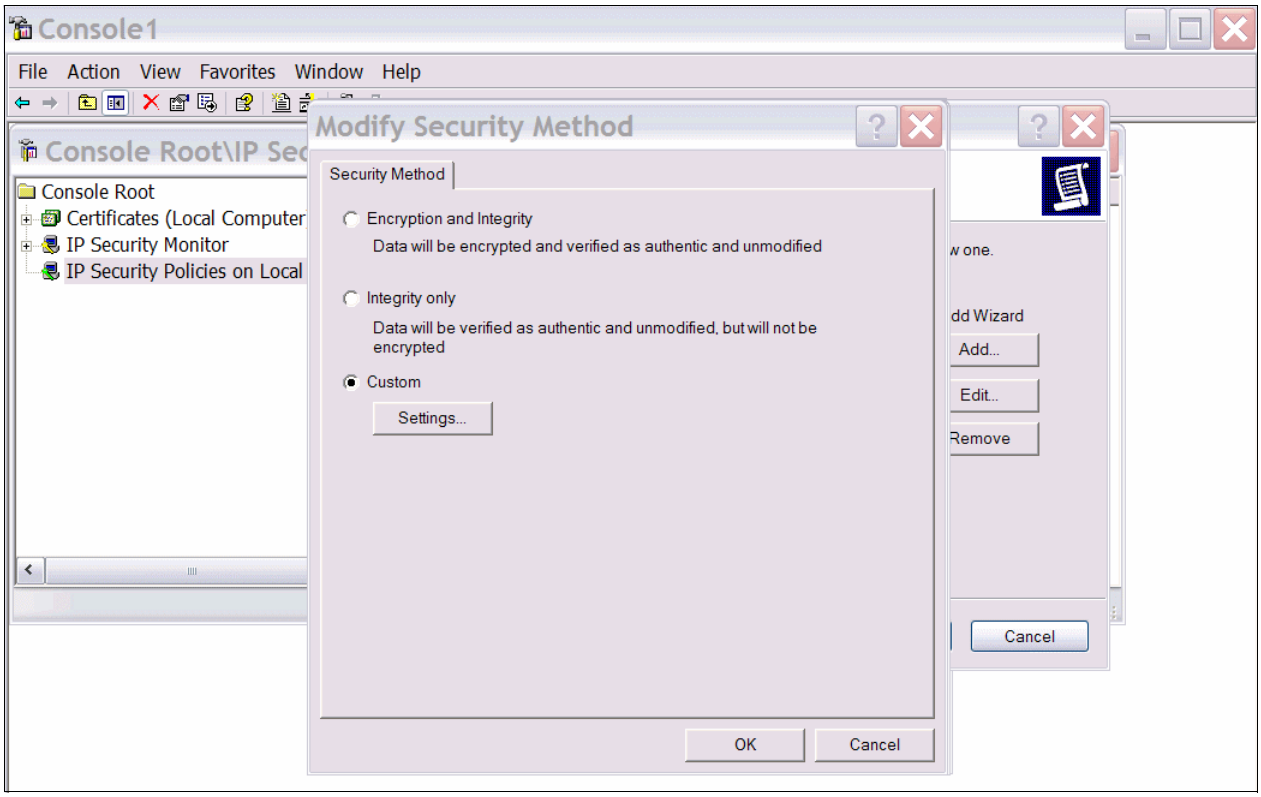


Figure C-17 Modify security method

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. Click to 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-18. Click **OK**.

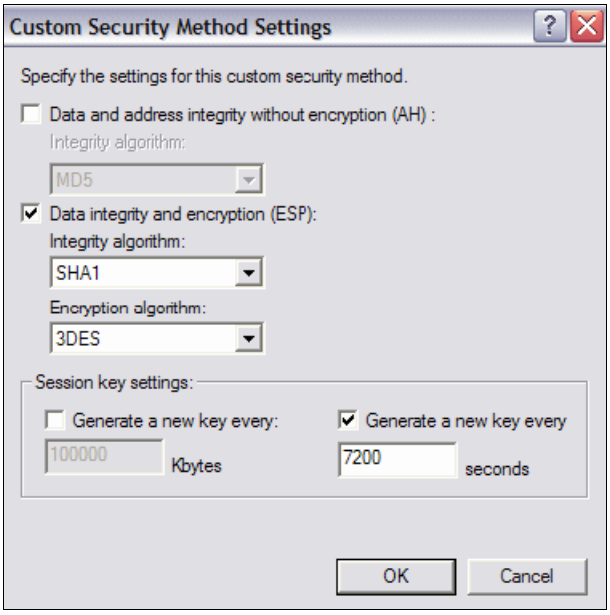


Figure C-18 Custom Security Method Settings

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, click to 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-19. Click **Close**.

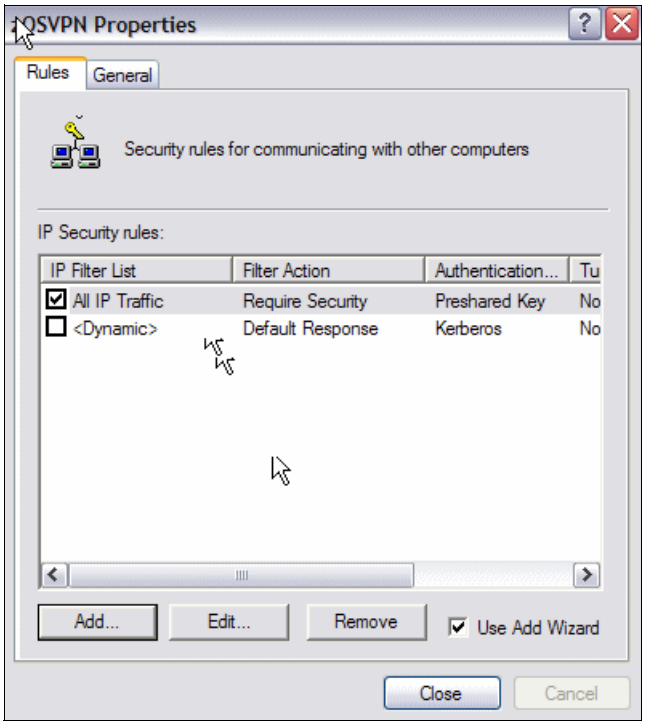


Figure C-19 zOSVPN properties

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-20.

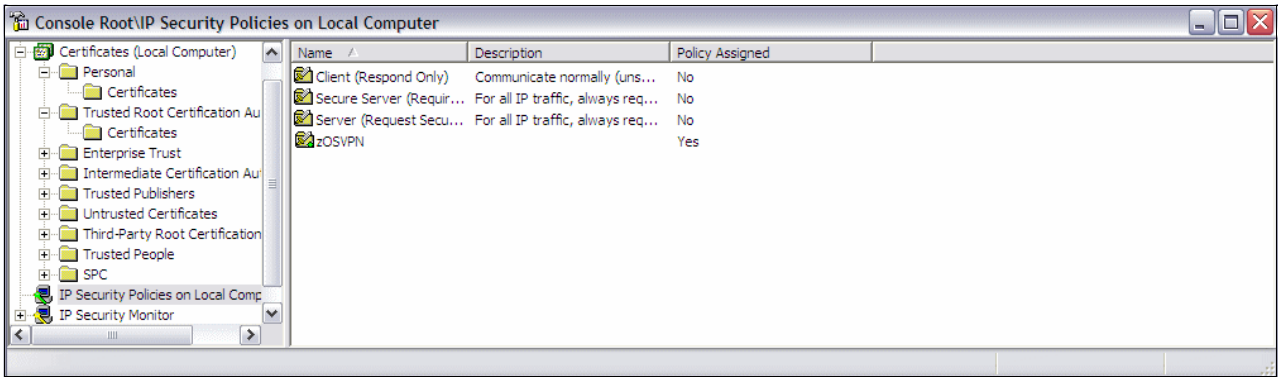


Figure C-20 IP Security Policies on Local Computer window with zOSVPN policy assigned

Verify that things are working

Complete the following steps to verify that things are working:

1. Check that IKED is running.
2. Check that the policy agent has read the new policy.

3. Check that you can run the **ping** command from Windows XP to the z/OS image.
4. Verify that phase 1 of the security association has started.
5. Verify that phase 2 of the security association has started.
6. If you have problems, check the syslogd messages.
7. Verify the security association on Windows.

Check that IKED is running

Check the IKED procedure is running and you receive syslog messages:

```
EZD0967I IKE RELEASE CS V1R13 SERVICE LEVEL CS110518 CREATED ON May
18 2011
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/iked.sc33.conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPE   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPE
EZD1046I IKE INITIALIZATION COMPLETE
```

Check that the policy agent has read the new policy

Check that the policy agent has read in the current policies.

After you use FTP to move 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 TCPIPE : NONE
```

If changes have been made, then you should receive a message like this:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : IPSEC
```

Check that the ping command is working

To verify that IP security is working, try the following **ping** command from Windows XP to the z/OS image:

```
ping 192.168.1.40
```

You should receive output similar to that shown in Example C-1.

Example C-1 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 2 has started

To verify that a dynamic tunnel has been created, run the **ipsec -y** command:

```
ipsec -p tcpipe -y display
```

You should receive output similar to that shown in Example C-2.

Example C-2 The ipsec -y command after the ping command from XP to z/OS

```
CSO2 @ SC33:/u/cs02>ipsec -p tcpipe -y display

CS V1R13 ipsec  Stack Name: TCPIPE  Mon Jul 18 16:54:15 2011
Primary:  Dynamic tunnel  Function: Display            Format:   Detail
Source:   Stack           Scope:    Current            TotAvail: 1

TunnelID:                     Y11
Generation:                   1
IKEVersion:                   1.0
ParentIKETunnelID:            K10
VpnActionName:                PFS
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
...
PassthroughDSCP:              n/a
*******************************************************************************
1 entries selected
```

