IBM

Sysplex eBusiness Security z/OS V1R7 Update

Exploiting the sysplex advantages in an exposed environment

Showing the newest z/OS V1R7 Security features at work

The many things to take into consideration

Patrick Kappeler
Giancarlo Rodolfi
Kristen Donceel
Matt Nuttall
Ian Hollamby
Jean-Marc Darees

Redbooks

ibm.com/redbooks

International Technical Support Organization

Sysplex eBusiness Security z/OS V1R7 Update

September 2006

SG24-7150-00

Note: Before using this information and the product it supports, read the information in "Notices" on page ix.

First Edition (September 2006)

This edition applies to Version 1 Release 7 of z/OS (product number 5694-A01).

Contents

Notices ix
Trademarks x

Preface xi
The team that wrote this redbook. xi
Become a published author xii
Comments welcome. xiii

Chapter 1. Introduction. 1
1.1 Security 2
1.1.1 Implementing the security mechanisms 3
1.2 How this book has been structured 4
1.3 z/OS V1R7 Security components and APIs 5
1.3.1 Packaging of the z/OS Security functions and APIs 5
1.3.2 z/OS Security Level 3 optional feature 10
1.3.3 Security functions in the z/OS Communications Server 11
1.3.4 Communications Server Security Level 3 optional feature 12
1.3.5 Additional product - OpenSSH for z/OS 12

Part 1. Basic Parallel Sysplex security 15

Chapter 2. Protection of the Sysplex-specific resources 17
2.1 Our Sysplex model 18
2.2 Resource-sharing and data-sharing in a Sysplex 21
2.2.1 Resource sharing configuration 21
2.2.2 Data sharing 27
2.3 Protection of the Sysplex-specific resources. 27
2.3.1 Protection of the Coupling Facility structures 28
2.3.2 Protection of Sysplex couple datasets 30
2.3.3 The protection of the System logger resources 32
2.3.4 Controlling the use of Automatic Restart Management. 37
2.3.5 Sysplex operator command protection 40
2.4 An overview of Multilevel Security (MLS) 46
2.4.1 Multilevel security 46
2.4.2 Why multilevel security 47
2.4.3 Access Controls 47
2.4.4 Introduction to Mandatory Access Control 48
2.4.5 Multilevel security in z/OS with RACF. 54
2.4.6 Before turning on multilevel security 56

2.4.7 Multilevel security vocabulary 56
2.4.8 Multilevel Security and Sysplex. 57

Chapter 3. UNIX System Services Security 59
3.1 The MVS and UNIX UID dual identity 60
3.1.1 Shared UID prevention 63
3.1.2 Automatic UID/GID assignment 64
3.2 z/OS UNIX Security. 66
3.2.1 BPX.DAEMON FACILITY 66
3.2.2 RACF enhanced program security, BPX.MAINCHECK 68
3.2.3 More on UIDs and UID switching 69
3.3 Limiting the amount of superusers 71
3.4 z/OS UNIX file systems: HFS and zFS 74
3.4.1 HFS basics 75
3.4.2 zFS basics. 78
3.5 Access permission to UNIX files and directories. 80
3.6 HFS Access Control Lists (ACLs) 84
3.6.1 ACLs and RACF 86
3.6.2 ACL auditing and reporting 88
3.7 z/OS UNIX System Services and Multilevel Security 88
3.7.1 Associating security labels with remote users. 89
3.7.2 Assigning a home directory and initial program depending on security label 89
3.7.3 Security labels for z/OS UNIX files and directories 90
3.7.4 Assumed security labels 91
3.7.5 Security labels for zFS file systems and their contents. 91
3.7.6 Security labels for HFS file systems 92
3.8 The Sysplex shared file system. 93
3.8.1 Procedures for establishing a shared file system in a sysplex 94
3.8.2 The Shared HFS structure 94
3.8.3 Security considerations with the shared HFS implementation 97
3.8.4 The AUTOMOVE function. 99
3.8.5 Root isolation 103
3.9 Additional BPX.* FACILITY class profile 103

Chapter 4. Sysplex Workload Management and Security 107
4.1 Basic concepts of WLM. 108
4.2 Limiting resources consumption with WLM. 111
4.1 HMC 114
4.1.1 HMC Physical Access Security. 115
4.1.2 Logical Access Security 115
4.1.3 Network Access Security 115
4.2 Sysplex Timer 118

4.2.1 Sysplex Timer (9037-002) 118
4.2.2 Server Time Protocol 119
4.1 Background 122
4.1.1 Sysplex support prior to ICSF HCR7730 122
4.2 Sysplex support with ICSF HCR7730 125
4.2.1 How the new CKDS Sysplex sharing works 125

Part 2. One Sysplex member with network connectivity 129

Chapter 5. Protecting the network connection 131
5.1 General discussion on non-secure network threats 132
5.2 TCP/IP oriented additional RACF protections 133
5.2.1 TCP/IP resources as protected by RACF 134
5.3 UNIX System Services security considerations 138
5.3.1 AUTOMOVE of the shared HFS owner 138
5.4 Considerations on WLM participation to Security 139
5.4.1 WLM and the TCP/IP address space 140
5.4.2 How WLM can limit the resources used by TCP/IP 140
5.4.3 Calculation of the usage limit values 142
5.5 SYSLOGD protection considerations 142
5.6 Overview of z/OS network level security 145
5.6.1 Protection from network intrusions 145
5.6.2 Controlling the contents of communications 145
5.6.3 Transport Layer Security and Virtual Private Networks 145
5.7 Controlling TCP/IP address space storage consumption 146

Chapter 6. Security at the network level 147
6.1 z/OS TCP/IP stack hardening 148
6.2 IP security filtering 149
6.2.1 The relevance of IP filtering in our configuration 149
6.2.2 IP Filtering concepts 150
6.3 Protecting the data on the wire 153
6.3.1 Introduction to IPSec VPNs and AT-TLS 153
6.3.2 z/OS V1R7 Integrated IP Security as compared to the z/OS Firewall Technologies 154
6.4 The z/OS Communications Server Policy Agent 157
6.5 Virtual Private Network (VPN) 159
6.5.1 ISAKMP and IKE 160
6.5.2 Security Associations (SA) and VPN 161
6.5.3 VPNs and certificates 163
6.6 Application Transparent TLS (AT-TLS) 164
6.6.1 AT-TLS concepts 165
6.6.2 AT-TLS z/OS implementation 166
6.7 z/OS Intrusion Detection Services 168

6.7.1 Intrusion detection overview 168
6.7.2 Network-based intrusion detection 169
6.7.3 Host-based intrusion detection 169
6.7.4 The z/OS Intrusion Detection Services 170
6.7.5 The z/OS IDS policy 171
6.7.6 TRMD 173

Part 3. All Sysplex members with network connectivity 175

Chapter 7. All Sysplex members with network connectivity 177
7.1 VIPA and Sysplex Distributor 178
7.1.1 Virtual IP Address (VIPA) 178
7.1.2 TCP/IP stacks participation in a Sysplex configuration 181
7.2 IP filtering considerations 182
7.3 VPN considerations. 185
7.3.1 Sysplex-wide security associations. 185
7.4 System SSL and AT-TLS considerations 186
7.5 Sysplex session ID caching. 187
7.5.1 z/OS Sysplex session-ID caching support 191
7.6 External load balancing security considerations 193

Chapter 8. Miscellaneous network-related considerations 195
8.1 The DeMilitarized zone (DMZ) 196
8.1.1 The z/OS TCP/IP stack in the DMZ 197
8.1.2 The dual-stack configuration. 198
8.1.3 Establishing a stack affinity. 199
8.2 Applicability of the DMZ principle to Parallel Sysplex 202
8.2.1 Words of caution - XCF Messaging 202
8.2.2 Words of caution - RACF user revocation. 204
8.2.3 Words of caution - Sysplex Distributor 204

Appendix A. RACF protection of MVS commands 209

Appendix B. TCP/IP configuration information 215
Basic configuration of the policy agent. 216
More details. 217
Sharing files 217
A quick look around - pasearch and ipsec 218

Appendix C. IP filtering implementation and management example . . . 221
z/OS IP filtering implementation. 222
IP filtering management 230
Using the ipsec command 232
Using the pasearch command 240

IP filtering logging considerations. 241

Appendix D. IP filtering and Sysplex Distributor 243

Appendix E. AT-TLS implementation 253
Creating the configuration file. 254

Appendix F. Sysplex session-ID caching setup example 257
Sysplex session-ID caching configuration example 258
Sysplex session-ID caching trace analysis example 262

Appendix G. VPN setup 269
VPN scenario 270
z/OS Network Configuration Assistant 270
TCP/IP profile statements generated by the Assistant 272
Policies generated by the Configuration Assistant 272
Setup of the IKE daemon 278
Setting up the digital certificates. 279
Windows XP setup 280
Certificate and VPN management. 280
Starting the VPN at the z/OS end 282
After we know that the VPN is operating. 288
VPN and filters in a Sysplex Distributor configuration 288
z/OS setup 289
Windows XP client changes for the Sysplex Distributor configuration . . . 299
Proving Sysplex-Wide Security Association (SWSA) is working. 299

Related publications 303
IBM Redbooks 303
Other publications 303
Online resources 304
How to get IBM Redbooks 304
Help from IBM 304

Index 305

Notices

Trademarks

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

Redbooks (logo) ™	IBM®	S/390®
z/OS®	IMS™	Sysplex Timer®
zSeries®	MVS™	System z™
z9™	MVS/ESA™	System z9™
CICS®	OS/2®	Tivoli®
DB2®	OS/390®	VTAM®
DRDA®	Parallel Sysplex®	WebSphere®
ESCON®	Redbooks™	
HiperSockets™	RACF®	

The following terms are trademarks of other companies:

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

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

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

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

Preface

This IBM® Redbook provides an overview of the z/OS® Security setups for Parallel Sysplex® installations that are considering serving users locally or over non-secure TCP/IP networks. It provides insight into what can be done to minimize the risks in such contexts by addressing the following operating environments:

- ► Parallel Sysplex (as a stand-alone system) security.
- ► One member of the Sysplex is exposed to a non-secure network.
- ► All members of the Sysplex can be reached from the non-secure network.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the Montpellier European Products and Solutions Support Center (PSSC) on behalf of the International Technical Support Organization, Poughkeepsie Center.

Patrick Kappeler led this project. For the past 35 years he has held many international specialist and management positions in IBM, all dealing with mainframe technical support. He is now part of the European Products and Solutions Support Center, located in Montpellier, France, where his area of expertise is the ebusiness Security on System z™. He has authored many IBM Redbooks™ and extensively writes and presents on this topic.

Giancarlo Rodolfi is a zSeries® Certified Consultant TSS in Brazil. He has 19 years of experience in the zSeries field. His areas of expertise include zSeries and Linux®. He has written extensively on the z/OS Communication Server.

Kristen Donceel is a zSeries System Architect from Belgium. She has 17 years of experience with IBM Large Systems, and most of the years she worked as a Pre-sales Technical Support Specialist for zSeries. Her areas of expertise include zSeries hardware (especially Parallel Sysplex).

Jean-Marc Darees joined IBM in 1984 as an MVS™ System Engineer. Since then he has held several specialist and architect positions dealing with mainframe and other technologies supporting customers and IBM internal projects. He joined the PSSC in Montpellier in 1997, where he now provides consulting and pre-sales technical support in the area of Siebel CRM infrastructure for large customers.

Matt Nuttall is a TCP/IP and network security specialist in IBM Canada. He has 16 years of experience working with z/OS. He holds a degree in Computer Science from the University of British Columbia. He has written extensively on networking and security.

Ian Hollamby is a RACF® Security Specialist in Australia. He has more than 10 years of experience in IBM supporting S/390®-based customers. He currently works for the IBM Australia Security support team from Adelaide.

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

Chris Ryans
International Technical Support Organization, Poughkeepsie Center

Frank Injey
International Technical Support Organization, Poughkeepsie Center

Eran Yona
International Technical Support Organization, Poughkeepsie Center

Alain Roger
IBM European PSSC - PC hardware and software setup

Pascal Marachian
IBM European PSSC- Network support

Jean Jacques Noguera
IBM European PSSC - z/OS ESP Coordinator and System Engineer

Pierre Cassier
IBM European PSSC - Sysplex performance

Linwood H. Overby Jr
STSM Enterprise Networking Solutions - Raleigh

Mark W. McClintock
TCP/IP Development - Raleigh

David A. Herr
TCP/IP Development - Raleigh

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@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

1

Introduction

This book is a follow-on to *Security Configuration in a TCP/IP Sysplex Environment*, SG24-6527, which was produced during an ITSO residency run on a z/OS V1R2 sysplex, and was updated, at edition time, with additional information pertaining to z/OS V1R3 and V1R4 contents.

This follow-on has been produced using a sysplex running z/OS V1R7 and contains a preview of z/OS V1R8 contents.

1.1 Security

For the sake of practicality we directly refer to the IBM Security Architecture, built upon the well-known definition of security as provided by ISO Standard 7498-2. ISO developed this standard in the context of an OSI model for a systematic approach to network security. The IBM Security Architecture is a model for integrating security services, mechanisms, objects, and management functions across multiple hardware and software platforms and networks. The architecture supports the strategy for providing end-to-end protection of applications and information within an organization.

We elaborate in the following chapters on this very general statement to illustrate what the objectives are that one wants to achieve to ensure security, how that matches with z/OS-provided security mechanisms, and how this fits with what we want to protect when operating a Parallel Sysplex and connecting it to a non-secure network.

ISO 7498-2 globally states that security is provided with the following basic mechanisms:

- Identification and authentication of users and communicating entities.
- Access control, meaning selectively allowing or denying authenticated users/entities access to resources.
- Data confidentiality, ensuring that data is exposed only to the people/entities you designate.
- Data integrity, ensuring that stored or transmitted data content has not been subject to unauthorized modifications.
- Non-repudiation, the capability to provide an undeniable proof that a transaction occurred.

Although considered to address a networked environment, and also involving the use of the cryptography to provide maximum security, these mechanisms also apply to the traditional view of security that one has when considering the software platform or operating system. It is required today that users do identify and authenticate themselves to the operating system. Then the access control functions make a decision based on this established user identity. The validity of these decisions relies on the integrity of the operating system code. Therefore we will keep this ISO 7498-2 definition of security in focus throughout this book.

New threats

We assume that the reader is already somehow acquainted with the platform security implemented on MVS and its successors: OS/390® and z/OS, and its objectives. However, when it comes to establishing a connection with a non-secure network, the core system becomes exposed to new kinds of threats that we can roughly summarize as:

- Attempts to acquire undue privileges by impersonating a legitimate user. That would imply stealing first the user identification and authentication data, presumably while it is travelling over the non-secure network. Getting the related privileges may further allow the possibility of stealing, replacing, or compromising the installation’s data or programs.
- Attempt to render the installation non-available to the network users. This is the so-called *denial of service attack*, where the objective is not to steal data or to get undue privileges but rather to prevent legitimate users from getting any services from the installation. A denial of service attack can be performed by:
  - Overloading the network and the system with spurious service requests.
  - Compromising the system’s code or functional data so that the system itself is not operating as it should. In low-end platforms this can be achieved by exploiting known software weaknesses (such as the well-known *buffer overflow* exposure). For more robust platforms, that would imply in a first pass to get the undue privileges, as mentioned above, in order to be able to modify the system’s code or functional data.

1.1.1 Implementing the security mechanisms

Having seen the objectives of the security mechanisms, the implementation of these mechanisms in an installation occurs at three levels:

- At the platform or operating system level. These mechanisms mainly address the functions of identification and authentication of users, access control, and system code integrity. A typical example of the platform security implementation in z/OS is the use of the SAF interface and the RACF external security manager.
- At the network level. These mechanisms enforce an access control policy to the networks an installation is connected to. They also address transmitted data confidentiality and integrity.

Note: These mechanisms initially designed to only permit or deny resource access are getting more proactive in that they can now detect suspicious network activities before the resource access itself is performed. Examples of the implementation of these mechanisms in z/OS are the Communications Server integrated IP Security functions and the Intrusion Detection Services.

- At the transaction level. This level of implementation implies providing additional services to the applications so that they can implement their own security paradigms above and beyond the pure platform and network security. Communicating party's strong authentication and data encryption, as performed by the SSL/TLS protocol, is an example of security implementation at this level.

We explain how these implementation levels relate to the z/OS components in a Parallel Sysplex configuration.

1.2 How this book has been structured

We have structured this book in three parts, along the line of increasing Security exposures when operating a sysplex as a stand-alone entity then connecting it to a non-secure network.

Part 1 - Sysplex Security

In this part we address the protection of the Sysplex-specific resources:

- The Coupling Facility structures
- The Couple Data Sets
- The System Logger resources
- The Sysplex operator commands
- The IXCMIAPU utility

Here we assume that the reader is already familiar with the protection mechanisms for the z/OS resources. In this section we concentrate on the basic Sysplex Security setup without being concerned by TCP/IP Security aspects. Part 1 begins at "Basic Parallel Sysplex security" on page 15.

Part 2- Sysplex with one member connected to the non-secure network

In this part we assume that only one member of the sysplex is communicating with a non-secure TCP/IP network. We develop Security considerations and setup recommendations that contribute to insure that only this very member is accessible from the non-secure network — that is, the other members of the sysplex are kept isolated from the non-secure network.

Part 2 begins at "One Sysplex member with network connectivity" on page 129.

Part 3 - Several Sysplex members accessible from the non-secure network

Here we address a typical Sysplex Distributor configuration, where a client on the non-secure network can get an implicit connection to several members of the Sysplex via the Sysplex Distributor technology.

Part 3 begins at “All Sysplex members with network connectivity” on page 175.

1.3 z/OS V1R7 Security components and APIs

In this section we provide an overview of the z/OS Security components and APIs, as per the functional categories they are put in the z/OS delivery package. A summary of what these components do, along with information about their latest updates and relevant remarks, is also given.

Note that all of these components, except for the z/OS Security Server, are provided as base components of z/OS.

1.3.1 Packaging of the z/OS Security functions and APIs

Starting with z/OS V1R5, the z/OS Security functions were repackaged along the following categories.

z/OS Cryptographic Services

Here we will review the z/OS Cyrptographic Services.

ICSF

The Integrated Cryptographic Service Facility is the z/OS component that both drives the zSeries hardware cryptographic coprocessors and provides the IBM Common Cryptography Architecture (CCA) API for the applications or middlewares to invoke the hardware cryptographic services.

The following IBM Redbooks specifically address the hardware coprocessors and ICSF support as provided on the different S/390 and System z models:

- *Exploiting S/390 Hardware Cryptography with Trusted Key Entry*, SG24-5455
- *S/390 Crypto PCI Implementation Guide*, SG24-5942
- *zSeries Crypto Guide Update*, SG24-6870
- *IBM @server zSeries 990 (z990) Cryptography Implementation*, SG24-7070
- *z9-109 Crypto and TKE V5 Update*, SG24-7123

The z/OS reference books pertaining to ICSF are:

- *z/OS ICSF Overview*, SA22-7519
- *z/OS ICSF System Programmer's Guide*, SA22-7520
- *z/OS ICSF Application Programmer's Guide*, SA22-7522
- *z/OS ICSF Administrator's Guide*, SA22-7521

- *z/OS ICSF Messages*, SA22-7523
- *z/OS Trusted Key Entry Workstation User's Guide 2000*, SA22-7524

New Sysplex support in ICSF FMID HCR7730: ICSF HCR7730 is not delivered with z/OS V1R7 but can be downloaded from the IBM downloads Web site. It provides full Sysplex support for the Cryptographic Key Data Set (CKDS). This is explained in Chapter 3, "UNIX System Services Security" on page 59.

OCSF

The Open Cryptographic Service Facility is the OS/390 implementation of the Intel® CDSA (Common Data Security Architecture) API. The last functional update to the component occurred at OS/390 2.10 - z/OS 1.1. It is described in IBM Redbook *OS/390 Security Server 1999 Updates Technical Presentation Guide*, SG24-5627, and *OS/390 Security Server 1999 Updates: Installation Guide*, SG24-5629.

The z/OS reference books pertaining to OCSF are:

- *z/OS Open Cryptographic Services Facility Application Programming*, SC24-5899
- *z/OS Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*, SC24-5900

OCSF can also exploit Trust Policy plug-ins delivered in z/OS. Roughly, a Trust Policy plug-in is a software component that is used to verify the validity of chains of digital certificates. z/OS comes with OCEP and PKITP as possible Trust Policy plug-ins.

Open Cryptographic Enhanced Plug-in (OCEP) allows OCSF to exploit certificates stored in the RACF database. The z/OS reference book pertaining to PKITP is *z/OS Open Cryptographic Enhanced Plug-ins Application Programming*, SC24-5925.

PKITP provides OCSF with additional capabilities regarding the validation of certificates and certificate chains using Certificate Revocation Lists (CRL) residing in an LDAP directory, or, beginning with z/OS V1R7, CRL accessible via the HTTP protocol or certificate revocation status provided via the Online Certificate Status Protocol (OCSP).

The z/OS reference book pertaining to PKITP is *z/OS Cryptographic Services PKI Services Guide and Reference*, SA22-7693.

Note: There is no specific Sysplex function provided by OCSF, OCEP, and PKITP. However, OCSF can use files residing in a shared HFS configuration, or, via OCEP, can exploit data residing in a shared RACF database. OCEP is described further below.

System SSL

System SSL is a base component of z/OS introduced at OS/390 V2R7. It is a generic set of APIs that applications can use to protect TCP/IP sockets communications using the Secure Socket Layer (SSL) and the newer Transaction Layer Security (TLS) protocols. System SSL is steadily updated to extend the protocol capabilities as proposed in existing or new RFCs and to take advantage of new hardware cryptographic coprocessors technology.

The z/OS reference book pertaining to System SSL is *z/OS Cryptographic Services System Secure Sockets Layer Programming*, SC24-5901.

Sysplex support by System SSL: System SSL provides Sysplex support via the Sysplex Session ID Caching function. This function allows an SSL/TLS negotiated data encryption key to be retrieved and shared by all members of the Sysplex, thus avoiding costly re-negotiations (the SSL *handshake*” to occur. SSL Session ID Caching is explained in 7.5, “Sysplex session ID caching” on page 187.

z/OS PKI Services

The z/OS Public Key Infrastructure (PKI) Services is introduced at z/OS 1.3 as a base component of z/OS. It is an industry-class Registration and Certification Authority software that runs on z/OS, exploiting the capability of RACF, or an equivalent product, to protect the Certification Authority RSA private key and using, whenever available, the hardware cryptography services. The z/OS PKI Services are specifically addressed in the IBM Redbook *Implementing PKI Services on z/OS*, SG24-6968.

The z/OS reference book pertaining to the z/OS PKI Services is *z/OS Cryptographic Services PKI Services Guide and Reference*, SA22-7693.

Sysplex support by the z/OS PKI Services: Multiple independent instances of PKI Services, one per Sysplex member, can work in unison sharing the same data sets that contain the certificate requests and the issued or revoked certificates. This sharing of common data is implemented via VSAM record-level sharing (RLS).

z/OS Security Server

Starting with z/OS V1.R5, the z/OS Security Server only contains the Resource Access Control Facility (RACF) product. Note that although RACF is always delivered with z/OS, you must have the proper licensing to be authorized to use it.

Some of the z/OS reference books pertaining to RACF are:

- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS security Server RACF Command Language Reference*, SA22-7687
- *z/OS Security Server RACF Auditor's Guide*, SA22-7684
- *z/OS Security Server RACF Callable Services*, SA22-7691
- *z/OS Security Server RACF Macros and Interfaces*, SA22-7682
- *z/OS Security Server RACF System Programmers Guide*, SA22-7681

A major addition to the RACF Security model has been made with Multilevel Security (MLS) at z/OS V1R6, with roll-back to z/OS V1R3. MLS is based on the use of *security labels* that allow to complement the RACF resource access control mechanisms with the notions of categories and security levels pertaining to the resources and their accessors. MLS is explained in IBM Redbook *z/OS 1.6 Security Services Update*, SG24-6448.

Sysplex support by RACF: RACF has supported the Sysplex configuration since the very beginning with RACF database sharing.

In a sysplex where members use MLS, the use of certain security labels can be limited to certain members of the sysplex (that is, they can work at different security classifications while still sharing the RACF database). MLS is explained at 2.4, "An overview of Multilevel Security (MLS)" on page 46.

z/OS Integrated Security Services

Here we review the z/OS Integrated Security Services.

Firewall technologies

These z/OS integrated firewall protection mechanisms are specifically addressed by the following redbooks:

- *Stay Cool on OS/390: Installing Firewall Technology*, SG24-2046
- *Implementing VPNs in a z/OS Environment*, SG24-6530

The z/OS reference book pertaining to the Firewall Technologies is *z/OS Security Server Firewall Technologies*, SC24-5922

Important: It has been announced that z/OS V1R7 is the last z/OS release providing the z/OS Firewall Technologies. They are replaced by the Communications Server IP Security functions, which comprise IP filtering and IPSec VPNs. The Communications Server IP Security is delivered beginning with z/OS V1R7.

For more details on the Communications Server IP Security functions see below.

LDAP Directory Server

The LDAP Directory Server supports several backends. Its relationship with security is twofold:

- It can host any security data that one enterprise might want to make available in an LDAP directory. In that case the data will be stored in DB2® tables and accessed through the TDBM backend of the server.
- The LDAP server can be used to access USER and GROUP profiles data in the RACF database. This is achieved via the SDBM backend. Note that TDBM and SDBM can coexist on the same LDAP server and that a third kind of DB2 based backend, the GDBM, can be used to externalize changes to RACF USER profiles or TDBM data.

Miscellaneous information about the z/OS LDAP Server can be found in IBM Redbooks:

- *OS/390 Security Server 1999 Updates Technical Presentation Guide*, SG24-5627
- *OS/390 Security Server 1999 Updates: Installation Guide*, SG24-5629
- *Putting the Latest z/OS Security Features to Work*, SG24-6540
- *z/OS 1.6 Security Services Update*, SG24-6448

The z/OS reference book pertaining to the LDAP server and client are:

- *z/OS Integrated Security Services LDAP Server Administration and Use*, SC24-5923
- *z/OS Integrated Security Services LDAP Client Programming*, SC24-5924

Sysplex support by the LDAP Directory Server: The z/OS LDAP Security Server can run in a multi-server environment where server instances are residing in Sysplex members and are exploiting data residing in a shared RACF database (for the LDAP SDBM backend or the RACF Native Authentication feature of the TDBM backend) or shared DB2 tables (for the TDBM or GDBM backends).

Network Authentication Service

The Network Authentication Service is the z/OS implementation of the Kerberos authentication protocol and Key Distribution Center. More information is available in IBM Redbook *Putting the Latest z/OS Security Features to Work*, SG24-6540.

Sysplex support by the z/OS Network Authentication Service: The Network Authentication Service provides the following specific functions in a Sysplex environment:

- Multiple instances of the SKRBKDC Kerberos server application in the Sysplex.
- Sysplex members shared Key Distribution Center (KDC) if the KDC is actually the RACF database. Note that the other KDC implementation option (that is, using HFS files) does not provide KDC sharing in the Sysplex.
- A sysplex credential cache allows delegated credentials created on one member to be used to initialize the security context on another member.

EIM

Enterprise Identity Mapping (EIM) is a set of APIs that provide a way for applications to easily get (from an installation central repository) information to map an installation-wide user identity to a local platform identity. This local identity is intended to eventually be used by the access control mechanism of the local platform. The EIM reference book is *z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference*, SA22-7875.

Sysplex support by EIM: EIM Sysplex support is indirect in that the EIM Domain Controller is an LDAP directory that can run as a z/OS LDAP multi-server in a Sysplex configuration.

1.3.2 z/OS Security Level 3 optional feature

There is a need for some of the z/OS components to abide with the exportation restrictions regarding cryptographic functions. Although the regulation is rather complicated and evolving, regarding the exact definition of those restrictions a rough approach is to say that the z/OS components are allowed to use symmetric algorithms (that is DES, RC2, RC4) with keys less than 128 bits long. There is no length limitation regarding the asymmetric algorithms (RSA, DSA) keys as long as these algorithms are used to encrypt symmetric keys or for digital signature purposes.