Check the syslogd for messages

In our test environment, 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 SYSLOGD configuration file.

```
obrowse /SC33/var/syslog/2011/07/18/iked.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.

Tip: 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 is created, complete the following steps:

1. In the Event Viewer window, select **Security** as shown in Figure C-21.

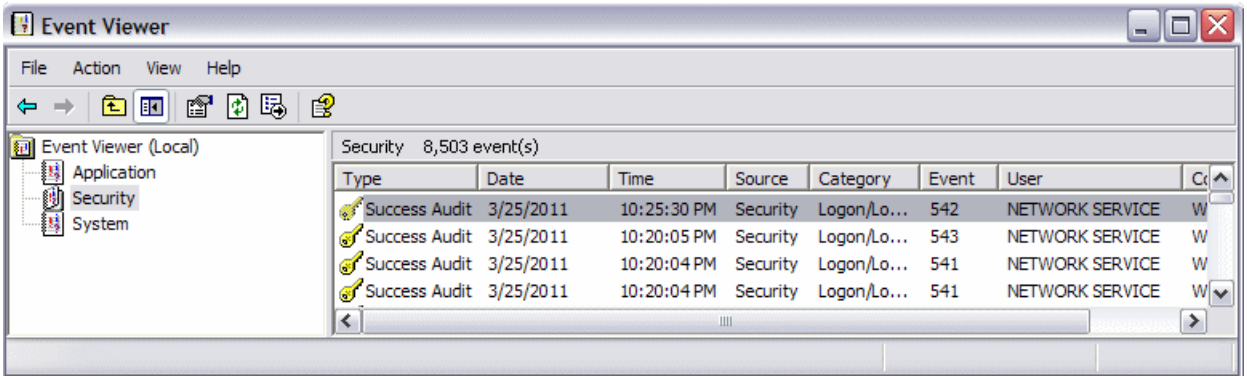


Figure C-21 Event Viewer Security option

2. Double-click an event. The window shown in Figure C-22 opens.

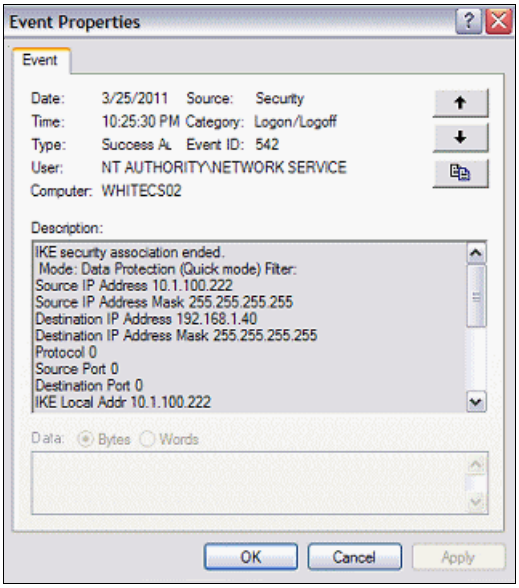


Figure C-22 Detailed Security event

Example C-3 shows a sample description of a Failure Audit case.

Example C-3 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
```

IPSec between z/OS and Windows: RSA mode

Figure C-23 shows the setup that we implemented for Dynamic Tunnels with RSA mode.

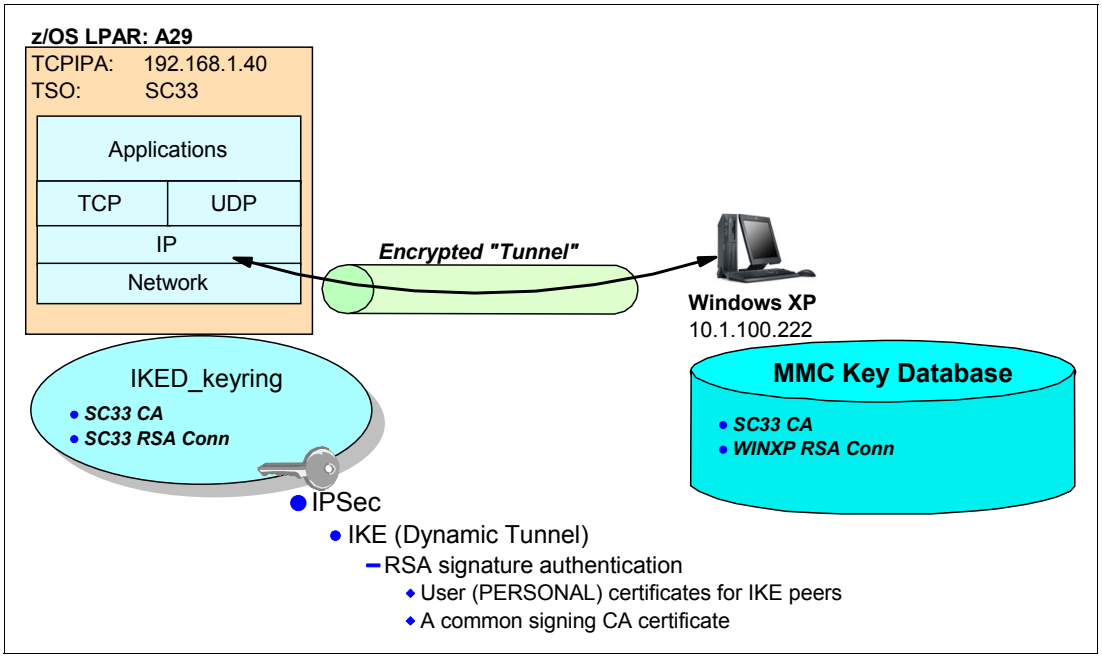


Figure C-23 VPN between z/OS and Windows

To configure a VPN between z/OS image and Windows XP using Dynamic Tunnels with RSA signature mode, follow these steps:

1. Set up the IKE daemon.
2. Set up the x.509 certificates for RSA mode.
3. Export the Certificates from RACF Database.
4. Set up the z/OS IPSec policy for RSA.

Although we do not show you the details for every step that are necessary to implement this scenario using RSA signature mode, we do include the processes that are necessary to implement the RSA.

You can implement the steps that we describe here because we have introduced the minor changes that RSA mode requires on z/OS and Windows in other sections in this book. You can verify the implementation just as it was for pre-shared key mode.

We explain these steps in more detail where necessary in the following sections.

Set up the IKE daemon

In terms of procedure, user ID, and other configuration choices, we set up the IKE daemon in our scenario using the same principles as outlined in 8.5.8, “Setting up the IKE daemon” on page 274.

We used the `_CEE_ENVFILE` environment variable to set up a STDENV DD card for controlling the environment variables as shown in Example C-4.

Example C-4 IKE daemon cataloged procedure

```
//IKED     PROC
//IKED     EXEC PGM=IKED,REGION=OK,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD PATH='/etc/iked.sc&SYSCLONE..env',PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

We did not need to specify a resolver configuration file in our scenario because, as is the case with the policy agent, one server should be used for all stacks within the image.

Example C-5 shows the contents of our STDENV file, `/etc/iked.sc33.env`.

Example C-5 /etc/iked.sc33.env contents

```
IKED_FILE=/etc/iked.sc33.conf
IKED_CTRACE_MEMBER=CTIIKE00
```

We used the configuration file located `/etc/iked.sc33.conf` as shown in Example C-6.

Example C-6 IKE daemon configuration file, /etc/iked.sc33.conf

```
IkeConfig
{
  IkeSyslogLevel    255
  PagentSyslogLevel 255
  KeyRing           IKED/IKED33_keyring
  IkeRetries        6
  IkeInitWait       2
  FIPS140           no
  Echo              no
  PagentWait        0
  SMF119            ikeALL
}
```

The configuration file includes the following variables:

- `IkeSyslogLevel` and `PagentSyslogLevel`: We set these levels to 255 during testing to obtain helpful messages. After a successful configuration, you need to use a minimum value to avoid over-filling of the `syslogd` file.
- `KeyRing`: This variable defines the RACF key ring name that is used to hold the certificate authority certificate that is required for ISAKMP/IKE authentication. This key ring was created using the user ID for the IKED started task, IKED. If the key ring was created under any other user ID, then this user ID needs to be prepended to the key ring name using the syntax `USERID/keyring`. Note, however, that accessing a key ring that is owned by another user ID requires UPDATE access to `IRR.DIGTCERT.LISTRING`.

Set up the x.509 certificates for RSA mode

Next, you need 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 establish a security association successfully.

Because the key exchange mechanism of IKE involves a direct request of the CA that is required, self-signed certificates were not an option. Instead, a CA certificate (CERTAUTH) 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. Example C-7 shows the JCL that we used.

Example C-7 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(IKED33_keyring)
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('SC33 CA')         -
           O('I.B.M Corporation') OU('ITSO') C('us'))      -
           NOTAFTER(DATE(2012-11-11))                      -
           keyusage(certsign)                              -
           WITHLABEL('SC33 CA')
RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('WINXP RSA Conn')  -
           O('I.B.M Corporation') OU('ITSO') C('us'))      -
           WITHLABEL('WINXP RSA Conn')                     -
           NOTAFTER(DATE(2012-11-11))                      -
           KEYUSAGE(HANDSHAKE) ALTNAME(IP(10.1.100.222))   -
           SIGNWITH(CERTAUTH Label('SC33 CA'))
RACDCERT ID(IKED) GENCERT SUBJECTSDN(CN('SC33 RSA Conn')   -
           O('I.B.M Corporation') OU('ITSO') C('us'))      -
           NOTAFTER(DATE(2012-11-11))                      -
           WITHLABEL('SC33 RSA Conn')                      -
           KEYUSAGE(HANDSHAKE) ALTNAME(IP(192.168.1.40))   -
           SIGNWITH(CERTAUTH Label('SC33 CA'))
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('SC33 RSA Conn')  -
             RING(IKED33_keyring) USAGE(personal))
RACDCERT ID(IKED) CONNECT(ID(IKED) LABEL('WINXP RSA Conn') -
             RING(IKED33_keyring) USAGE(personal))
```

```
RACDCERT ID(IKED) CONNECT(CERTAUTH LABEL('SC33 CA') -
            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
racdcert listring(IKED33_keyring) ID(IKED)
/*
```

Export the Certificates from RACF Database

To keep the configuration scenario simple, we used the same CA and personal certificate on the XP workstation. To do this, we exported the certificates from the RACF database into the MVS data sets using the commands shown in Example C-8.

Example C-8 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 CA')) -
     DSN('CSO2.CERT.CACERT') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
  RACDCERT ID(IKED) EXPORT(LABEL('WINXP RSA Conn')) -
     DSN('CSO2.CERT.XPCERT') -
     FORMAT(PKCS12DER) -
     PASSWORD('security')
/*
```

Note that the CA (signing) certificate does not require a private key. However, the personal certificate should include the private key. Thus, we used the `PKCS12DER` format along with a password.

These exported files are used later during the windows IPSec implementation.

Set up the z/OS IPSec policy for RSA

To simplify our RSA scenario, we used the IPSec definitions that we created earlier in this appendix and changed our IPSec connectivity rule to use RSA. For further information about the IPSec definitions, review “Set up the z/OS IPSec policy” on page 861.

To implement RSA, we renamed and changed an IPSec Connectivity rule name to `All_Traffic_RSA` using the z/OSMF Configuration Assistant as follows:

1. Open the z/OSMF Configuration Assistant.
2. In the Main Perspective window, z/OS Communications Server technologies table, select **IPSec** technology, then click **Select Action** → → **Configure**.

3. In the IPSec Perspective window, in the IBM Configuration Assistant Navigation tree, select **z/OS Images** → **Image -** ***image_name*** → **Stack** - ***stack_name***.
4. In the Connectivity Rules table, select the rule named **All_Traffic_PFS**. Click **Select Action** then **Modify** to change this connectivity rule.
5. In the Connectivity Rule: Data Endpoints window, change the name of the rule. In our configuration, we entered `All_Traffic_RSA`.
6. Next, select the Remote Security Endpoint tab. For our scenario, we used the x.500 distinguished name identity to identify the peers and used this identity for our scenario with RSA mode. In the Indicate how to authenticate the IKE peers section, select **Digital signature**. Then, select the X.500 distinguished name identity in the Remote identity type field as shown in Figure C-24 on page 887. This name can be retrieved from the RACF database using the contents of the “Subject’s Name” field in the display output of the command:

```
racdcert id(IKED) list(label('WINXP RSA Conn'))
```

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 as follows:

```
RDN<period>RDN<period>RDN<period>RDN
```

However, the syntax that is required in the IKE definition window (Figure C-24 on page 887) requires a comma (,) as a delimiter. The coding pattern in this case is as follows:

```
RDN<comma>RDN<comma>RDN<comma>RDN
```

Therefore, the display output of Subject’s Name is as follows:

```
CN=WINXP RSA Conn.OU=ITSO.O=I.B.M Corporation.C=us
```

This line must be converted to the following (using commas instead of periods) when it is defined as an IKE identity in the IBM Configuration Assistant:

```
CN=WINXP RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us
```

The IBM Configuration Assistant GUI builds the `iked.conf` file from these definitions.

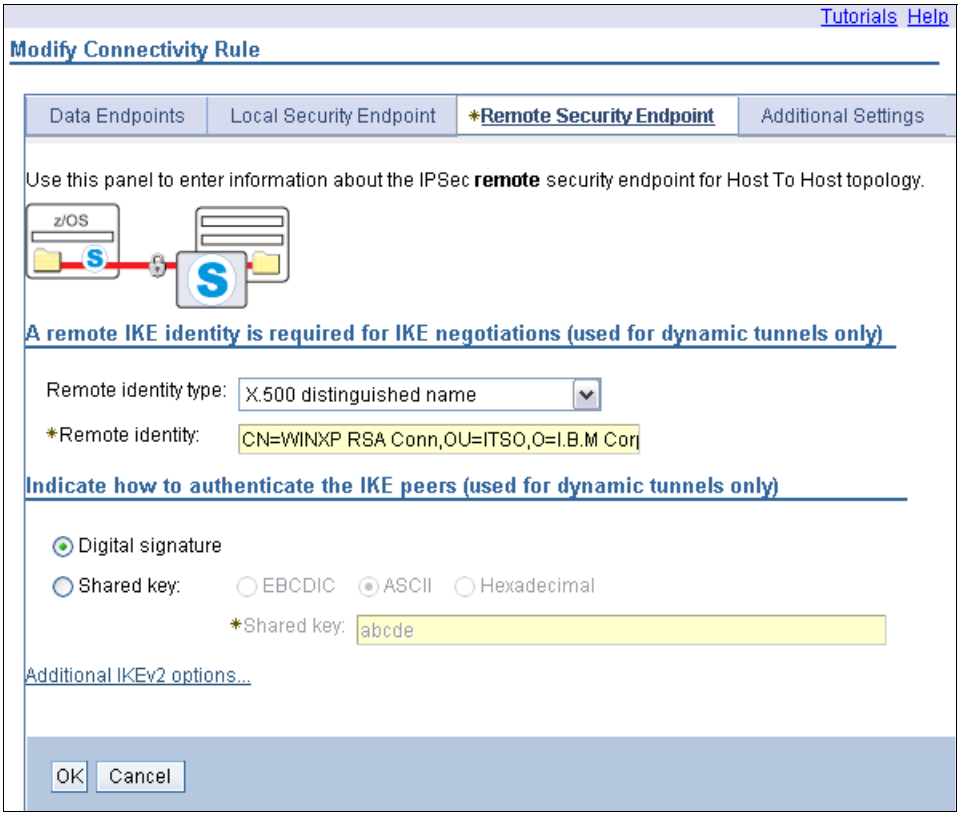


Figure C-24 Connectivity Rule: Remote Security Endpoint window

7. Click the Connectivity Rule: Additional Settings tab and select **Yes, log all filter matches**. Click **OK**.
8. Back in the IPSec perspective window, select the rule named **All_Traffic_RSA**. Click **Select Action** → **Modify** then click the Local Security Endpoint tab. Select the X.500 distinguished name identity type, and write the local identity as follows:

   ```
   CN=SC33 RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us
   ```

   Click **OK,** then **Apply Changes**, and then click the Rules tab.

   The IPSec Perspective window now includes the new rules in the Connectivity Rules table as shown in Figure C-25 on page 888.

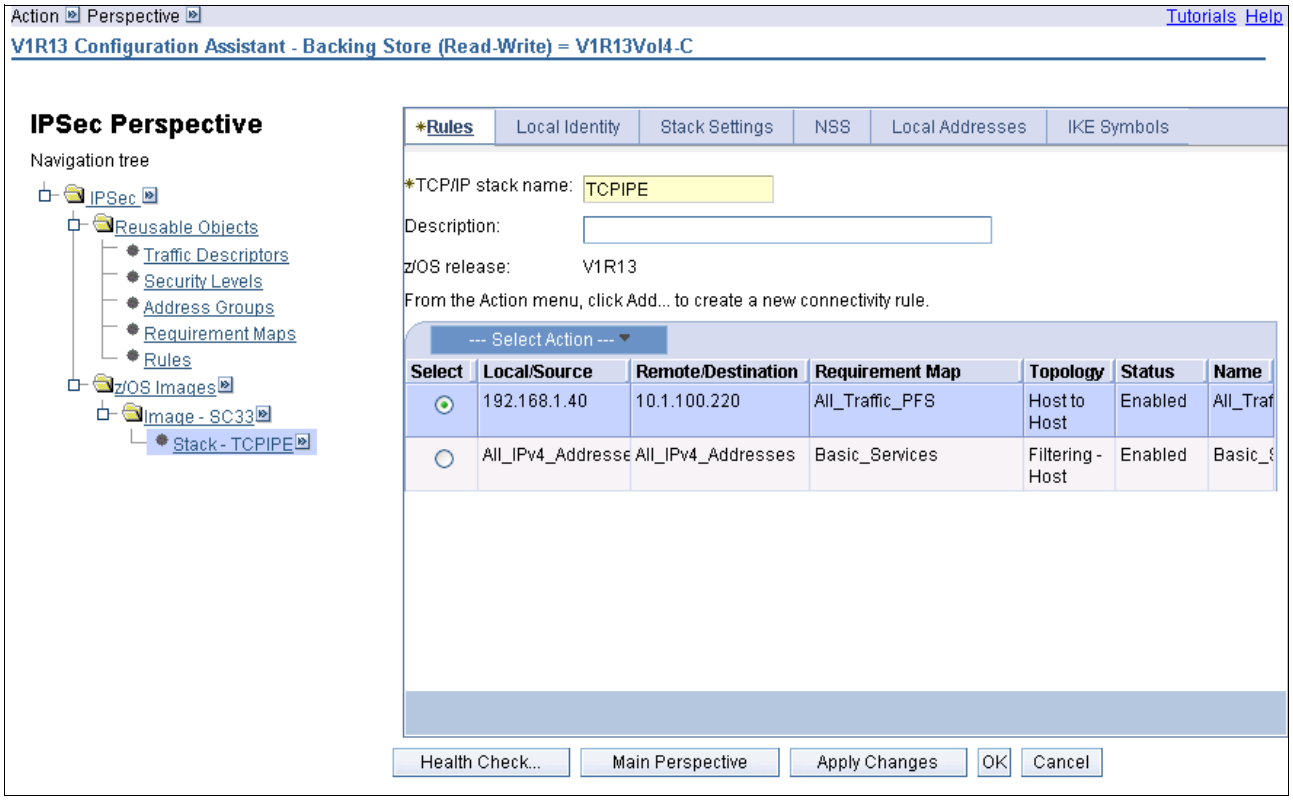


Figure C-25 IPSec perspective window with the All_Traffic_RSA rule included

Tip: Verify that the rules are in the correct order; use **Select Action** then **Move Up** if necessary to change the order of the rules. Packets are checked against the connectivity rules in the order they appear in the table. If there is a match with the definition of this first rule, it is used. If there is no match, the second rule is checked, and so on.

For performance, order the rules to ensure the majority of the traffic will match the first rules listed if possible.

Inherent in the rules that we discuss 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.

Updating the IPSec policy of the z/OS image

Install the new policy to the z/OS image as explained in 8.4, “Working with the z/OS Communications Server Network Management Interface” on page 262. 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 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 is handled on the Windows XP workstation.

Set up a Windows IPSec policy for RSA mode

In this section, we describe how to set up the Microsoft Management Console (MMC). The MMC will have two snap-ins added:

- Certificates

  The Certificates Snap-in is used to import the certificates that are created by z/OS RACF.

- IP Security Management

  The IP Security Management Snap-in is used to configure the VPN connection between the Windows XP client and the z/OS server.

To set up a Windows IPSec policy for RSA mode, follow these steps:

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 **File** → **Add/Remove Snap-in**.
4. In the Add/Remove Snap-in window, click the Standalone tab and then 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. In the Select Computer window, select **Local computer**, and click **Finish**.
8. Back 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, click **OK**.

12. Back to the Console1 window, verify that two snap-ins have been added (Certificates (Local computer) and IP Security Policies on Local Machine), as shown in Figure C-26.

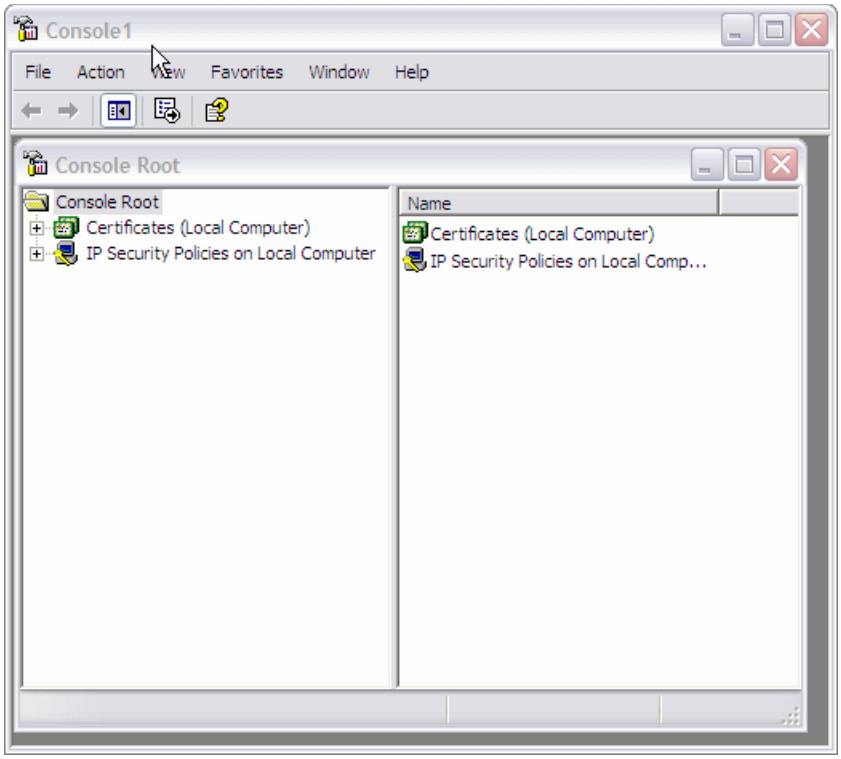


Figure C-26 Console1 window with the new snap-ins added

Import 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 CA certificate
- 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.

We transferred the certificate files in z/OS image that we exported earlier in this scenario in “Export the Certificates from RACF Database” on page 885 using FTP, as shown in Example C-9.

Example C-9 Transfer the certificates to the Windows XP client