The capability of using symmetric keys of at least 128 bits is provided by installing the unpriced feature of z/OS, *z/OS Security Level 3*. The feature must be explicitly ordered (that is where the export control comes to play) and extends the allowed key length for the following z/OS components:

- LDAP 31-bit Client
- LDAP 64-bit Client
- OCSF
- Network Authentication Service
- System SSL, and therefore all z/OS server and client applications that use SSL/TLS for TCP/IP sockets data protection

With the Security level 3 feature installed, these components can use these algorithms:

- Triple-DES, with a key 168-bits long
- RC4, with a key 128-bits long
- AES, with a key 128-bits, 192-bits, or 256-bits long

Further details on the /OS Security Level 3 optional feature can be found in *z/OS and z/OS.e Planning for Installations*, GA22-7504.

1.3.3 Security functions in the z/OS Communications Server

Here we focus on the TCP/IP Security-related functions provided in the z/OS Communications Server. These functions pertain either directly to TCP/IP services provided by the Communications Server, such as FTP, Telnet, and so on, or to more generic functions such as IP filtering, Virtual private Networks, and so on.

The TCP/IP services Security protections consist of:

- FTP protection via SSL/TLS or Kerberos authentication (and optional encryption).
- TN3270 via SSL/TLS.
- Telnet via Kerberos authentication.
- rshd via Kerberos authentication.
- sendmail via SSL/TLS.
- The Open Shortest Path First (OSPF) dynamic routing protocol supports message authentication and message integrity of OSPF routing messages through the use of the OSPF MD5 Authentication security protocol as defined by RFC 2328. This is implemented via the OMPROUTE service of the Communications Server.

- The Communications Server supports DNS at the Version 9.1 of BIND. This level of DNS has built-in security features, with extensions to DNS that provide data integrity and authentication to security-aware resolvers and applications through the use of cryptographic digital signatures. Also, transaction-level authentication, using shared secrets and one way hashing, authenticates dynamic updates as coming from an approved client, or responses as coming from an approved recursive name server.
- SNMPv3 security level. The SNMPv3 framework defines several security functions, such as USM for authentication and privacy, and view-based access control model (VACM), which provides the ability to limit access to different MIB objects on a per-user basis, and the use of authentication and data encryption for privacy.

More generic functions that we describe in this book are the z/OS Communications Server:

- Integrated IP Security, which provides static IP filtering and IPSec VPNs
- Application Transparent TLS (AT-TLS), where the Communications Server does provide SSL/TLS protection for applications not enabled for SSL or TLS
- Intrusion Detection Services (IDS)

Relevant details on the Security features of the z/OS Communications Server can be found in *z/OS Communications Server IP Configuration Guide*, SC31-8775.

1.3.4 Communications Server Security Level 3 optional feature

As for the z/OS Security Level 3 optional feature, this feature extends, in compliance with the exportation restrictions, the length of symmetric cryptography keys that components of the Communications Server use. The following component is affected by the installation of Security Level 3: IP Security, with the IPSec VPNs being able to use the Triple-DES encryption algorithm.

1.3.5 Additional product - OpenSSH for z/OS

OpenSSH for z/OS comes as the *z/OS IBM Ported Tools* product, Program Number 5655-M23. OpenSSH is a suite of popular UNIX® connectivity tools providing secure remote login, remote shell, and FTP between SSH client and server. z/OS IBM Ported Tools is an unpriced product that nevertheless needs to be ordered, and comes as an SMP/E installable program.

The OpenSSH reference book is *z/OS IBM Ported Tools for z/OS User’s Guide*, SA22-7985.

Note: There is no specific direct Sysplex support by OpenSSH for z/OS. However, running OpenSSH can take advantage of running in a Sysplex configuration:

- The OpenSSH configuration files can be shared between members of the sysplex.
- OpenSSH supports Multilevel Security, and as such can use security labels assigned on a per-system basis when running in a Sysplex configuration.

Part 1

Basic Parallel Sysplex security

In this part we address the protection of the sysplex-specific resources in the system and the direct or indirect relationship of some sysplex functions with security.

2

Protection of the Sysplex-specific resources

In this chapter we focus on how to use RACF to protect resources of the sysplex environment. At this stage we consider that the environment we are protecting does not include a connection to a non-secure network.

We describe how RACF can be used to protect:

- The coupling facility structures
- The couple datasets
- The Sysplex operator commands
- The System Logger resources
- The IXCMIAPU utility functions

Besides the IBM RACF reference books (for those installations using RACF), the reference document that we recommend here is *z/OS MVS Setting Up a Sysplex*, SA22-7625.

2.1 Our Sysplex model

Figure 2-1 on page 19 describes our theoretical Sysplex model that we use to stress where resource protection is required:

- MVS and UNIX applications are running under a user identity. They are getting the privileges granted to this user.
- Access to Sysplex resources is controlled on the basis of the accessor's identity and therefore the privileges owned by this identity. This is the case, for example, for controlling access to the Coupling Facility structures, or the Sysplex couple data sets.
- More Sysplex resources that need to be protected are the console commands, including commands with a Sysplex global scope such as the ROUTE command.
- Although not specifically linked to the Sysplex functions, logical and physical security of ancillary devices such as the Hardware Monitor Console (HMC) must be addressed.

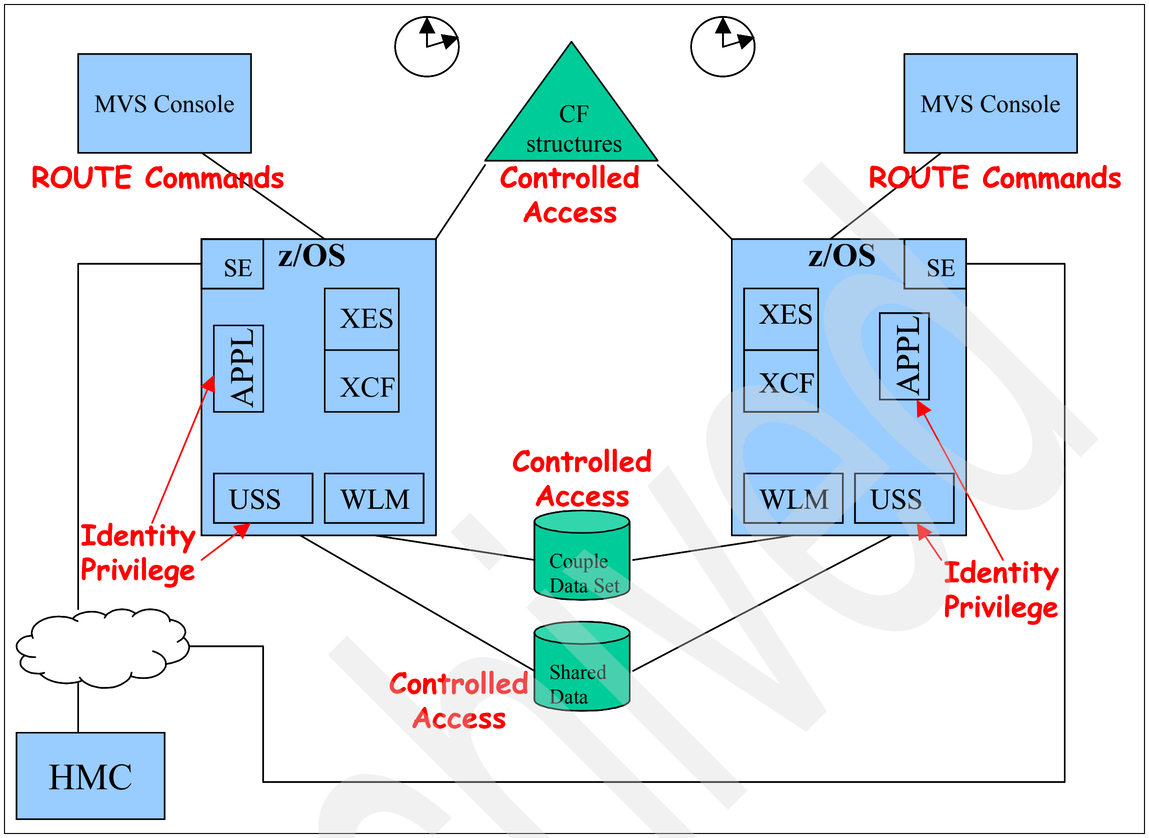


Figure 2-1 Our Sysplex configuration model

Preliminary comments

For this exercise we only concern ourselves with the IBM RACF being used as the external security manager called by the SAF API of the Sysplex members. Other external security manager products are available from vendors as a replacement to RACF, and we do not make any assumption here as to whether they provide functions equivalent to RACF.

ISO 17799

Even though your Sysplex environment may not be open to an external network it is still good practice to ensure that you have adequately protected your current installation from external and internal potential threats and have a good security policy implemented to reduce and deter any current or future threat. A good starting point would be to implement an overall security policy based on ISO 17799. ISO 17799 is a generic Security standard in that it does not describe specific implementations and setups, but rather addresses general Security practices. Its purpose is to be a single reference point regarding Security and Compliance controls to be put in place (in other words, to provide guidelines to run Security assurance evaluation at an installation level). More information about ISO 17799 can be found at:

```
http://www.iso17799software.com
```

Picking up information from the net

A number of other documents you can refer to have been made available on the Internet by various companies and security discussion groups that list the potential Security exposures that might be exploited in a mainframe environment should RACF be improperly set up. We also recommend that your security personnel subscribe to discussion lists such as the RACF-L discussion list.

Customers and IBM participants discuss RACF on the RACF-L discussion list. The list is not operated or sponsored by IBM; it is run by the University of Georgia. To subscribe to the RACF-L discussion and receive postings, send a note to listserv@listserv.uga.edu. Include the following line in the body of the note, substituting your first name and last name, as indicated:

```
subscribe racf-l first_name last_name
```

To post a question or response to RACF-L, send a note, including an appropriate subject line, to racf-l@listserv.uga.edu.

Preliminary guidelines

The areas you want to have locked down in RACF setup, as it applies to our sample sysplex configuration, would be:

- Identification of user IDs accessing the system in both the MVS and the UNIX System Services environments and positive authentication through the enforcement of a tight password policy. z/OS V1R7 provides increased protection by including the ability to use mixed case passwords as well as now providing a configurable minimum mandatory period of days between permitting password changes for the same user ID.
- Control of access to the master and all alternate system consoles, by use of the LOGON keyword in SYS1.PARMLIB(CONSOLxx). This is discussed in "Establishing console security" on page 40.
- Remove all inappropriate access that permits the invocation of system administration resources, such as MVS operator commands (protected by profiles in the RACF class OPERCMDS), privileged TSO commands (the TSOAUTH RACF class), access to various UNIX System Services resources protected by the BPX and profiles in the FACILITY class and profiles in the UNIXPRIV class.

- Ensure that all profiles protecting the Couple Data Sets have limited access and are UACC(NONE).
- Control access to all datasets via specific permissions of user IDs and groups on access lists. Do not expose critical and confidential data via setting UACCs and ID(*) permissions greater than NONE.
- Access lists to your DASD volumes should include specific RACF groups, and not be controlled solely by use of the UACC.

2.2 Resource-sharing and data-sharing in a Sysplex

Furthermore, we can distinguish the use of coupling facility structures as being applied to a resource-sharing or a data-sharing configuration environment, as explained below.

2.2.1 Resource sharing configuration

This environment is characterized as one in which the Coupling Facility technology is exploited by key z/OS components to significantly enhance overall Parallel Sysplex management and operations.

A number of base z/OS components exploit Coupling Facility shared storage, providing an excellent medium for sharing component information for the purpose of multi-image resource management. This exploitation, called IBM zSeries Resource Sharing, enables sharing of physical resources, such as files, tape drives, consoles, catalogs, and so forth, with significant improvements in cost, performance, and simplified systems management. The zSeries Resource Sharing delivers immediate value, even for customers who are not leveraging data sharing, through exploitation delivered with the base z/OS software.

Typical resource sharing structures include:

- XCF
- GRS
- ECS (Catalog)
- JES2 Checkpoint
- RACF
- System Logger

Figure 2-2 shows an overview of a typical resource sharing environment and the potential benefits yielded to the installation.

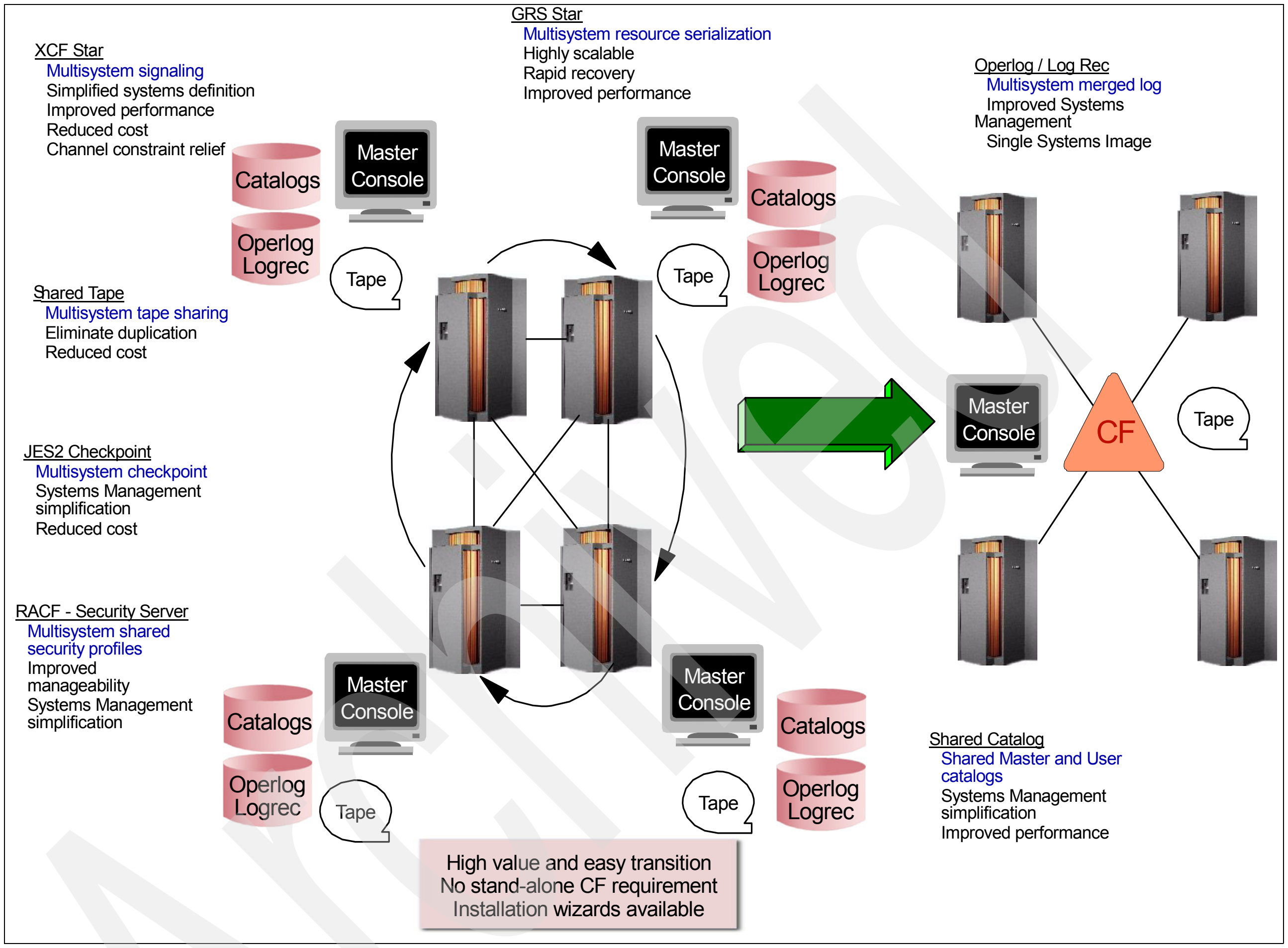


Figure 2-2 Resource sharing functions and values

The connectors to the structures are most often system address spaces such as XCFAS or GRS. In order to set up proper access control for the structures, each connector must be given a specific RACF identity.

Example 2-1 shows how to define XCFAS in the RACF STARTED class and associate a specific user ID with the XCFAS address space. The user ID in this example is XCFUSER.

Example 2-1 Define XCFAS in the RACF STARTED class

```
ADDUSER XCFUSER NAME('XCF USER') DFLTGRP(SUPMVS) OWNER(SUPMVS) NOPASSWORD
RDEFINE STARTED XCFAS.* OWNER(SUPMVS) UACC(READD)   +
        STDATA(USER(XCFUSER) GROUP(SUPMVS) TRUSTED(YES))
SETROPTS RACLIST(STARTED) REFRESH
```

Example 2-2 shows how to protect an XCF structure resource so that the user XCFUSER is granted access. Note that other applications that need to connect to the structure must also be granted access in the RACF profile.

Example 2-2 Define XCF structure protection in FACILITY class

```
RDEFINE FACILITY IXLSTR.IXC2 UACC(NONE) OWNER(SUPMVS)
PERMIT IXLSTR.IXC2 CLASS(FACILITY) ID(XCFUSER) AC(ALTER)
SETROPTS RACLIST(FACILITY) REFRESH
```

RACF database resource sharing

The sharing of the RACF database between members of a sysplex and the implicit exploitation of the Coupling Facility by RACF is considered to be an example of the usage of a structure in the resource sharing environment.

RACF needs first to be enabled for Sysplex communication. A flag in the RACF data set name table (ICHRDSNT) enables the system for sysplex communication. When enabled for sysplex communication, a RACF system can be in one of several modes for accessing the RACF database. The mode is set by another flag in the data set name table. The mode determines whether RACF is to use the coupling facility. Refer to *z/OS Security Server RACF System Programmer's Guide*, SA22-7681, for further information about how to establish Sysplex communication.

When RACF is enabled for Sysplex communication, it uses XCF to join the RACF data sharing group. There is only one RACF data sharing group per sysplex, and it has a fixed name of IRRXCF00.

See Figure 2-3 for an overview of RACF database resource sharing.

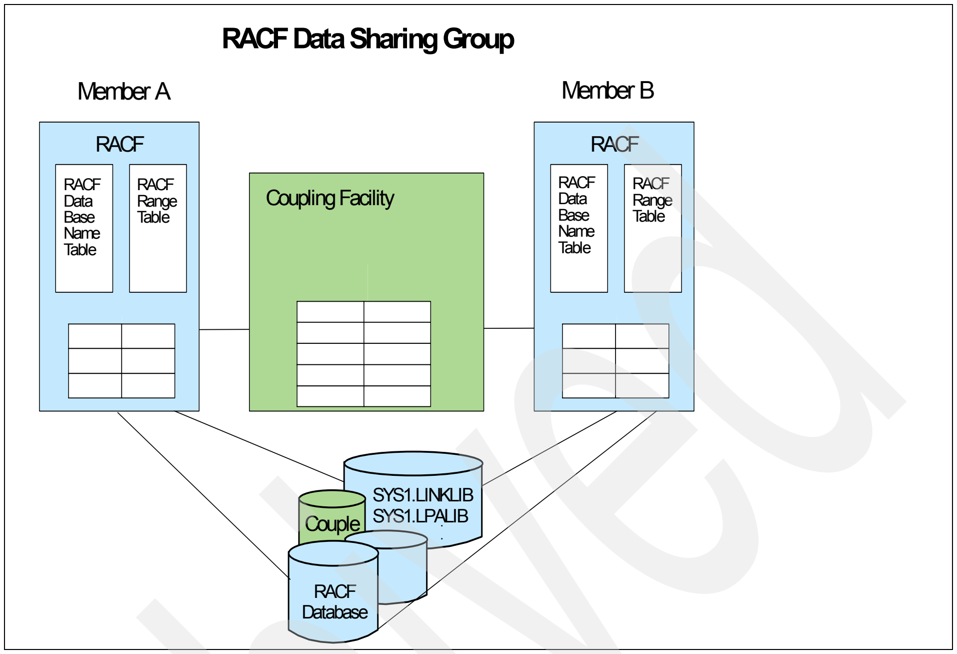


Figure 2-3 RACF data sharing

Note that when RACF has been enabled for Sysplex communication, you have the ability to enter certain commands that now affect the whole sysplex, thereby simplifying security management. The command that can be entered once and then propagate through the rest of the sysplex is the RVARY command, with the following specific parameters:

- RVARY SWITCH
- RVARY ACTIVE
- RVARY INACTIVE
- RVARY DATASHARE
- RVARY NODATASHARE

And also the SETROPTS command with the parameters below:

- SETROPTS RACLIST (classname)
- SETROPTS RACLIST (classname) REFRESH
- SETROPTS NORACLIST (classname)
- SETROPTS GLOBAL (classname)
- SETROPTS GLOBAL (classname) REFRESH
- SETROPTS GENERIC (classname) REFRESH
- SETROPTS WHEN(PROGRAM)
- SETROPTS WHEN(PROGRAM) REFRESH

Figure 2-4 shows the high-level organization of the command propagation scheme. The Sysplex member where the command is entered becomes the coordinating system that makes sure that the commands are propagated to the other members in the Sysplex group.

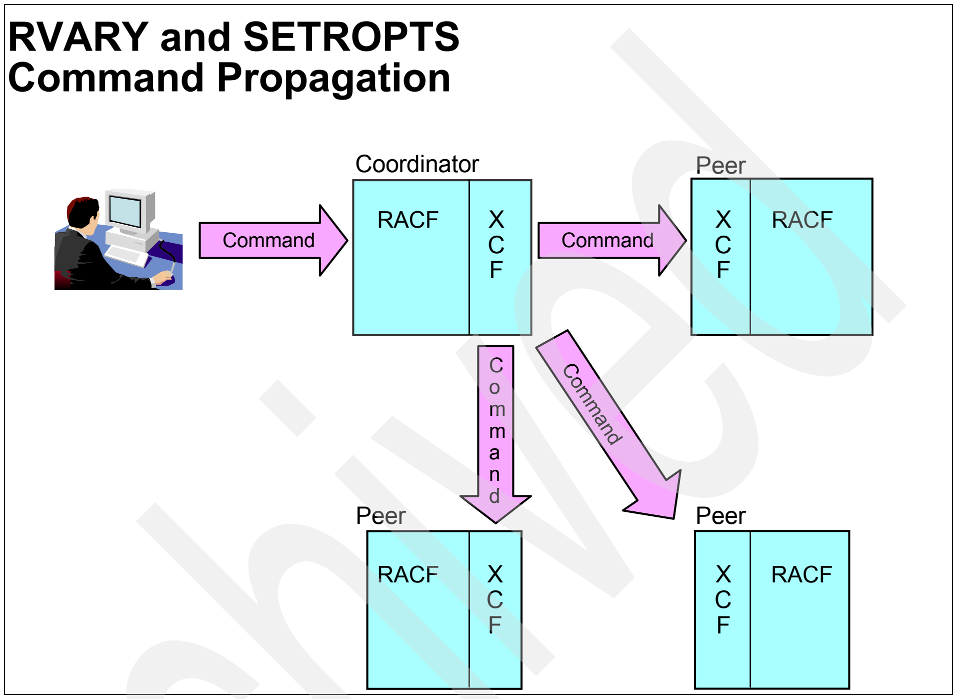


Figure 2-4 RVARY and SETROPTS command propagation

Planning for a coupling facility shutdown with RACF database sharing

Before removing the coupling facility from a Parallel Sysplex configuration, whether for replacement or maintenance, you should put RACF in non-data sharing mode, which automatically deletes the RACF structures on the coupling facility to be removed.

RVARY NODATASHARE is the command necessary to place RACF in non-data sharing mode issued from the coupling facility to be removed.

Note: If the installation has defined RACF as a subsystem in IEFSSNxx of SYS1.PARMLIB, you can issue the command from an MVS operator console, but you need to use the subsystem recognition character that the installation has defined through the command prefix facility. Otherwise, you must issue the command from an authorized TSO/E user ID.

There will be a prompt to the MVS operator console for the RVARY password to be entered before the command will proceed.

When RACF is no longer in data sharing mode, the RACF database cache structures are deleted.

When the coupling facility maintenance is complete and it is back online, RACF data sharing mode needs to be restored to reallocate the structures.

RVARY DATASHARE is the command necessary to restore RACF to data sharing mode.

Advanced considerations on RACF database sharing in a Sysplex

RACF does not have a requirement that all systems in the sysplex use the same RACF database. It does have a requirement that all systems in the sysplex that have RACF using sysplex communications must use the same RACF database.

RACF also has a requirement that all systems using the same RACF database must be in the same sysplex if any of the systems use RACF data sharing mode (that is, via the coupling facility).

The coupling facility used also leads to a requirement that all systems using the same coupling facility, and having RACF in data sharing mode, must have RACF using the same database.

If you have a system using a separate RACF database, then as long as you do not configure it for sysplex communications, theoretically you should be able to have that system in the sysplex. In practice you may find problems. Many products that participate in sysplex may assume that you have the same, or at least compatible, RACF databases contents in all members of the sysplex.

Important: Because work can be distributed around the sysplex, all systems in the sysplex must have the same security information. Therefore, we recommend that all systems in a sysplex use the same RACF database.

Only those systems sharing the same RACF database in sysplex communication will benefit from the RVARY / SETROPTS propagation.

2.2.2 Data sharing

This Parallel Sysplex environment is representative of those customers that have established a functional Parallel Sysplex cluster and have implemented IMS™, DB2, or VSAM/RLS data sharing. In this case the connectors to a Coupling Facility structure are sub-system address spaces like DB2.

Example 2-3 shows the command necessary to protect a DB2 structure.

Example 2-3 Define DB2 structure protection in FACILITY class

```
RDEFINE FACILITY IXLSTR.DSNDBOP_SCA UACC(NONE) OWNER(SUPMVS)
PERMIT IXLSTR.DSNDBOP_SCA CLASS(FACILITY) ID(DBDBOP) AC(ALTER)
SETROPTS REFRESH RACLIST(FACILITY)
```

2.3 Protection of the Sysplex-specific resources

In this section we focus on the RACF setup that is required to protect Sysplex-specific resources. We assume that the reader is already informed, or is going to get informed using his own sources, of the RACF setup required to protect the non sysplex-specific system and applications resources.

2.3.1 Protection of the Coupling Facility structures

One of the basic elements of a sysplex is the structures in the coupling facility. These structures can be accessed by subsystems or applications through the XCF/XES components of z/OS, as shown in Figure 2-5.

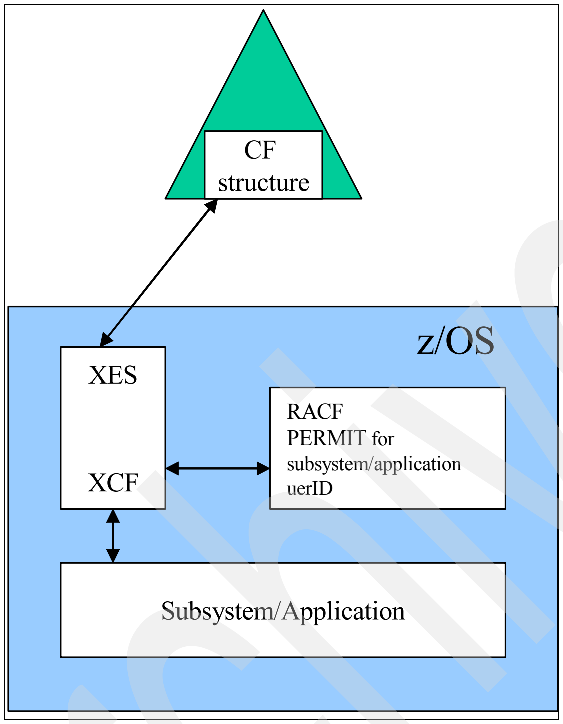


Figure 2-5 Structures Resource Manager

The security administrator might want to control access to data within a structure when requests (such as IXLCONN, IXLREBLD, and IXLFORCE) are issued against this very structure. As these requests are being executed by the XES component of z/OS. XES has been designed to act as a Resource Manager for the Coupling Facility structures and, as such, invokes RACF for controlling access to the target structures on the basis of the requestor's identity.