```
C:\Documents and Settings\RESIDENT>ftp wtsc33.itso.ibm.com
Connected to wtsc33.itso.ibm.com.
220-FTPMVS1 IBM FTP CS V1R13 at wtsc33.ITSO.IBM.COM, 16:16:48 on 2011-07-19.
220 Connection will close if idle for more than 5 minutes.
User (wtsc33.itso.ibm.com:(none)): cs02
331 Send password please.
Password:
230 CS02 is logged on.  Working directory is "CS02.".
ftp> bin
200 Representation type is Image
ftp> get cert.xpcert xpclient.p12
200 Port request OK.
125 Sending data set CS02.CERT.XPCERT
250 Transfer completed successfully.
ftp: 2492 bytes received in 0.04Seconds 62.30Kbytes/sec.
ftp> get cert.cacert racfca.p12
200 Port request OK.
125 Sending data set CS02.CERT.CACERT
250 Transfer completed successfully.
ftp: 1718 bytes received in 0.01Seconds 171.80Kbytes/sec.
ftp>.
```

Tip: 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 Console1 window, click the plus sign (+) next to Certificates (Local Computer) to show the list of available tasks.
2. Right-click **Trusted Root Certification Authorities** and select **All Tasks** from the menu. Select **Import** from the next menu and click it.
3. In the Certificate Import Wizard window, click **Next**.
4. In the Certificate Import Wizard - File to Import window, specify the Trusted Root CA file name (in this example, we choose `C:\racfca.p12` for the Trusted Root CA file). Click **Next**.
5. Work through the Certificate Import Wizard, entering the following information:
   a. Enter the password that you gave to the CA certificate. We entered `security` as the password.
   b. Do *not* select the **Mark this key as exportable** option.
   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 receive a message that indicates 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 our scenario, **SC33 CA** is installed as a Trusted Root CA as shown in Figure C-27.

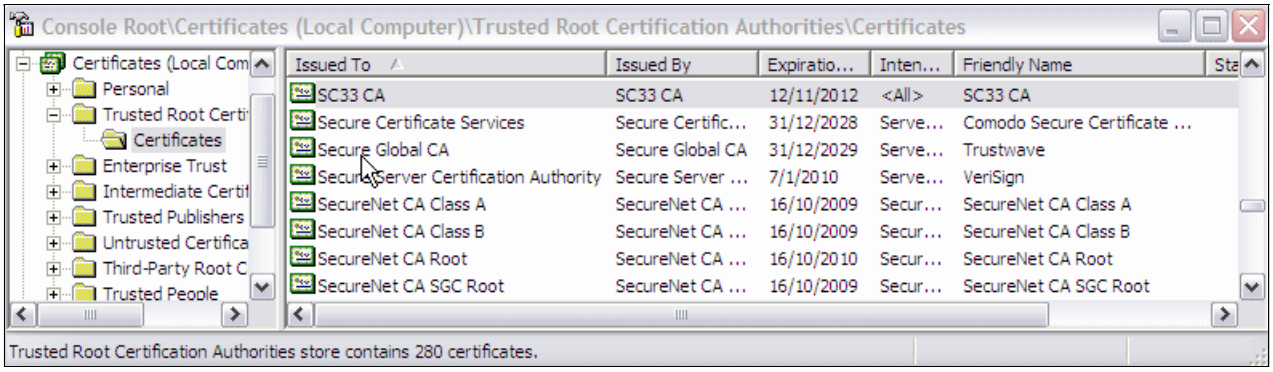


*Figure C-27 The Certificates list with the new **SC33 CA** certificate*

7. Right-click **Personal** and choose **All Tasks** from the menu. Choose **Import** from the next menu and select it.
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). Click **Next**.
10. Continue to work through the Certificate Import Wizard, indicating the following information:
    a. Enter the password you gave to the certificate. We entered `security` as the password. Remember, do *not* select the **Mark the private key as exportable** option.
    b. Make sure the **Place all certificates in the following store** option is selected and that `Personal` is shown in the certificate store column. Click **Next**.
    c. Click **Finish** in the Completing the Certificate Import Wizard window.

    You receive a message that indicates 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 RSA Conn** is installed as a client certificate as shown in Figure C-28.

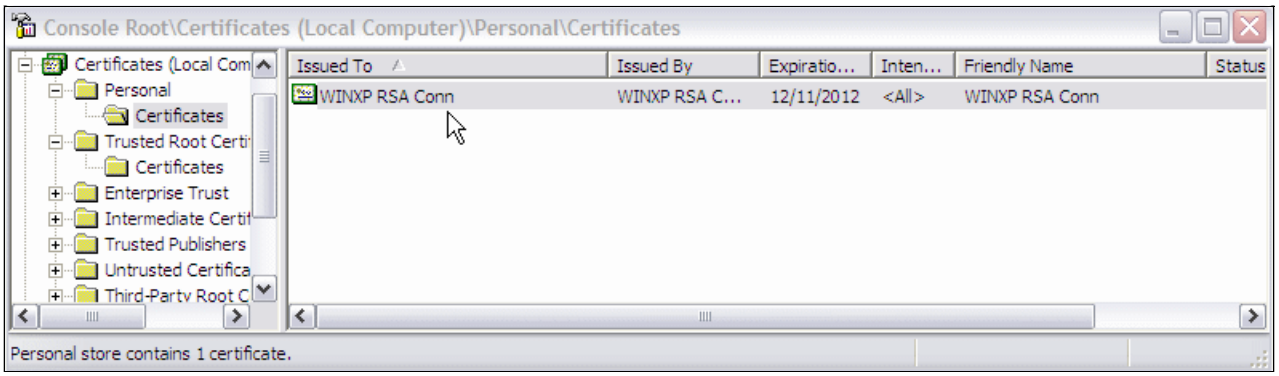


Figure C-28 Personal certificates list with WINXP RSA Conn certificate

Create the IP security policy

To create the IP Security policy on the Windows XP workstation for the VPN connection between z/OS and the Windows XP client, follow these steps:

1. In the MMC - IP Security Policies on Local Computer window, right-click **IP Security Policies on Local Computer**. In the menu, click **Create IP Security Policy** as shown in Figure C-29.

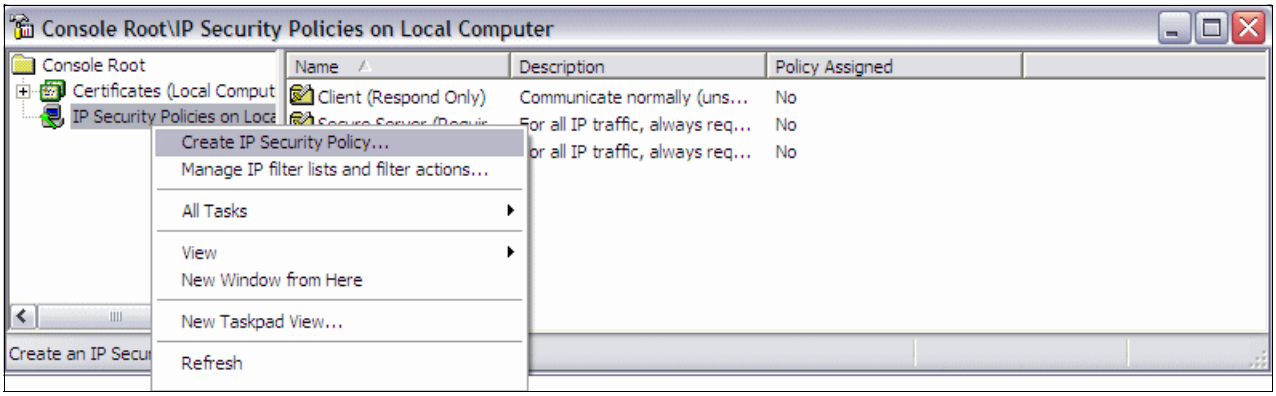


Figure C-29 MMC: IP Security Policies on Local Computer

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 check mark from the **Activate the default response rule** option. Click **Next**.
5. In the IP Security Policy Wizard - Completing the IP Security Policy Wizard, make sure that **Edit properties** is selected. Click **Finish**.
6. In the IP Policy Properties window (in this example, the zOSVPN Properties window), select **Use Add Wizard**. Click **Add**.
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.

Tip: If you want to limit the remote access connection, select **Remote Access**.

10. In the IP Security Policy Wizard - Authentication Method window, select **Use a certificate from this certificate authority (CA)**, click Browse then select the CA certificate that signed the z/OS server certificate as shown in Figure C-30. Click **Next**.

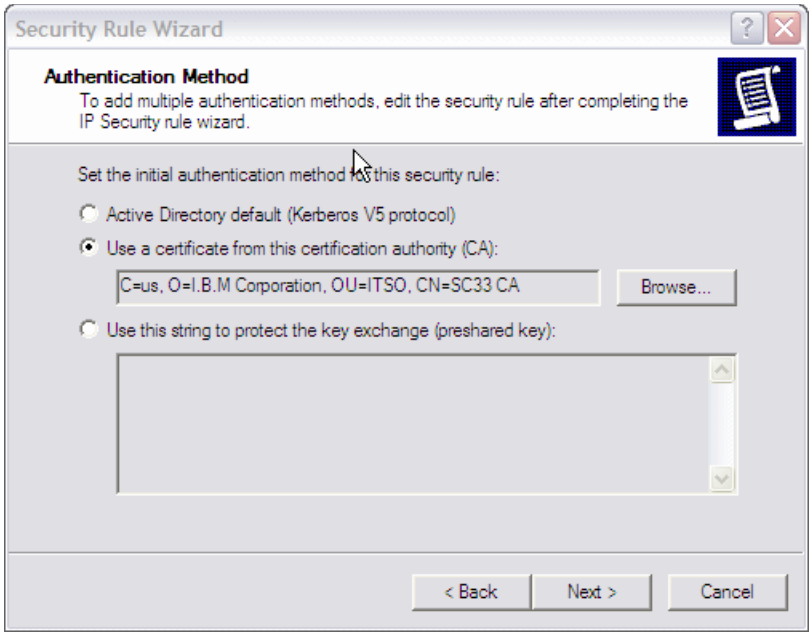


Figure C-30 Authentication Method window: using a certificate option

11. In the Security Rule Wizard - IP Filter List window, select **All IP Traffic**. Click **Edit**.

> **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 creates a VPN tunnel between the Windows XP and 192.168.1.40.

12. In the IP filter List window, click **Edit**.
13. In the Filter Properties window (shown in Figure C-31 on page 895), 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**.

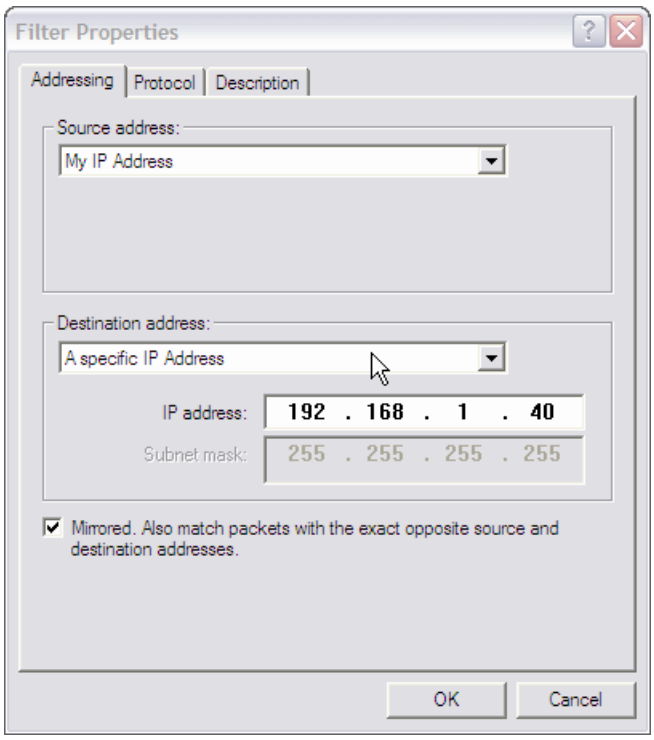


Figure C-31 Filter properties

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, select **Require Security**. Click **Edit**.
17. In the Require Security Properties window (Figure C-32 on page 896), make the following selections:
    - Negotiate security
    - Accept unsecured communication, but always respond using IPSec
    - 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**. Select the topmost security method and click **Edit**.

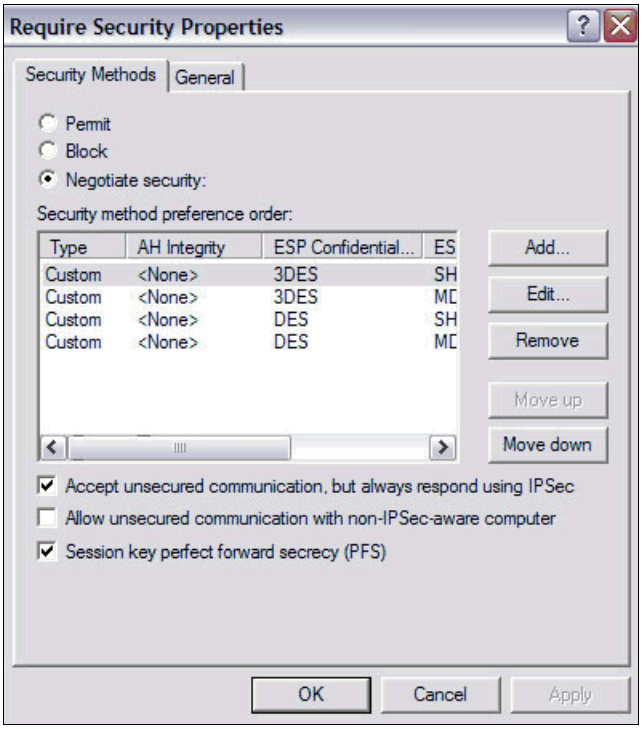


Figure C-32 Require Security Properties

18. In the Modify Security Method window, choose **Custom** (for expert users). Click **Settings**.
19. In the Custom Security Method Settings window, make sure that the “Data and address integrity without encryption (AH)” option 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. Click **OK**.
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” option and click **Finish**.
24. In the zOSVPN Properties window, make sure that **All IP Traffic** is selected. Click **Close**.
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-33 on page 897.

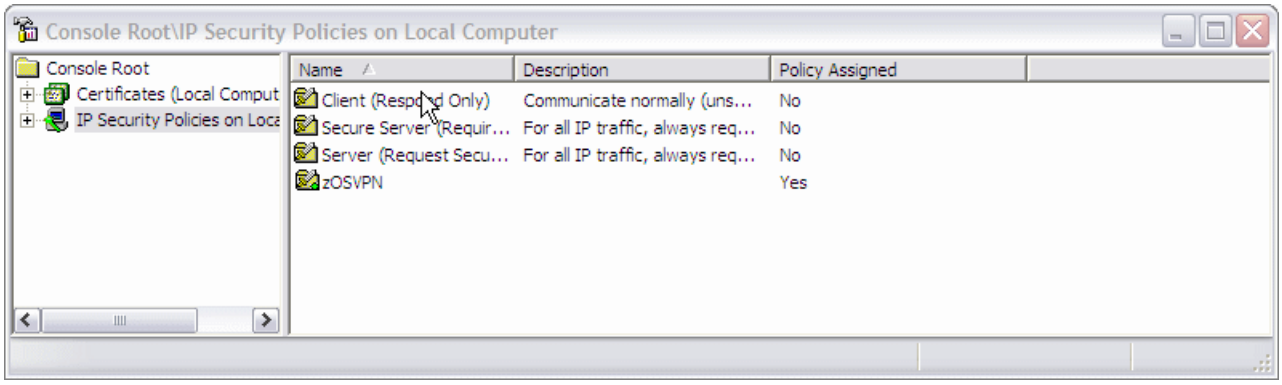


Figure C-33 IP Security Policies on Local Computer window with zOSVPN policy assigned

Verify that things are working

To verify the IPSec tunnel is working as expected, we can follow the same steps we executed for the IPSec tunnel with pre-shared key mode:

1. Check IKED is running.
2. Check that the policy agent has read the new policy.
3. Check that you succeed in running the **ping** command from the Windows XP to the z/OS image.
4. Verify that phase 1 of the security association has started.
5. If you have problems, check the syslogd messages.
6. Verify the security association on Windows.

For more information about the verification steps, see “Verify that things are working” on page 878.

Check that IKED is running

Check that the IKED procedure is running and you are receiving syslog messages such as this:

```
EZD0967I IKE RELEASE CS V1R13 SERVICE LEVEL CS110518 CREATED ON May
18 2011
EZD0911I IKE CONFIG PROCESSING COMPLETE USING FILE /etc/iked.sc33.conf
EZD1061I IKE CONNECTING TO PAGENT
EZD1059I IKE CONNECTED TO PAGENT
EZD1058I IKE STATUS FOR STACK TCPIPE   IS UP
EZD1068I IKE POLICY UPDATED FOR STACK TCPIPE
EZD1046I IKE INITIALIZATION COMPLETE
```

Check that the policy agent has read the new policy

First, check that the policy agent has read in the current policies. After you move the configuration file using FTP 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 that the polices were read, you receive a message as follows:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : NONE
```

If changes have been made, then you receive a message as follows:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPE : 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-10.

Example C-10 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 IPSec tunnel has been created, run the **ipsec -k** command as shown in Example C-11:

```
ipsec -p tcpipe -k display
```

Example C-11 The ipsec -k command after the ping command from XP to z/OS

```
CSO2 @ SC33:/u/csO2>ipsec -p tcpipe -k display

CS V1R13 ipsec  Stack Name: TCPIPE  Tue Jul 19 13:17:39 2011
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                     K1
Generation:                   1
IKEVersion:                   1.O
KeyExchangeRuleName:          All_Traffic_RSA~5
KeyExchangeActionName:        All_Traffic_RSA
LocalEndPoint:                192.168.1.40
LocalIDType:                  ID_DER_ASN1_DN
LocalID:                      CN=SC33 RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us
RemoteEndPoint:               1O.1.1OO.222
RemoteIDType:                 ID_DER_ASN1_DN
RemoteID:                     CN=WINXP RSA Conn,OU=ITSO,O=I.B.M Corporation,C=us
ExchangeMode:                 Main
State:                        DONE
AuthenticationAlgorithm:      HMAC-SHA1
EncryptionAlgorithm:          3DES-CBC
 KeyLength:                   n/a
PseudoRandomFunction:         HMAC-SHA1
DiffieHellmanGroup:           2
LocalAuthenticationMethod:    RsaSignature
RemoteAuthenticationMethod:   RsaSignature
InitiatorCookie:              0x6790104CA1ED3750
ResponderCookie:              0xF8O67ABO4635BA94
Lifesize:                     OK
CurrentByteCount:             608b
Lifetime:                     480m
LifetimeRefresh:              2011/07/19 19:53:51
LifetimeExpires:              2011/07/19 20:03:30
ReauthInterval:               480m
ReauthTime:                   2011/07/19 19:53:51
Role:                         Responder
AssociatedDynamicTunnels:     0
NATTSupportLevel:             None
NATInFrntLclScEndPnt:         No
NATInFrntRmtScEndPnt:         No
zOSCanInitiateP1SA:           Yes
AllowNat:                     No
RmtNAPTDetected:              No
RmtUdpEncapPort:              n/a
*******************************************************************************
```

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 following command:

```
tso omvs
```