The following steps describe how the security administrator can define RACF profiles to control the use of Coupling Facility structures:

1. Define a resource profile with a name of IXLSTR.<structure-name> in the FACILITY class.
2. Specify the user IDs of the subsystem or application that require access to the structure using the RACF PERMIT command.

3. Make sure the FACILITY class is active and generic profile checking is in effect. If the FACILITY class is RACLISTed, refresh the class using the SETROPTS RACLIST(FACILITY) REFRESH command.

For example, if an installation wants to permit an application with an identifier of SUBSYS1 to issue the IXLCONN macro for structure-name CACHE1, the security administrator can use the following commands:

```
RDEFINE FACILITY IXLSTR.CACHE1 UACC(NONE)
PERMIT IXLSTR.CACHE1 CLASS(FACILITY) ID(SUBSYS1) ACCESS(ALTER)
SETROPTS RACLIST(FACILITY) REFRESH
```

You can specify RACF user IDs or RACF group IDs on the ID keyword of the PERMIT command. If RACF profiles are not defined for the target structure, the default allows any authorized user or program (supervisor state and PSW Key Mask allowing key 0–7) to issue coupling facility macros for the structure.

If you need to check what structures are defined to your Sysplex you can use the following command to collect pertinent information:

```
d xcf,str
d xcf,str,strnm=structure_name,connm=all
```

Example 2-4 shows sample output from this command.

Example 2-4 D XCF,STR command

```
D XCF,STR
IXC359I  12.01.00  DISPLAY XCF 202
STRNAME            ALLOCATION TIME   STATUS
ATRRRS_ARCHIVE   11/22/2001 14:28:11 ALLOCATED
ATRRRS_DELAYED   11/22/2001 14:28:10 ALLOCATED
ATRRRS_MAIN      11/22/2001 14:28:07 ALLOCATED
ATRRRS_RESTART   11/22/2001 14:28:10 ALLOCATED
ATRRRS_RMDATA    11/16/2001 15:10:56 ALLOCATED
DSNDBOP_GBP0          --        --    NOT ALLOCATED
DSNDBOP_GBP1          --        --    NOT ALLOCATED
DSNDBOP_GBP2          --        --    NOT ALLOCATED
DSNDBOP_GBP3          --        --    NOT ALLOCATED
DSNDBOP_GBP32K        --        --    NOT ALLOCATED
DSNDBOP_GBP32K1       --        --    NOT ALLOCATED
DSNDBOP_LOCK1    11/15/2001 10:11:58 ALLOCATED
DSNDBOP_SCA      11/15/2001 10:11:56 ALLOCATED
IGWLOCK00             --        --    NOT ALLOCATED
IRRXCF00_B001    12/05/2001 11:57:31 ALLOCATED
IRRXCF00_P001    12/05/2001 11:57:30 ALLOCATED
ISGLOCK          11/22/2001 14:27:16 ALLOCATED
ISTGENERIC       11/14/2001 21:53:50 ALLOCATED
IXC1             11/22/2001 14:27:05 ALLOCATED
IXC2                  --        --    NOT ALLOCATED
```

```
SYSTEM_LOGSTREAM      --        --    NOT ALLOCATED
SYSTEM_OPERLOG   12/11/2001 14:53:09 ALLOCATED
```

2.3.2 Protection of Sysplex couple datasets

In this section we discuss Sysplex couple datasets.

Couple datasets

It is the responsibility of the installation to provide the security environment for the couple data sets. Note that the XCF address space (XCFAS) is to access the couple data sets and it does not need to be explicitly authorized in RACF to do so. Therefore the couple data set resource profiles can be defined with UACC(NONE) in RACF and have further access granted as per the installation needs and policy.

As of the writing of this book, the couple data sets (primary and alternate data sets) are:

- SYSPLEX couple data sets, which mainly maintain the status of the running members
- ARM couple datasets, to hold the Automatic Restart Manager policy
- BPXMCDS couple datasets, to keep track of the shared file system status
- CFRM couple datasets, which provide the Coupling Facility structure specifications
- LOGR couple datasets, to hold the System Logger files specifications
- SFM couple datasets, which specify the sysplex recovery policy
- WLM couple dataset, to hold the WLM policy

The specified data set names in use can be found in SYS1.PARMLIB(COUPLExx) or through the D XCF,COUPLE command.

The couple data sets, except for the SYSPLEX couple data sets, contain policies that have to be shared by the sysplex members. Most of these policies are defined by using the IXCMIAPU utility program, and we explain later how to protect access to it with RACF.

Formatting and installing policies in the couple data sets

Before being used, the couple data sets must be formatted with the IXCL1DSU utility program. Once formatted, policies must be installed, and later maintained, in the ARM, CFRM, LOGR, and SFM couple data sets using the IXCMIAPU utility.

Formatting the couple data sets

There is no specific protection applied to the use of the IXCL1DSU utility beside the regular protections for the involved data sets and optionally protection of the utility program.

Installing or maintaining policies in the couple datasets

Each one of the following policies can be protected from changes made by the IXCMIAPU utility by RACF profiles in the FACILITY class, with a name of MVSADMIN.XCF.<*policy name*>, where <*policy name*> can be:

- ARM - Automatic restart management (See 2.3.4, "Controlling the use of Automatic Restart Management" on page 37.)
- CFRM - Coupling facility resource manager
- LOGR - Logstream management
- SFM - Sysplex failure management

Assign UPDATE access authority to users who must alter or maintain the policy; assign READ access authority to users who require reports on the policy, but who will not change the policy.

Note: These FACILITY class profiles will be used for authority checking on an ACTIVE couple data set. When a couple data set is INACTIVE, the rules for normal data set protection apply.

Example 2-5 shows how you can protect these IXCMIAPU functions.

Example 2-5 Protect IXCMIAPU functions

```
RDEFINE FACILITY MVSADMIN.XCF.ARM UACC(NONE) OWNER(SUPMVS)
RDEFINE FACILITY MVSADMIN.XCF.CFRM UACC(NONE) OWNER(SUPMVS)
RDEFINE FACILITY MVSADMIN.XCF.SFM UACC(NONE) OWNER(SUPMVS)
RDEFINE FACILITY MVSADMIN.LOGR UACC(NONE) OWNER(SUPMVS)
PE MVSADMIN.LOGR CLASS(FACILITY) ID(SYSPLXAD) ACC(UPDATE)
SETR RACLIST(FACILITY) REFRESH
```

Access control to the BPXMCDS couple dataset

There is no policy associated with the z/OS UNIX couple data set.

Access control to the WLM policy

For the WLM couple data sets, use an ISPF interface to create policy administrative data. Protections apply here at different levels:

- ISPF menu and application protection. There is nothing to be considered here that is specific to WLM.
- Service definitions can be prepared and managed in a partitioned data set before being installed and activated in the WLM couple dataset. Here again regular dataset protection applies, with permission given to whoever has need to access the data set. Give READ access only to those people who should be able to view the service definition in the partitioned data set, and UPDATE access to those people who should be able to create or modify the service definition in the partitioned data set.
- Access to the WLM policy in the couple data set. Once you have determined who needs access to the WLM couple data set itself, define the kind of access authority required, as follows:
  - With READ access, the user can extract a service definition from the WLM couple data set.
  - With UPDATE access, the user can:
    - Do all the functions available for READ access.
    - Install a service definition to a WLM couple data set.
    - Activate a service policy.

To control access to the WLM couple data set, use RDEFINE to add a profile to the RACF database. Then use PERMIT to permit or deny access to the RACF profile. Do not forget to issue the SETROPTS REFRESH command after the PERMIT command to refresh the RACF database and activate the changes you have made.

```
RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE) NOTIFY(user)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user) ACCESS(READ)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user) ACCESS(UPDATE)
```

User indicates the user or user group that needs access to the WLM couple data set and ACCESS indicates the type of access (either READ or UPDATE).

2.3.3 The protection of the System logger resources

Figure 2-6 on page 33 shows a System Logger configuration for a coupling facility logstream. The System Logger resources to be protected are:

- System logger address space
- User ID setting up the CFRM and LOGR policies

- System logger applications (for example, logrec or OPERLOG) that will be accessing the log streams.

Note that the same considerations for protection apply for a DASD-only log streams configuration.

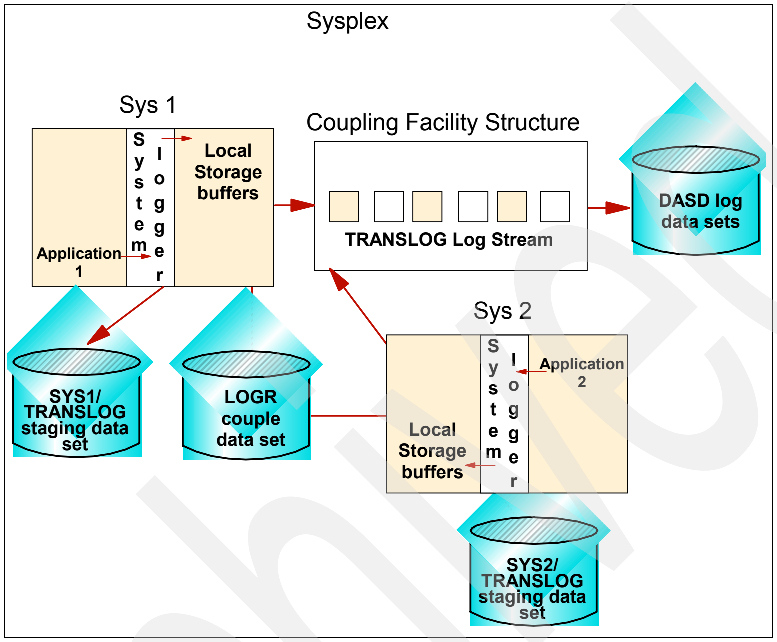


Figure 2-6 System Logger configuration

Define authorization for the System Logger address space

IBM recommends that the system logger address space (IXGLOGR) be assigned privileged or trusted RACF status. After assigning the system logger address space with the privileged or trusted status, you must restart the system logger address space for the newly assigned authorization to take effect. You can stop and restart the system logger address space with:

```
FORCE IXGLOGR,ARM
S IXGLOGRS
```

If the system logger address space for your installation is neither privileged nor trusted, make sure to give IXGLOGR the following SAF authorizations:

- Define IXGLOGR in either the RACF started procedures table (SPT) or in the STARTED RACF class of profiles.

- For coupling facility log streams, define ALTER access to the resource <IXLSTR.structure_name> in the FACILITY class for access to the log stream coupling facility structures.
- Define ALTER access to the resource <hlq.data_set_name> in the DATASET class for each DASD log stream and staging data set. Note that the resource could also be identified by an extended high-level qualifier, such as <ehlq.data_set_name>.
- Define read access to the resource <sys1.parmlib_data_set_name> in the DATASET class for access to SYS1.PARMLIB.

The DASD log stream and staging data set name high-level qualifier (*hlq*) or extended high-level qualifier (*ehlq*).name are specified in the DEFINE LOGSTREAM statement used when defining the LOGR policy with the IXCMIAPU utility. This is illustrated in Example 2-6.

Example 2-6 DEFINE LOGSTREAM SYSIN for IXCMIAPU utility

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE STRUCTURE NAME(SYSTEM_OPERLOG)
   LOGSNUM(1)
DEFINE LOGSTREAM NAME(SYSPLEX.OPERLOG)
       DESCRIPTION(OPERLOG_SYSTEME)
       STRUCTNAME(SYSTEM_OPERLOG)
       AUTODELETE(YES)
       RETPD(3)
       LS_DATACLAS(SYSPOSC)
       LS_STORCLAS(SYSPOSC)
       LS_SIZE(18000)
       HLQ(MVSLOG)
       HIGHOFFLOAD(80)
       LOWOFFLOAD(00)
       STG_DUPLEX(NO)
```

The extended high-level qualifier is similar to the high-level qualifier, but it provides more flexibility to help meet the installation’s naming conventions for log stream data sets. The maximum length of the extended high-level qualifier, including periods, is 33 characters. The high-level qualifier and the extended high-level qualifier are mutually exclusive and cannot be specified for the same log stream definition.

The high-level qualifier will be used in the DASD log data sets and DASD staging data sets for each log stream. For example, depending on the needs of your installation, you could set the high-level qualifier to one of the following:

- Sysplex name
- Subsystem group name
- System logger application type name

System logger does not do SAF authorization checking for the high-level qualifier you select for the DASD log data sets. This means that while you can select any high-level qualifier you like, you should also carefully plan the name you choose. For example, system logger will allow you to choose SYS1 as your high-level qualifier, but your DASD log stream data sets will end up in your master catalog. IBM recommends that you plan your high-level qualifier carefully and add an alias for each high-level qualifier in the master catalog that points to a user catalog.

If you do not specify a high-level qualifier, system logger uses the default, IXGLOGR.

The format of the name of the system logger staging data set is:

- *hlq.logstreamname.*A0000000
- *hlq.logstreamname.*A0000000.DATA
- *hlq.logstreamname.*A0000000.*sysname*

Where:

- *hlq* is the high-level qualifier defined in the LOGR policy, which is MVSLOG in the example above. If there is no high-level qualifier specified, the default value IXGLOGR will be used.
- *logstreamname* is the name of the logstream name specified in the LOGR policy, which is SYSPLEX.OPERLOG it the example above.
- *sysname* is the name of the system. POS1 will be used as the system name, as in Example 2-7.

The corresponding logstream and staging dataset name are shown in Example 2-7.

Example 2-7 Resulting LOGSTREAM and STAGING dataset

```
MVSLOG.SYSPLEX.OPERLOG.A0000000
MVSLOG.SYSPLEX.OPERLOG.A0000000.DATA
MVSLOG.SYSPLEX.OPERLOG.POS1
```

A generic profile for your chosen high-level qualifier (such as MVSLOG.** in Example 2-7, or a more specific profile) must be created with UACC(NONE) in the DATASET class. This profile will be used to protect log stream data sets and staging data sets. Appropriate authorization should be given to administrators or people needing to access these datasets.

Define authorization for setting up policies

The LOGR policy information describes the characteristics of each log stream and coupling facility structure that will be used when there are active connections to the log stream.

The CFRM policy contains the definition of all coupling facility structures used by system logger, if applicable.

Before setting up these policies for your installation with the IXCMIAPU Administrative Data Utility program, authorize who can use IXCMIAPU:

1. Define a resource profile for the resource name MVSADMIN.LOGR to the FACILITY class. Assign ALTER access authority to users who must alter or maintain the policy. Assign READ access authority to users who require reports on the policy, but who will not change the policy.
2. If applicable, define a resource profile for the resource name MVSADMIN.XCF.CFRM to the FACILITY class. Assign UPDATE access to users who must define coupling facility structures in the policy.

The users of the IXCMIAPU utility require authorization to the resources accessed by the utility. Define the appropriate authorizations before users attempt to add, update, delete, or extract a report from the LOGR policy. The LOGR couple data set must be activated and available before you add log streams and structure definitions to it. The following authorizations are required to set up the LOGR policy:

- To DEFINE, UPDATE, or DELETE LOGSTREAM, specify ALTER access authority to the resource <log_stream_name> in the LOGSTRM class.
- To DEFINE or DELETE STRUCTURE, specify ALTER access authority to the resource <IXLSTR.structurename> in the FACILITY class.
- To specify a structure name on a LOGSTREAM definition (for example, DEFINE LOGSTREAM(log_stream_name) STRUCTNAME(structname)), specify UPDATE access authority to the resource <IXLSTR.structurename> in the FACILITY class.

Authorization for System Logger applications

For an installation or vendor-written system logger application, the access to system logger resources, such as log streams or coupling facility structures associated with log streams, depends upon which system logger services the application issues:

- For a vendor-written application, see the documentation provided by the vendor.
- For an installation-written application, see the application programmer for authorization requirements of the program.

The application user ID needs an UPDATE access to profiles in the LOGSTRM class. The profile can have any name, but is expected to somehow explicitly designate the log stream it protects. Example 2-8 shows how these profiles are defined.

Example 2-8 Define logstream resource in LOGSTRM class

```
RDEFINE LOGSTRM SYSTEM_OPERLOG UACC(NONE) OWNER(SUPMVS)
PE SYSTEM_OPERLOG CLASS(LOGSTRM) ID(SYSPLXAD) ACC(UPDATE)
SETR CLASSACT(LOGSTRM)
SETR RACLIST(LOGSTRM)
```

2.3.4 Controlling the use of Automatic Restart Management

The reference documentation is *MVS Programming Sysplex Services Guide*, SA22-7817.

The Automatic Restart Management (ARM) policy defines how to process restarts for started tasks and batch jobs that have registered with the automatic restart management.

In order to perform automatic restarts ARM issues commands from the XCF address space. To allow this to happen, XCFAS (the XCF Address Space) must have enough authority to issue operator commands to restart any failed element. We recommend that the XCFAS procedure be defined in the RACF STARTED class or the started procedures table with the TRUSTED attribute.

The security administrator can control the use of automatic restart management services, via the IXCARM macro, by unauthorized applications through the use of RACF. Unauthorized applications can use only the element names that are registered as ELEMBIND=CURJOB elements.

Through the IXCARM macro, a program can:

- Register as an element of automatic restart management and, optionally, specify restart parameters and an event exit (REGISTER parameter).
- Indicate when it is ready to receive work (READY parameter).
- Deregister from automatic restart management when the program or application no longer needs to be restarted.
- Indicate that MVS should delay the restart for this program until MVS completes the restart of a related program, which is called a predecessor element (WAITPRED parameter).
- Identify itself as the backup program for another element of the automatic restart manager.

An unauthorized application can request all functions of IXCARM. However, there are restrictions. For example, an unauthorized application cannot specify ELEMBIND=CURSYS for any of its elements.

To define RACF profiles that control unauthorized applications’ use of automatic restart management, the security administrator can:

1. Define a resource profile IXCARM.elemtype.elemname in the FACILITY class.
2. Specify the users who have access to automatic resource management services using the RACF PERMIT command.
3. Make sure the FACILITY class is active and generic profile checking is in effect. If in-storage profiles are maintained for the FACILITY class, refresh them.

For example, if a user wants to permit an unauthorized application with an *elemtype* of xxxxxxxx and an *elemname* of yyyyyyyyyyyyyyyy to use automatic restart management services, the security administrator would use the following commands:

```
RDEFINE FACILITY IXCARM.XXXXXXXX.YYYYYYYYYYYYYY UACC(NONE)
PERMIT IXCARM.XXXXXXXX.YYYYYYYYYYYYYYYY CLASS(FACILITY) ID(userid)
ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
```

Establishing security for restarted jobs

When restarting an element, automatic restart management either can use the JCL that previously started the element (persistent JCL) or can override that JCL by specifying the installation-supplied name of a data set that contains the JCL for restarting the element.

The following security considerations apply when restarting elements with either persistent or override JCL:

- When restarting an element with persistent JCL, automatic restart management establishes the same security environment as existed when the element last ran.
- When restarting an element with override JCL, automatic restart management establishes the same security environment as existed when the element most recently registered with automatic restart management. Within this security environment, automatic restart management both opens the data set containing the override JCL and submits the job. In most cases this has the effect of propagating the original element's job security information to the restarted element. However, there are a few special cases that you should consider:
  - If the override JCL specifies a different user ID (with the USER= parameter on the JOB statement), then MVS does not propagate the most recent user ID to the new element and instead uses the new user ID specified. The override JCL also must specify the new user's password, unless the most recent user has the appropriate RACF SURROGAT authority to specify the new user ID.
  - If the override JCL does not specify a different user ID, then MVS propagates the most recent user ID to the new element. However, MVS does not propagate the most recent group ID to the new element. Instead, MVS uses the most recent user's default group as the new element's group ID unless the GROUP= parameter on the JOB statement specifies a different group ID. The fact that MVS does not use the most recent group ID should not normally cause any security problems for the new element. However, there are cases where access might be denied. The following examples might experience this effect:
    - When using RACF, you run with SETROPTS NOGRPLIST (disabling list-of-groups processing) specified.
    - When using RACF, you use &RACGPID (affecting the user's current connect group) in some members of the GLOBAL class.

  a. When using override JCL, the input source (port of entry, POE) for the new element will be INTRDR. This might differ from the input source of the original element, but should not normally cause any security problems. However, access might be denied in some cases, such as when using RACF and the element requires conditional access list entries that specify WHEN(JESINPUT(*xxx*)), where *xxx* is the input source of the original element.

Note: When using override JCL, submission of the new element will fail if all the following conditions exist:

- You use RACF.
- The JOB statement does not specify a new user ID.
- The most recent execution user is protected by RACF's PROPCNTL class.

This applies particularly to the restart of CICS® regions, where many customer installations disallow propagation of the CICS region's user ID to a submitted job using the PROPCNTL option. However, this would only pose a problem if you use override JCL during the restart.

2.3.5 Sysplex operator command protection

It is even more important in a sysplex installation to establish access control of the operator commands, as some commands may have a sysplex-wide scope.

Establishing console security

Console security means controlling which commands operators can enter through their consoles to monitor and control z/OS.

The LOGON keyword in the CONSOLxx member of the SYS1.PARMLIB is where you define the logon requirements for the consoles (except for the system console). The logon requirements affect how access to the operator commands is controlled. The options are:

- Have the system automatically log each console on as the console is activated.
- Require each operator to log on to the system before issuing commands.
- Allow MCS console command authorization to control access to commands.

To audit all command activity by operator user ID or to control which commands individual operators may issue, specify LOGON(REQUIRED) on the CONSOLE statement. Specifying this is especially important for SMCS (SNA MCS) consoles.

LOGON in the CONSOLxx member

- LOGON(REQUIRED):
  - Specifies that an operator must log on before issuing commands from the console. If an operator is not logged on to that console, the system rejects commands issued from that console and issues a violation message.

- No operator should leave the console unattended without first issuing the LOGOFF command. Issuing LOGOFF leaves the console in a secure, unattended state.
- You should also ensure that at least one console or operator is logged on with master authority to be able to communicate with the system.
- Before setting LOGON(REQUIRED), your installation must define RACF profiles for all operators and for the commands and consoles you want to protect.

- LOGON (AUTO)
  - Specifies that consoles are automatically logged on when the consoles are activated.
  - The automatic LOGON uses the console name as the logon user ID.
  - This is the recommended option.
- LOGON (OPTIONAL)
  - Specifies that the operators can optionally log on to the MCS consoles. This option is the default. You are required to log on to a console if you have to enter a command that belongs to a higher command group than what is accepted from the console. After logon, you can enter all commands that are authorized for your user ID. The opposite is also true. If your user ID is not authorized to enter any command, after logon all commands entered from the console are rejected.
  - This option is not recommended.

Console command authorization without RACF checking

When RACF is not active or when the RACF class OPERCMDS is not active, MVS command processing accepts all commands from any console that belongs to command groups with the same or lower authorization requirements as assigned for the console. For example, consoles with MASTER authority can issue all commands, including those that affect other consoles (including extended MCS consoles). Consoles with I/O authority can issue in the I/O command group.

Console Command authorization with RACF checking

When protecting commands and consoles with RACF resources profiles, both OPERCMDS and CONSOLE classes must be active.

RACF command authorization checking overrides MCS command authority. The command issuer’s RACF profile and group authority determine what commands can be successfully entered into the system.

When the RACF class CONSOLE is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to log on has the proper authority to do so. To be able to log on, an operator on an MCS console must have read access to the profile in the CONSOLE class named with <console_name>.

EMCS command authorization checking

An extended MCS console is established dynamically by an authorized program through the MCSOPER REQUEST=ACTIVATE macro service. When activating the extended console, the console attributes, such as its command authority, routing codes, message data space size, need for a migration ID, and whether it is to receive the hardcopy message set are specified through either:

- The OPERPARM segment of the user IDs RACF profile
- The OPERPARM field in the MCSOPER macro
- System defaults

If your installation plans to use extended MCS consoles, you should consider ways to control what an authorized TSO/E user can do during a console session. Because an extended MCS console is associated with a TSO/E user ID and not a physical console, you might want to use RACF to limit not only the MVS commands a user can enter, but from which TSO/E terminals the user can enter the commands. You can control whether an operator can enter commands from a console through either of the following:

- The AUTH keyword on the CONSOLE statement of CONSOLxx, specifying the group of operator commands that can be entered from the console.
- The LOGON keyword of the DEFAULT statement of CONSOLxx, if any, and RACF commands and profiles.

An example of a CONSOLxx parmlib member is given in Example 2-9.

Example 2-9 CONSOLxx parmlib member

```
INIT
DEFAULT LOGON(REQIRED/AUTO/OPTIONAL)
CONSOLE DEVNUM(020) ALTERNATE(021) ROUTCODE(ALL)
      NAME(MAST020)
      PFKTAB(PFKTAB1)
      AUTH(MASTER/SYS/INFO/IO/CONS/ALL)
      LOGON(AUTO/REQUIRED/OPTIONAL/DEFAULT)
      UNIT(3279-3B)
      MONITOR(JOBNAMES-T)
      CON(N) SEG(28) DEL(RD) RNUM(20) RTME(1/4)
MFORM(J,S,T) AREA(NONE)
```

The security administrator can define a RACF user profile to control the console attributes of the extended MCS console user. The following example shows how to define a RACF profile for the new TSO/E user TAPE1:

```
ADDUSER TAPE1  OPERPARM(ROUTCODE(46) AUTH(SYS) MFORM(S) ALTGRP(TAPEGR))
```

This example defines the user ID TAPE1 as an extended MCS console with console attributes defined by the OPERPARM keyword.

RACF command resource names

Your installation's security policy determines which commands you must protect. A RACF profile for the command in the OPERCMDS class protects the command. When an operator logs on to a console and issues an MVS command that requires a higher authority than the console allows, RACF can check the access list of the command profile to determine whether the user is authorized to issue the command.

To link the command the operator issues with the profile that protects the command, MVS provides a construct, or structure, called a resource-name for each command.

The command resource names enable you to logically group operator commands and to name each group for the purpose of controlling RACF access to the commands.

A command resource name can include up to four parts: a system identifier, the command or a variation of the command, a command qualifier, and a command object. These parts enable you to define a naming hierarchy that can identify specific commands or command subsets. We recommend that you use the syntax:

```
system_identifier .command [.command_qualifier [.command_object]]
```

Where:

system_identifier	Identifies the system, subsystem, or application to which the command belongs. For example, IBM uses MVS to identify MVS operator commands, JES2 to identify JES2 commands, and JES3 to identify JES3 commands.
command	Identifies a specific command or some variation of a command. Where possible, use the command name. In cases where the command name does not provide the level of identification you require, use a variation of the command name.
command_qualifier	Allows you to more precisely identify the command variation in question.

command_object Identifies the object of the command. Including the *command_object* as part of the command resource name enables you to control access to commands based on the object the command affects.

Example 2-10 is an example of the protection of the MVS ROUTE command and its command resource name with a command_qualifier and a command_object.

Example 2-10 Define MVS commands in OPERCMDS class