From the UNIX System Services environment, browse the log file that you defined for IKED in the IKED configuration file.

```
obrowse /SC33/var/syslog/2011/07/19/iked.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 correct configuration file.

Tip: The IKED log file has an important role in problem determination. When implementing IP security for the first time, verify the following items:

- IKED is configured to write log messages.
- TRMD is running.
- The syslogd is running with the correct configuration file.

Verify the security association on Windows

To verify that a dynamic tunnel was created, in the Event Viewer window shown in Figure C-34, select **Security**.

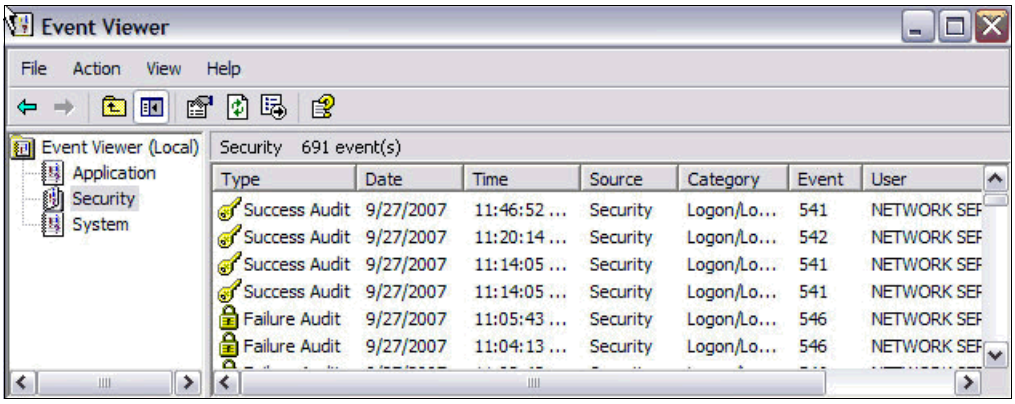


Figure C-34 Event Viewer -Security option

Double-click the first event in the list. A window opens as shown in Figure C-22 on page 881.

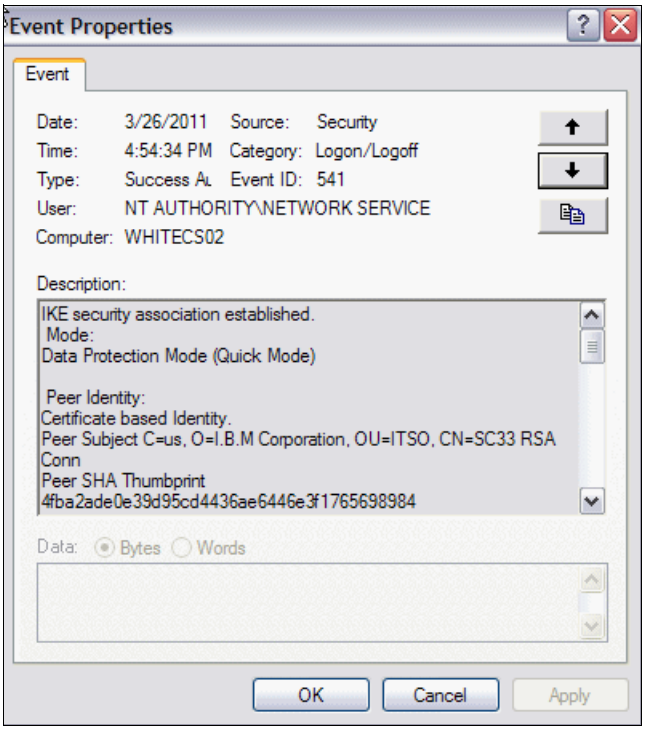


Figure C-35 Detailed security event

D

zIIP Assisted IPSec

The IBM System z Integrated Information Processor (zIIP) is a specialty engine that is 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.

We discuss the following topics in this appendix.

Section	Topic
“Background” on page 904	Achieve a significant reduction in general purpose CP consumption by using the zIIP
“Configuring zIIP Assisted IPSEC” on page 904	Configuration options
“Example of zIIP Assisted IPSec implementation” on page 905	Provides an example of how to implement a zIIP Assisted IPsec

Background

The IP Security function of z/OS Communications Server uses the System z crypto hardware, called Crypto Express (CE) or CP Assist for Cryptographic Function (CPACF), which is standard on every processing unit (for example, IFL, CP). CE2, CE3, and CPACF are used for data encryption and decryption, and for authentication.

Both CPACF (in conjunction with Integrated Cryptographic Service Facility (ICSF)) and CE3 create an increase in CPU utilization when IPSec is added, especially CPACF because it runs on the general purpose CPs. Even using CE3, 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 certain 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, which can magnify this problem. To minimize the problem the IPSec workload could run on a separate 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 might 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 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, navigate to the following URL 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 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 might 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.

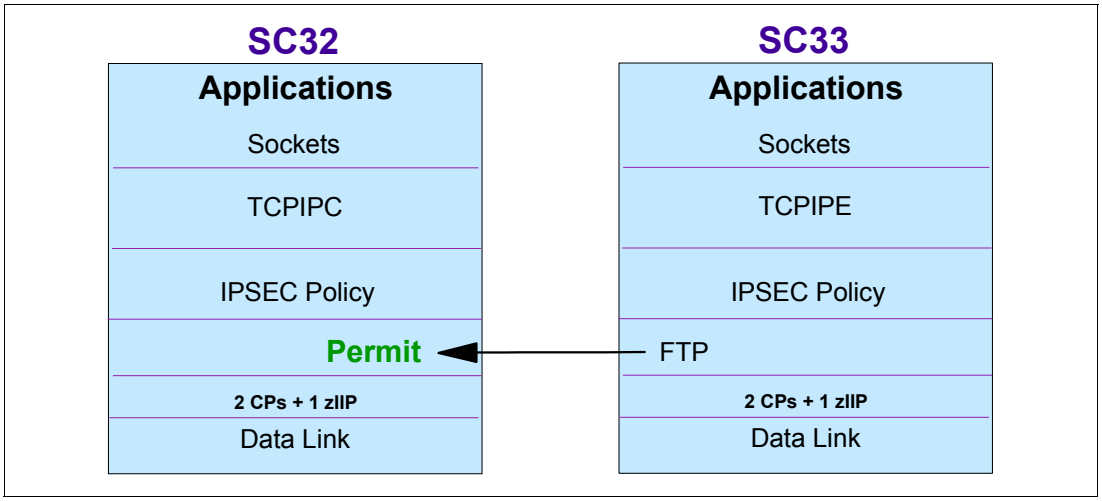


Figure D-3 Network configuration

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

```
D M=CPU

IEE174I 14.25.53 DISPLAY M 602
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                    163BD52817
01  +                    163BD52817
02  +A                   163BD52817
03  +I                   163BD52817
04  -
05  -
06  -A
07  -I

CPC ND = 002817.M32.IBM.02.0000000B3BD5
CPC SI = 2817.715.IBM.02.00000000000B3BD5
         Model: M32
CPC ID = 00
CPC NAME = SCZP301
LP NAME = A16        LP ID = 16
CSS ID  = 1
MIF ID  = 6

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

A        APPLICATION ASSIST PROCESSOR (zAAP)
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.

Example D-2 Display CPU from SC33

```
D M=CPU

IEE174I 14.27.12 DISPLAY M 832
PROCESSOR STATUS
ID  CPU                 SERIAL
00  +                    183BD52817
01  +                    183BD52817
02  +A                   183BD52817
03  +I                   183BD52817
04  -
```

```
05  -
06  -A
07  -I

CPC ND = 002817.M32.IBM.02.0000000B3BD5
CPC SI = 2817.715.IBM.02.00000000000B3BD5
         Model: M32
CPC ID = 00
CPC NAME = SCZP301
LP NAME = A18        LP ID = 18
CSS ID  = 1
MIF ID  = 8

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

A        APPLICATION ASSIST PROCESSOR (zAAP)
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
```

To define a VPN in z/OS, you need 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 255
TcpImage TCPIPE /etc/pagent.sc33.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 255
TcpImage TCPIPC /etc/pagent.sc32.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/pagent.sc33.tcpipe.ipsec.ziip.policy
```

Figure D-8 SC33 TCP/IP stack policy

Figure D-9 shows the SC32 TCP/IP stack policy.

```
IPSecConfig  /etc/pagent.sc32.tcpipc.ipsec.ziip.policy
```

Figure D-9 SC32 TCP/IP stack policy

We used the IPSEC configuration for the SC32 and SC33 created in 8.4, “Working with the z/OS Communications Server Network Management Interface” on page 262. We used the IBM Configuration Assistant to build this environment.

Only two IPSEC rules are 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.

> **Tip:** An implicit rule denying all traffic is always created by having either the default policy rules or the pagent policy.

For this implementation, we created one batch FTP job to transfer data. Each 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.

```
D TCPIP,TCPIPC,N,STATS
IP STATISTICS (IPV4)
  PACKETS RECEIVED                  = 192225
  RECEIVED HEADER ERRORS            = 0
  RECEIVED ADDRESS ERRORS           = 0
  DATAGRAMS FORWARDED               = 0
  UNKNOWN PROTOCOLS RECEIVED        = 4
  RECEIVED PACKETS DISCARDED        = 0
  RECEIVED PACKETS DELIVERED        = 198806
  OUTPUT REQUESTS                   = 1622429
  OUTPUT DISCARDS NO ROUTE          = 4
...
REASSEMBLY FAILURES               = 0
  DATAGRAMS SUCCESSFULLY FRAGMENTED  = 0
  DATAGRAMS FAILING FRAGMENTATION    = 0
  FRAGMENTS CREATED                  = 0
  INBOUND PACKETS HANDLED BY ZIIP    = 48
  OUTBOUND PACKETS HANDLED BY ZIIP   = 14
```

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 TCPIP Name: TCPIPC
Global Configuration Information:
TcpIpStats: No   ECSALimit: 0000000K  PoolLimit: 0000000K
MlsChkTerm: No   XCFGRPID:            IQDVLANID: 0
SegOffLoad: No   SysplexWLMPoll: 060  MaxRecs:   100
ExplicitBindPortRange:  00000-00000   IQDMultiWrite: No
Sysplex Monitor:
  TimerSecs: 0060  Recovery: No   DelayJoin: No   AutoRejoin: No
  MonIntf:   No    DynRoute: No
zIIP:
  IPSecurity: Yes  IQDIOMultiWrite: No
```

Figure D-11 A netstat -f command example

The IBM Resource Measurement Facility™ (IBM 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 details of how the zIIP is being utilized by LPARs and address spaces.

The CPC Capacity window in Monitor III shows how the CPU is being used by partitions (see Example D-4).

Example D-4 RMF CPC capacity example

```
RMF V1R12  CPC Capacity                 Line 43 of 53
 Command ===>                                          Scroll ===> CSR

 Samples: 100     System: SC32  Date: 10/13/10  Time: 10.20.00  Range: 100   Sec

 Partition:   A16        2817 Model 715
 CPC Capacity:    1648   Weight % of Max: ****   4h Avg:    2   Group:   N/A
 Image Capacity:   439   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

 A2F                     NO   1.0        0.4     0.4    0.0     0.1    0.1
 PHYSICAL                                           0.6            0.6

 *IIP                         8.0                   0.0     0.0    0.0
 A11                     NO   1.0        0.0     0.0    0.0     0.0    0.0
 A13                     NO   1.0        0.0     0.0    0.0     0.0    0.0
 A16                     NO   1.0        0.0     0.0    0.0     0.0    0.0
 A18                     NO   1.0        0.0     0.0    0.0     0.0    0.0
 A21                     NO   2.0        0.0     0.0    0.0     0.0    0.0
 A22                     NO   2.0        0.0     0.0    0.0     0.0    0.0
```

In Figure D-12, note how the logical partitions are using the zIIP on SC33 and SC32.

The RMF examples (Figure D-12 and Figure D-13 on page 912) show how the zIIP is being used by the address spaces on SC33 and SC32 in the RMF Processor Usage window.

```
 RMF V1R12  Processor Usage                  Line 1 of 5
 Command ===>                                          Scroll ===> CSR

 Samples: 95      System: SC33  Date: 10/13/10  Time: 10.30.00  Range: 100
Sec

             Service    --- Time on CP % ---   ----- EAppl % -----
 Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

 RMFGAT   SO SYSSTC       0.5    0.0    0.0      0.5    0.0    0.0
 XCFAS    S  SYSTEM       0.3    0.0    0.0      0.3    0.0    0.0
 SMSVSAM  S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 WLM      S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 RMF      S  SYSSTC       0.1    0.0    0.0      0.1    0.0    0.0
```

Figure D-12 SC33 RMF Processor Usage example

Figure D-13 shows the SC32 system.

```
 RMF V1R12  Processor Usage                    Line 1 of 6
 Command ===>                                              Scroll ===> CSR

 Samples: 100     System: SC32  Date: 10/13/10  Time: 10.20.00  Range: 100
Sec

             Service    --- Time on CP % ---   ----- EAppl % -----
 Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

 RMFGAT   SO SYSSTC       0.5    0.0    0.0      0.5    0.0    0.0
 XCFAS    S  SYSTEM       0.2    0.0    0.0      0.2    0.0    0.0
 GRS      S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 WLM      S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 RMF      S  SYSSTC       0.1    0.0    0.0      0.1    0.0    0.0
 CSO6     T  SYSOTHER     0.1    0.0    0.0      0.1    0.0    0.0
```

Figure D-13 SC32 RMF Processor Usage example

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 off the zIIP on SC32 and see how the system will behave (see Figure D-14 and Figure D-15 on page 913).

```
 RMF V1R12  Processor Usage                    Line 1 of 7
 Command ===>                                              Scroll ===> CSR

 Samples: 90      System: SC32  Date: 10/13/10  Time: 10.40.00  Range: 100
Sec

             Service    --- Time on CP % ---   ----- EAppl % -----
 Jobname  CX Class      Total    AAP    IIP       CP    AAP    IIP

 RMFGAT   SO SYSSTC       0.5    0.0    0.0      0.5    0.0    0.0
 XCFAS    S  SYSTEM       0.3    0.0    0.0      0.3    0.0    0.0
 *MASTER* S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 WLM      S  SYSTEM       0.1    0.0    0.0      0.1    0.0    0.0
 RMF      S  SYSSTC       0.1    0.0    0.0      0.1    0.0    0.0
 CSO6     T  SYSOTHER     0.1    0.0    0.0      0.1    0.0    0.0
 OMPA     SO SYSSTC       0.1    0.0    0.0      0.1    0.0    0.0
```

Figure D-14 SC32 RMF Processor Usage Example 2

```
 RMF V1R12  CPC Capacity                  Line 1 of 53
 Command ===>                                          Scroll ===> CSR

 Samples: 90      System: SC32  Date: 10/13/10  Time: 10.40.00  Range: 100
Sec

 Partition:   A16        2817 Model 715
 CPC Capacity:    1648   Weight % of Max: ****   4h Avg:    2   Group:   N/A
 Image Capacity:   439   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                        43.0                       0.6     8.6    9.2
 A0B            0     0  NO   1.0       0.0     0.0    0.0     0.0    0.0
 A01            0     3  NO   2.0       1.3     1.4    0.0     0.2    0.2
 A02            0     1  NO   2.0       0.2     0.3    0.0     0.0    0.0
 A03            0     3  NO   2.0       1.2     1.3    0.0     0.2    0.2
 A04            0     2  NO   2.0       0.7     0.7    0.0     0.1    0.1
 A05            0   113  NO   2.0      51.3    51.3    0.0     6.8    6.8
 A06            0     2  NO   2.0       0.7     0.7    0.0     0.1    0.1
 A07            0     1  NO   1.0       1.1     1.1    0.0     0.1    0.1
 A1B            0     0  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, see the Capacity Planning for zIIP Assisted IPSec website:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100988

zIIP performance projection

If you are running IPSec, zIIP might significantly reduce the CPU utilization of your standard CPs. In planning for zIIP, you need to determine the following information:

- 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, traffic modeling might 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 as follows:

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).

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             CSO3      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

1 service class is defined in this workload.

    * Service Class IPSECCL - IPSec traffic service class

      Created by user CSO3 on 2007/10/01 at 11:56:39
      Base last updated by user CSO3 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 (shown in Example D-8) from Example D-5 on page 914.

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 910, first install the definitions on the WLM couple data set, and then activate the service policy:

   a. Select **Utilities** from the top of your ISPF window.
   b. Select **1 Install definition**, and then press Enter.
   c. Select **3 Activate service policy**, and 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 shown in 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=1600,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, 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                   SYSPLEX COMMPLEX          DATE 10/01/2007          INTERVAL 19.59.994   MODE = GOAL
                           RPT VERSION 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 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 916. 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                    SYSPLEX COMMPLEX          DATE 10/01/2007          INTERVAL 19.59.923   MODE = GOAL
                            RPT VERSION 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               no 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.

Tip: For more information, see the Capacity Planning for zIIP Assisted IPSec website:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100988

The difference between APPL% in the system running without zIIP and the system running with zIIP is because of the Large System Performance Report (LSPR).

- APPL% without zIIP = 8.74
- APPL% with zIIP = 8.54

The following formula is used to calculate this value:

$$\text{APPL\% CP} = \frac{\text{CPU} + \text{SRB} + \text{RCT} + \text{IIT} + \text{HST} - \text{AAP} - \text{IIP}}{\text{Interval length}} * 100$$

For more information about RMF, see the *RMF User’s Guide,* SC33-7990.

E

z/OS Communications Server IPSec RFC currency

The RFC standardization is a lengthy process. Many RFC standardization processes can take years. An RFC includes the following stages:

- Proposed Standard
- Draft Standard
- Standard

An RFC is considered an approved RFC when it is in the *Standard* status.

Because of this lengthy process, the market cannot always follow or wait for an RFC. Many current standards were in *Draft Standard* status when these products were developed and marketed. The *Draft Standard* is considered the final specification, but changes are likely, primarily changes to workaround issues that surface during the implementation or testing of the RFC.

Another cause for incompatibility between the product and the RFC is the introduction of new functions in a newly developed RFC.

Table E-1 lists RFCs that are relevant to z/OS Communications Server. The column on the right includes the status of implementation of each RFC in CS.

Table E-1 z/OS Communications Server

RFC	Topic	DoD Adv UNIX Server Profile	NIST Host Profile	z/OS status of V1R12
3041	Privacy Extensions for Stateless Address Auto config in IPV6	SHOULD+	SHOULD	Not implemented
3566	The AES-XCBC-MAC-96 Algorithm and its use with IPSec	SHOULD+	SHOULD+	Compliant
3566	AES-XCBC-MAC-96	N/A	SHOULD+	Compliant
3566	Suite VPN-B, as defined in RFC 4308	SHOULD+	SHOULD+	Compliant

RFC	Topic	DoD Adv UNIX Server Profile	NIST Host Profile	z/OS status of V1R12
3664	The AES-XCBC-MAC-128 Algorithm for the Internet Key Exchange protocol (IKE)	SHOULD+	SHOULD+	Compliant
3664	Suite “VPN-B, as defined in RFC 4308.”	SHOULD+	SHOULD+	Not implemented
3686	Using Advanced Encryption (AES) Counter Mode with IPSec Encapsulating Security Payload (ESP)	N/A	SHOULD	Not implemented
3948	UDP encapsulation of ESP packets	N/A	MAY	Compliant
3971	SEcure Neighbor Discovery (SEND)	SHOULD+	SHOULD+	Not implemented
3972	Cryptographically Generated Address (CGAs)	SHOULD+	SHOULD+	Not implemented
4109	Algorithm for IKE Version 1 (IKEv1) AES-128 in XCBC mode for PRF functions (RFC 3566 and (RFC 3664) SHOULD be supported	N/A	N/A	Not implemented
4301	Security Architecture for the Internet Protocol	MUST	MUST	Compliant
4302	IP Authentication Header (AH)	MUST	MAY	Compliant
4303	IP Encapsulation Security Payload (ESP)	MUST	MUST+	Compliant
4304	Extended Sequence Number (ESN)	SHOULD	MUST	Compliant
4305	Crytographic algorithm implementation requirements for ESP and AH ► AES-CTR (RFC3686) ► AES-XCBC-MAC-96 (RFC3566)	MUST N/A N/A	SHOULD SHOULD SHOULD+	Compliant Not implemented Compliant
4307	Cryptographic algorithm for use in the Internet Key Exchange version 2 (IKEv2)	SHOULD+	SHOULD+	Compliant
4308	Cryptographic suites for IPSec ► Suite “VPN-B” (includes RFCs 3566 and 3664)	MUST SHOULD+	MAY SHOULD+	Compliant
5996	Internet Key Exchange version 2 (IKEv2)	SHOULD+	SHOULD+	Compliant

RFC 4301 *Security Architecture for the Internet Protocol* specifies the base architecture for IPSec-compliant systems, including restrictions on the routing of fragmented packets. RFC4301 compliance enforcement is an optional setting in the z/OS IPSec policy, but it might become a required setting in a future release. A migration health check that is introduced as a new function APAR prepares for the eventual mandatory compliance with RFC 4301 by warning if the systems are not yet compliant.

F

Our implementation environment

We wrote the four *IBM z/OS Communications Server TCP/IP Implementation* books at the same time. Given the complexity of this project, we needed to be 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 that we used for the four books and the environment that we used for this book.

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 F-1.

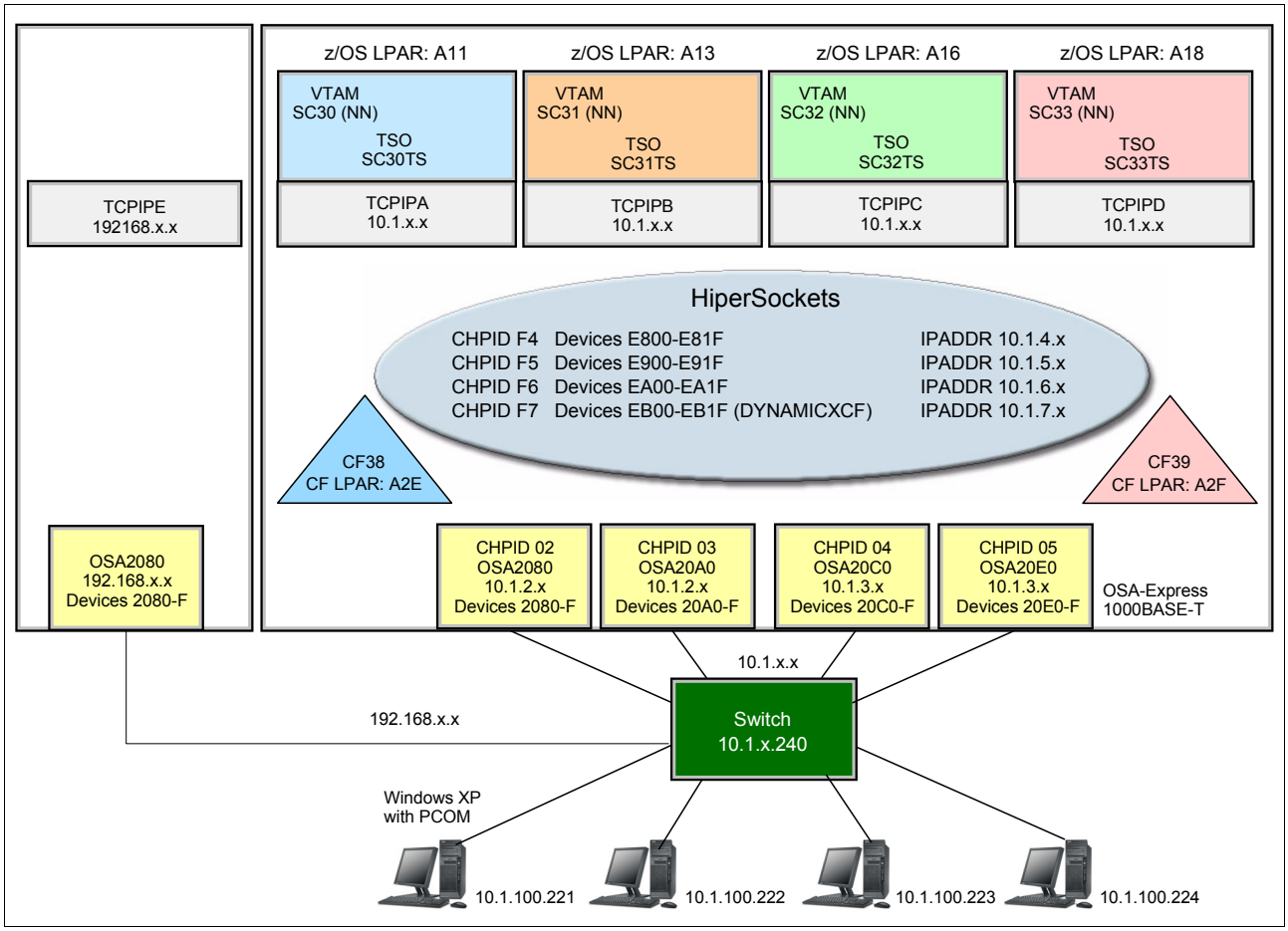


Figure F-1 Our implementation environment

We wrote our books (and ran our implementation scenarios) using four logical partitions (LPARs) on an IBM System z196-32 (referred to as LPARs: A11, A13, A16, and A18). 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 IBM Parallel Sysplex® scenarios
- Eight OSA-Express3 1000BASE-T Ethernet ports connected to a switch

Finally, we shared four Windows workstations, representing corporate network access to the z/OS networking environment. The workstations are connected to the switch. For verifying our scenarios, we used applications such as TN3270 and FTP.

The IP addressing scheme that we 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 (Figure F-2).

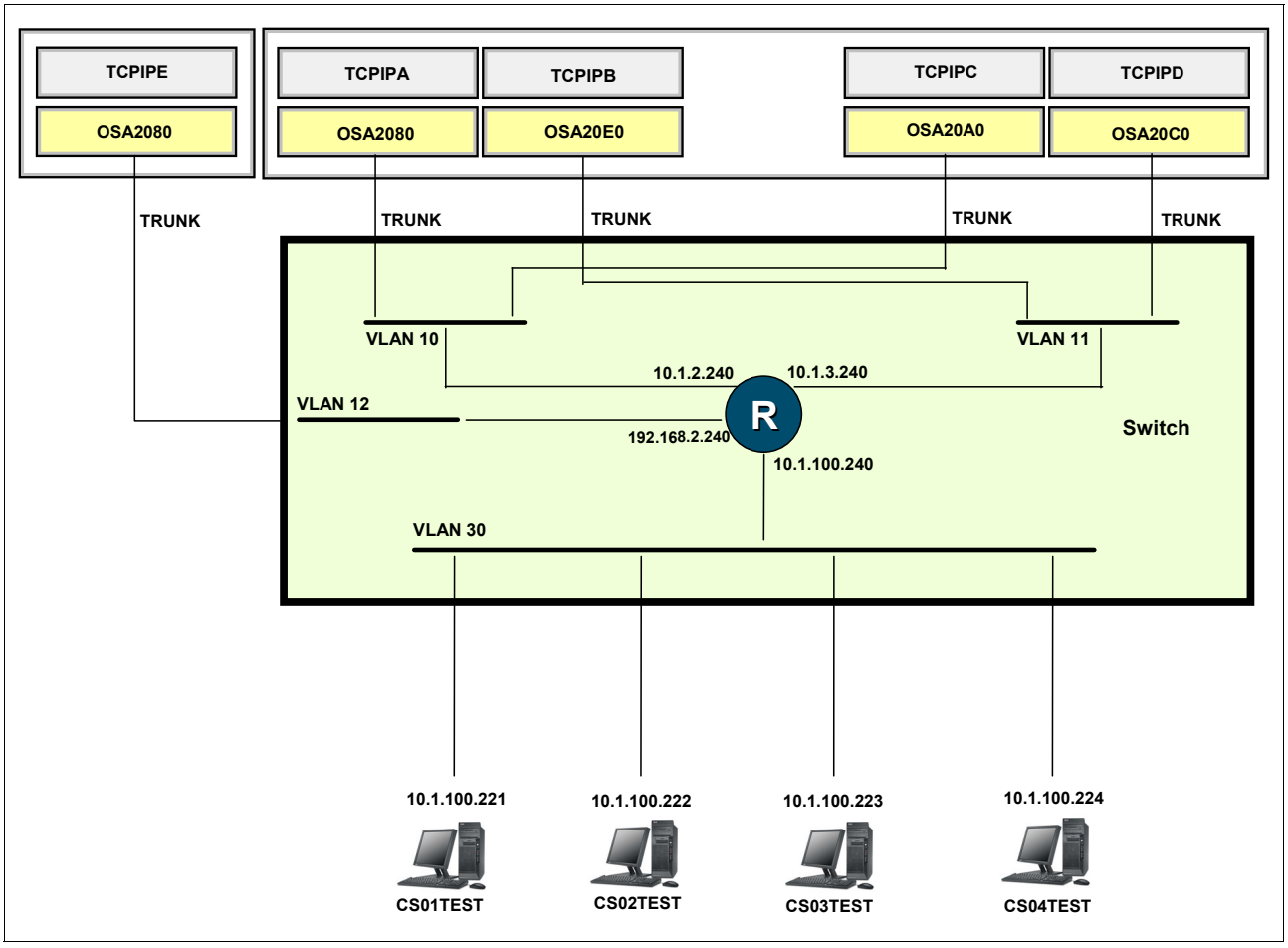


Figure F-2 LAN configuration: VLAN and IP addressing

Our focus for this book

Figure F-3 depicts the environment that we worked with, as required for our security and policy-based implementation scenarios. To prevent interference with other activities, certain 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.

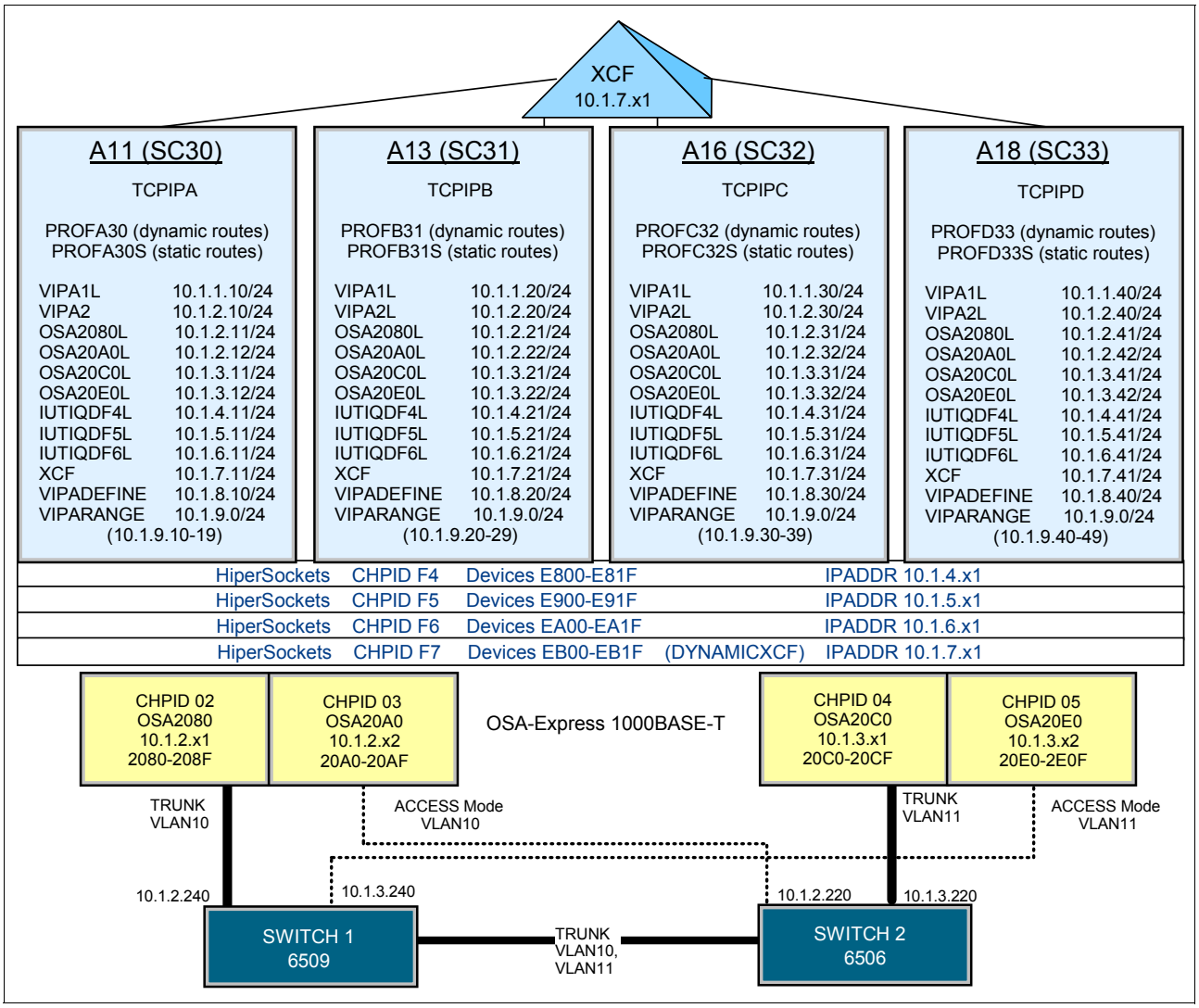


Figure F-3 Our environment for this book

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 publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 928. Note that some of the documents that we reference here might be available in softcopy only.

- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7996
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7997
- *IBM z/OS V1R13 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7998
- *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074
- *HiperSockets Implementation Guide*, SG24-6816
- *Implementing PKI Services on z/OS*, SG24-6968
- *IMS in the Parallel Sysplex Volume II: Planning the IMSplex*, SG24-6928
- *IP Network Design Guide*, SG24-2580
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *OSA-Express Implementation Guide*, SG24-5948
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *z/OS 1.6 Security Services Update*, SG24-6448
- *z/OS Infoprint Server Implementation*, SG24-6234

Other publications

The following publications are also relevant as further information sources:

- *Enterprise Systems Architecture/390 Reference Summary*, SA22-7209
- *z/OS Migration*, GA22-7499
- *z/OS UNIX System Services Planning*, GA22-7800
- *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 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 XL C/C++ Run-Time Library Reference*, SA22-7821
- *OSA-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: New Function Summary*, GC31-8771
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: SNA Operation*, 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://publib.boulder.ibm.com/tividd/td/TRM/GC32-1323-00/en_US/HTML/admin18.htm

- *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/ssl/ssl-information-center/enterprise-ssl-certificates/index.html

How to get IBM Redbooks publications

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 website:

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

X

Z

Redbooks

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

IBM z/OS V1R13 Communications Server TCP/IP Implementation: Volume 4 Security and Policy-Based Networking

Redbooks®

Describes z/OS Communications Server TCP/IP security and policy capabilities

Shows how to protect your z/OS networking environment

Includes security and policy implementation examples

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

This IBM Redbooks publication explains how to set up security for the z/OS networking environment. Network security requirements have become more stringent and complex. Because many transactions come from unknown users and untrusted networks, careful attention must be given to host and user authentication, data privacy, data origin authentication, and data integrity. We also include helpful tutorial information in the appendixes of this book because security technologies can be quite complex.

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

SG24-7999-00 ISBN 0738436585