```
SETR CLASSACT(OPERCMDS)
SETR RACLIST(OPERCMDS)
RDEFINE OPERCMDS MVS.ROUTE.CMD.POS1 UACC(NONE) OWNER(SUPMVS)
PE MVS.ROUTE.CMD.POS1 CLASS(OPERCMDS) ID(OPEGROUP) ACC(READ)
SETR RACLIST(OPERCMDS) REFRESH
```

The OPERCMDS class must first be activated and raclisted, and a profile may be defined for each of these commands. A list of the sysplex-related MVS commands that should be protected can be found in Appendix A, “RACF protection of MVS commands” on page 209.

The case of the route command

The ROUTE command can be used to direct a command to one or more systems in a sysplex for processing. You can direct a command to:

- All systems in the sysplex
- A subset of the systems in the sysplex
- One system in the sysplex

For most system commands routed to multiple systems, the system combines the command response. The combined response sorts the command responses by system name. The response to the command is returned to the issuing console.

You cannot send more than one command on a single invocation of the ROUTE command. If you need to route multiple commands in strict sequential order, you should route one command, wait for a successful response from all systems to which you routed the command, and then route the next command.

Command flooding

Most z/OS commands are executed by attaching a task in either the *MASTER* or CONSOLE address space. If too many of these tasks are attached at one time, the system could run short of space in Local SQA (LSQA) and eventually enter wait state, which would require a re-IPL to restart the system.

The SEND and the ROUTE commands are two commands that can take a long time to execute. If a lot of these commands are entered, the system could run short of storage with an ABEND as a consequence.

A new function, integrated in z/OS beginning with z/OS V1R4, will prevent commands from flooding the system, which would eventually cause an LSQA exhaustion. Attached commands that run in the *MASTER* or CONSOLE address space are divided into six command classes, which are:

- Class M1 commands may be essential to clearing a backlog of Class M2 commands (like DISPLAY GRS, SET XCF, DISPLAY XCF, and so on).
- Class M2 commands are ordinary attached commands that run in the MASTER address space (like ACTIVATE, START, DISPLAY CF, DISPLAY OMVS, DISPLAY IPLINFO, and so on).
- Class M3 is only for SEND commands executed in the MASTER address space. These commands can take a long time to execute. Thus they require a command class different from Class M2.
- The Class C1 command might be needed to clear a backlog of Class C2 commands (like DISPLAY CONSOLES, DISPLAY EMCS, VARY CN, and so on).
- Class C2 commands are ordinary attached commands that run in the CONSOLE address space (like DISPLAY JOBS, CHNGDUMP, RESET CN, SWITCH CN, and so on).
- Class C3 is only for the ROUTE command executed in the CONSOLE address space. These commands can take a long time to execute. Thus they require a command class different from Class C2.

In each class, a maximum of 50 commands can be executing at any given time. Any additional commands in that class must wait for execution. This prevents the out-of-space condition and the resulting WAIT STATE from occurring. New messages warn of a command flood. Those commands not attached are not lost but rather held in the IEECMCAD data space until a previous command is processed.

To manage the number of commands that are awaiting execution, the system operator can issue the CMDS command to:

- Display the status of commands.
- Remove selected commands that are awaiting execution.
- Cancel commands that are executing.

When a command is removed before execution, the command issuer receives message IEE065I COMMAND NOT EXECUTED, CMD=command instead of the usual command response message.

The usage of the CMDS command needs authority from RACF. The corresponding resource name in RACF is listed in Table 2-1.

Table 2-1 Command, authority, and resource name

Command	Authority	Resource name
CMDS DISPLAY	READ	MVS.CMDS.DISPLAY
CMDS SHOW	READ	MVS.CMDS.SHOW
CMDS REMOVE	CONTROL	MVS.CMDS.REMOVE
CMDS ABEND	CONTROL	MVS.CMDS.ABEND

Note: A list of the MVS commands related to sysplex that we recommend to protect is given in Appendix A, “RACF protection of MVS commands” on page 209.

2.4 An overview of Multilevel Security (MLS)

In this section we give an overview of multilevel security. We look at what multilevel security is and why one would use it.

2.4.1 Multilevel security

A multilevel security system is a security environment that allows the protection of data based on both traditional discretionary access controls and controls that check the sensitivity of the data itself through mandatory access controls.

These mandatory access controls are at the heart of a multilevel security environment, which prevents unauthorized users from accessing information at a classification they are not authorized to, or changing the classification of information they do have access to. These mandatory access controls provide a way to segregate users and their data from other users and their data regardless of the discretionary access they are given though access lists, and so on.

Creating a multilevel security environment requires a combination of several software and hardware components that enforce the security requirements needed for such a system. The security-relevant portion of software and hardware components that make up this system can be also known as the trusted computing base.

2.4.2 Why multilevel security

The primary arena in which multilevel security is valuable is government agencies that need a security environment that keeps information classified and compartmentalized between users. In addition to the fundamental identification and authentication of users, auditing and accountability of the actions by authenticated users on these systems is provided by the security environment.

Normally in such a highly secure environment, to manage the compartmentalization of information between users, each compartment is on its own system, making it difficult for classified information to spill from one system to another, since the connections between systems can be highly controlled. With multilevel security, these systems can be consolidated onto a single system, with each compartment independent of the other, so that no transfer of data can occur between compartments within that system. This will take advantage of the cost savings of not having to manage multiple systems, but only a few, or one system.

Commercial customers may also find some features of multilevel security useful, such as to separate sensitive customer information from the general populace, or from another user. New government regulations, such as HIPAA, or corporate mergers, are examples of where security of information based on the information itself is important in the commercial world.

RACF, also known as the z/OS Security Server, has several options that can be turned on and off to manipulate different aspects of a multilevel security environment. Because it is possible to have some features of multilevel security on at one time (creating a partial multilevel security environment), commercial customers may find this type of environment useful to meet these needs versus running a full fledge multilevel security environment.

2.4.3 Access Controls

Discretionary Access Control (DAC) allows access to data to be controlled by the discretion of the owner of the data (or those with administrative authority to the data). Since access to the data could be given out at the discretion of an authorized person, it would be easy to unintentionally give access to data to people who do not need access to it. For example, the owner of some data gives access to a GROUP since that group of people needs access to his data, then the group owner connects a new user to the GROUP since the new user needs access to resources that the GROUP has access to, not knowing that the GROUP also has access to sensitive data that this new user should not have access to.

Mandatory Access Control (MAC) imposes additional restrictions upon users so they now access data based on a comparison of the classification of the user and the classification of the data as well as the standard DAC checking. This additional security check verifies that users can only access data and resources that their classification allows them to, even though their discretionary access will allow them to access such data or resources.

Mandatory Access Control was implemented in RACF 1.9 with MVS/ESA™ 3.1.3 (in 1990) as part of the effort to meet the US Government criteria (TCSEC B1) as specified in the *Department of Defense Trusted Computer System Evaluation Criteria,* DoD 5200.28-STD (also known as the TCSEC or Orange Book).

In addition to checking a user's access to data based on the classification of the user and the data, the z/OS trusted computing base implementation of multilevel security provides some of these additional functions as well:

- The system does not allow a storage object to be reused until it is purged of residual data.
- The system enforces accountability by requiring each user to be identified and creating audit records that associate security-relevant events with the users who cause them.
- The system labels all hardcopy with security information.
- The system optionally hides the names of data sets, files, and directories from users who do not have access to those data objects.
- The system does not allow a user to declassify data by *writing down* (that is, writing data to a lower classification than the classification at which it was read) except with explicit authorization to do so.
- The system does not allow a user to view rows of data in a DB2 table that he is not classified to view.

2.4.4 Introduction to Mandatory Access Control

There are several principles one needs to comprehend to fully understand the interactions of MAC checking — the given classification for a particular piece of data, a resource, and a user as defined by the security label for that item. These security labels are then used to determine what access will be allowed. For example, a user looking at some data in a file, the subject's (in this case the user) security label is compared to the object's (in this case the file) security label. A dominance or equivalence relationship is determined, should one exist, and the type of access requested by the subject onto the object is determined. Based on that, MAC access to the object will be determined.

Breaking this down, one can see that there are three major principles that should be understood. They are:

- Security labels
- The type of relationship between security labels (dominance, equivalence, and disjoint)
- The type of access requested (MAC Access)

These principles are fully covered in “Security labels” on page 49, but here we take a high-level look at them.

Security labels

A security label is a combination of a hierarchical level of classification (security level) and a set of zero or more non hierarchical categories (security category).

In Figure 2-7 on page 50 we show a basic table that has a list of hierarchical levels from 1 to 4, where 1 is the lowest and 4 is the highest security level, and a list of non hierarchical categories from category A to category E. Using this table we can quickly create several security labels combining a security level, and zero or more categories. Some example security labels are:

- L1A is a combination of security level 1 with category A.
- L2BCD is a combination of security level 2 with categories B, C, and D.
- L3N is a combination of security level 3 with no categories.

Note: The security labels we use for this example are defined as they are for ease of use, and to allow the reader to quickly determine a security label’s security level and categories. So as an example, using Figure 2-7 on page 50, one could also create the additional security labels:

- ARTHUR is a combination of security level 2 with category C.
- FORD is a combination of security level 3 with categories A, C, and E.
- MARVIN is a combination of security level 3 with no categories. This security label is also equivalent to L3N.

Figure 2-7 has several more security labels defined, and one could easily create several more based on this simple example.

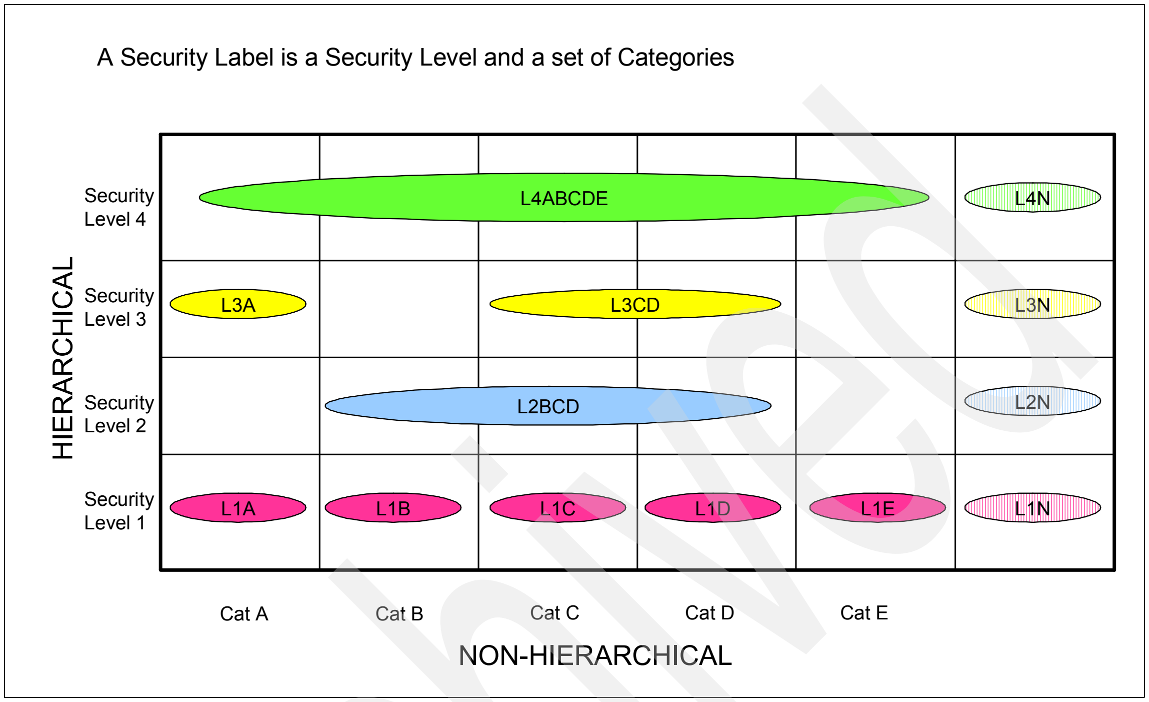


Figure 2-7 Simple security label diagram

Dominance, equivalence, and disjoint

Access is granted or rejected based on the relationship between the subject's security label and the object's security label, and the type of access that is requested. Two important relationships between security labels are dominance and equivalence. Though other relationships between two security labels may exist, in those cases authority to the object is not allowed.

Dominance

For security label A to dominate security label B the following must be true:

- The security level of A is greater than or equal to the security level of B.
- Security label A has at least all the categories that define security label B.

Returning to Figure 2-7, security label L3CD, for example, would dominate security labels L1N, L1C, L1D, L2N, L3N, and L3CD, since L3CD is at an equivalent or higher security level then the security labels it is dominating, and it contains all the categories that these security labels have. Notice that even though security label L2N has no categories, it will be dominated by all security labels at a higher or equivalent security level and any categories.

With reverse dominance access checking, the access rules are the reverse of the access rules for dominance access checking.

Equivalence

For security label A to be equivalent to security label B the following must be true:

- The security level of A is equal to the security level of B.
- Both security labels A and B must have the same set of categories.

Another way one can say that two security labels are equivalent is if both security labels dominate each other.

Returning to Figure 2-7 on page 50, the only case of equivalence in the figure is if a security label is being compared to itself (that is, the subject and object security labels are the same).

Disjoint

If neither security label dominates the other then the two security labels are said to be *disjoint* or *incompatible.*

MAC access

In addition to the type of relationship, as documented above, the type of access that is requested is also important. There are only three types of accesses that can be requested: read-only, read/write, and write-only.

Read-only test

The users intend to only read information. Therefore, they should be able to read information at their classification or information at a lower classification. In terms of MAC checking, the users' security label must *dominate* the object's security label that they intend to read. This allows them to read information covered by their security label, but not information of a higher level, or outside the users' category.

Write-only test

The user is intending to only write information. Therefore she should be able to write information at her classification or information at a higher classification. In terms of MAC checking, the object's security label that she intends to write to must *dominate* the user's security label. This allows her to write information at his security label or higher, but not information of a lower level, or outside the user's category (that is, she will not be able to declassify information).

Read/write test

The user intends to read and write information. Like in the read-only case, the user's security label must be able to *dominate* the object's security label; *and*, like the write-only case, the object's security label must be able to *dominate* the user's security label. Therefore, for the user to do a read/write action to the object, the user's and object's security labels must both be *equivalent* to each other. This allows his to read and write information only at his security label, but not outside his security label (that is, he will not be able to declassify information).

Controlled write-down

There might be cases where you want to allow for controlled situations of write-down. The security administrator can assign a *write-down by user* privilege to individual users or groups of users that allows them to select the ability to write down. The security administrator activates and deactivates the privilege by creating the profile IRR.WRITEDOWN.BYUSER in the facility class. A user can activate write-down mode if the profile exists, the user has at least read access to it, and the FACILITY class is active and SETROPTS RACLISTed. If the user has update or higher access to the profile, write-down mode is active by default when the user enters the system. RACF provides the RACPRIV command and z/OS UNIX provides the **`writedown`** command, which allows users who are authorized to the write-down privilege to reset and query the setting of there write-down mode.

Access rules

Access rules depend on the purpose for which a subject accesses an object and whether the subject is allowed to write down.

The subject is not allowed to write down

A subject is not allowed to write down if the RACF MLS option is active and the security administrator has not set up controlled write-down and given the subject authorization to write down. This should be the case for most users in a multilevel secure environment. In this case the access rules, depending on the purpose for which the subject accesses an object, are:

- Read only: A subject can read an object when the subject's security label dominates the object's security label.
- Write only: A subject can write to an object when the object's security label dominates the subject's security label.
- Read/write: A subject can read from and write to an object only if the security labels of the subject and object are equivalent.

Only the basic case is documented here — that is, a normal MAC check is being done with the *no write-down* rule in effect. With RACF there are several other

RACF options that could effect how these three different MAC access tests are done, which are covered in more detail in “Security labels” on page 49.

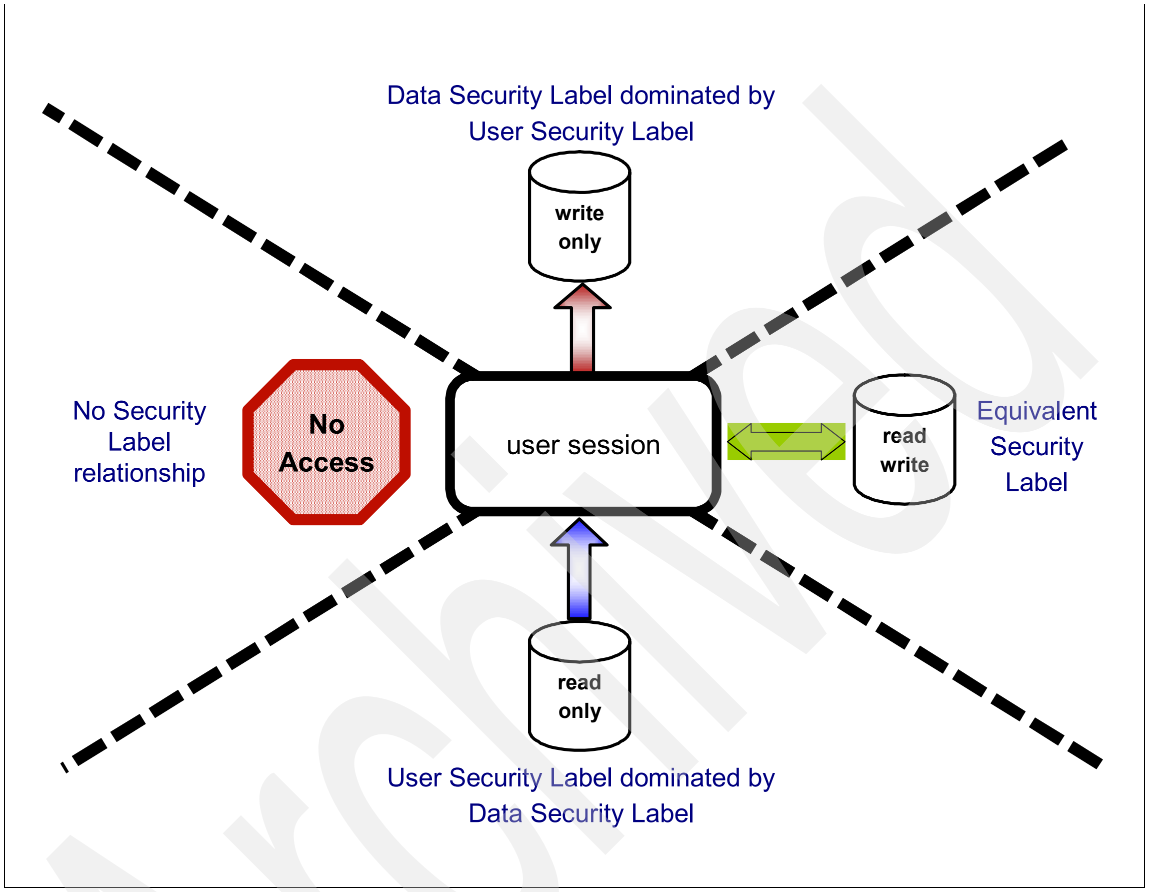

Data Security Label dominated by User Security Label
write only
No Security Label relationship
No Access
user session
read write
Equivalent Security Label
read only
User Security Label dominated by Data Security Label

Figure 2-8 MAC principle

To summarize the basic MAC principle (as seen in Figure 2-8):

- Should the user’s security label dominate the data’s security label, the user can read the data.
- Should the data’s security label dominate the user’s security label, the user can write to the data.
- Should the user’s and data’s security labels be equivalent, then the user can read and write to the data.
- Should there be no equivalence or dominance relationship between the user’s security label and data’s security label, the user will be denied access to the data.

2.4.5 Multilevel security in z/OS with RACF

Even though the theoretical implementation of multilevel security may seem reasonable, the actual implementation that is done by a trusted computing base maybe slightly different. The trusted computing base, in this case z/OS, must make certain decisions about what defines certain actions, and then make the appropriate security decision based on that definition of that action. This is important to understand because what one might think is a certain type of MAC access, the trusted computing base may have decided is a different type of MAC access.

For example, in z/OS the case of a user attempting to do a read-only or read/write action against a data set is straightforward enough. But because of the way z/OS handles data set processing, there is no real case, in the z/OS trusted computing base, where one could do a write-only action against a data set. This is because every time a user needs to write to a data set, z/OS must read some part of the data set before preforming the write action. As a result, the z/OS trusted computing base has decided that there is no case where it will do a write-only test against a data set, so no such test exists.

So as one can see from this simple example, it is not only important to know how multilevel security works, but also how it is implemented by the trusted computing base.

SECLABELs

In RACF, security labels are defined in the SECLABEL class, and are often called SECLABELs (versus security labels). As discussed earlier, a SECLABEL is a combination of a security level (saved in the SECLEVEL profile in the SECDATA class) and a set of zero or more categories (saved in the CATEGORY profile in the SECDATA class).

Users are then granted access to any SECLABELs they need authority to. Although a user can only have one SECLABEL active at a time for a session, at logon time, he can request to log on with any SECLABEL he has authority to. So during the life of that session, he is considered to be at the SECLABEL he logged on with. Resources such as a data sets that require a security label are given SECLABELs as well.

Now with all the SECLABELs in place and SECLABEL class activated, when a user requests access to resources (such as a user reading a data set), a MAC check is done using the user and resource SECLABELs, followed by a DAC check using the resource's access list. Note that MAC will occur first (once the SECLABEL class is activated), then DAC.

Multilevel security in action

Now that we have a basic idea of how multilevel security works, let us go though a simple example of multilevel security in action.

In the following example, we use the security labels that were defined in Figure 2-8 on page 53. We have the user MATT and provide him with the authority to the SECLABEL L4ABCDE (that is, he has a security level of 4, and access to categories A, B, C, D, and E), and all the data sets he owns have his SECLABEL. The user ROLAND has authority to the SECLABEL L3CD, and all the data sets he owns have his SECLABEL. Finally, the user CHRIS has authority to the SECLABEL L1A, and all the data sets he owns have his SECLABEL as well. In addition, we have data sets defined to the group ITSO, and give all the data sets with its high-level qualifier a SECLABEL of L1D. Let us finally assume that all the users also have UPDATE authority to all the data sets in this example.

Working with the user ROLAND (who will be the subject taking these actions), he will be able to read and write to his own data sets (the data sets being the object receiving the actions in all these cases) since his user ID will have an equivalent SECLABEL to the data set SECLABEL of L3CD.

The user ROLAND can now only read those data sets that belong to the group ITSO, since ROLAND's SECLABEL of L3CD will dominate the ITSO's data set SECLABEL of L1D (that is, his SECLABEL's security level is greater than the ITSO's security level, and his SECLABEL contains all the categories that the ITSO's SECLABEL has).

He cannot read anything in the data sets belonging to MATT since the security level is higher then his, and his SECLABEL has more categories then ROLAND's SECLABEL (that is, the MAC check fails), even though his discretionary access allows him access to those data sets.

Also, ROLAND cannot access any of the data sets that belong to CHRIS. Even though ROLAND's SECLABEL has a higher security level then CHRIS's, the categories that ROLAND's SECLABEL can look at are not included in the categories that CHRIS has access to.

The above is a very simple picture of multilevel security in a z/OS environment. One can also see that movement of data is very restrictive. It is also possible to implement only certain parts of multilevel security to create a partial multilevel security environment. This reduces some of the restrictions on the movement of data, though making a less secure system (from an multilevel security point of view).

2.4.6 Before turning on multilevel security

A very important part of preparing to run in a multilevel security environment, or even a partial multilevel security environment, is planning. There is no way to emphasize this more than to say that the next most important thing is planning.

To put it simply, *planning is key*.

Part of the complexity that is created by this environment is the approach that we have seen most people take towards multilevel security. Commonly, it is taking a system and breaking it down into multiple compartmental environments (though in truth it maybe better to take an approach of imagining things as individual compartmentalized systems being incorporated into a single system). Each compartment is independent of the other, such that no data movement can occur between compartments within that system, yet taking advantage of the fact that everything is on a single system so that individuals that have the security label that allows them to see information at their classification or lower can, without having to move from one compartment to another.

Another part of the complexity of establishing a working multilevel security environment (or even a partial environment) is that when turning on or off any feature of multilevel security, one is not just turning it on for an application, but for the entire MVS image. Even though there are several multilevel security options, these options affect the entire MVS image, not just a single application. So one does not only have to consider the one application that he is using to take advantage of multilevel security, but how that will effect the entire MVS image.

So one can see, before starting to work with a multilevel security environment (even a partial multilevel security environment), proper prior planning will help prevent problems as one starts to establish this in a production environment.

2.4.7 Multilevel security vocabulary

As one works with multilevel security, there is a lot of vocabulary one needs to be very careful about. Confusion can easily arise while working in this environment, especially since there are several terms that sound very similar, or have different meanings in different contexts.

A very good example of this is the term *MLS*. One use of this term means the RACF option MLS, which activates the *no-write down* option. The term MLS has also been used as a shortcut for the term *multilevel security*. If used in the correct context, one can easily be confused by which meaning the writer or speaker may be intending.

For the purposes of this book, we have maintained the following terms:

- Multilevel security - Running in an environment with all the multilevel security options turned on. This would include protecting all security-relevant resources by RACF with a security label, and full no-write down protection. To be in this environment, one has taken all the applicable steps as documented in Chapter 3 in *Planning for Multilevel Security and the Common Criteria,* GA22-7509-01.

  Multilevel security is also used when talking about an environment where the potential of having all the multilevel security options turned on could exist.
- Partial multilevel security - Running in an environment with some of the multilevel security options on.
- MLS or SETR MLS - This term will be reserved for the RACF SETROPTS option MLS, which turns on and off the no-write down protection (also known as the *-property).

The following terms are not very confusing when seen in written text, but when spoken can easily be confused:

- SECLEVEL - A hierarchical designation for data that represents the sensitivity of the information
- SECLABEL - A name that represents the combination of a hierarchical level of classification and a set of non hierarchical categories

2.4.8 Multilevel Security and Sysplex

In this section we further discuss multilevel security and the sysplex.

Using system-specific security labels in a sysplex

In a sysplex it can be useful to limit the use of certain security labels to certain members of the sysplex. This allows one member of the sysplex to run work at security label A, while another handles work at security label B, keeping work separated based on security classification while still sharing the RACF database. The SETROPTS option SECLBYSYSTEM allows you to use security labels on a per-system basis.

To define system-specific security labels, the security administrator specifies on which systems a security label is to be active by adding a member list to the SECLABEL resource class profile. The member names are system SMF IDs containing 1–4 characters. For example, to define the security label named SECRET as being active only on the systems with SMF system IDs SYSA and SYSB, the security administrator could define SECRET with a command like:

```
RDEFINE SECLABEL SECRET....ADDMEM(SYSA,SYSB)
```

If no member list of system IDs is added, the security label is considered to be active on all systems sharing the RACF database.

The security labels SYSHIGH, SYSLOW, SYSNONE, and SYSMULTI are always considered to be active on all systems. If a member list is added to one of their profiles, it is ignored.

To activate the use of system-specific security labels, activate the SECLBYSYSTEM option and refresh the SECLABEL class:

```
SETROPTS SECLBYSYSTEM
SETROPTS RACLIST(SECLABEL) REFRESH
```

Restrictions

The following restrictions apply to system-specific security labels:

- JES3 does not support the use of system-specific security labels. Do not activate the SECLBYSYSTEM SETROPTS option if you are using JES3.
- JES2 does not support using system-specific security labels for systems that perform NJE and OFFLOAD processing. These systems must have all security labels active. In addition, JES2 printers cannot process output unless the security label associated with the output is active on the system controlling the printer.
- If you define system-specific security labels, be aware that using a generic TSO system name at logon might not work, because the user could be allocated to a system where the user's security label is not active.
- If you use Application Restart Manager (ARM) to manage applications and you use system-specific security labels, ensure that the systems that you have told ARM to use when restarting an application are systems that have the appropriate security label active. Otherwise, ARM might try to restart an application requiring a particular security label on a system where the security label is not active, and the application restart will fail.

3

UNIX System Services Security

In this chapter we focus on the Security functions in the UNIX System Services that can be used in the context of UNIX or mixed UNIX/MVS applications running in the Sysplex. Note that at this stage of the book, as we assume that there is no connection to a non-secure network, we do not address those functions that are network oriented.

We cover the basic concepts of:

- Dual user identity and relevant access controls
- The HFS file system
- The zFS file system
- The AUTOMOVE function of the sysplex shared HFS/zFS file system

Note that we continue using the terms *MVS* and *UNIX* in this chapter to specify which Security model an item belongs to. Besides the IBM RACF reference books (for those installations using RACF), the reference document that we recommend here is *z/OS UNIX System Services Planning*, GA22-7800.

3.1 The MVS and UNIX UID dual identity

UNIX System Services Security exploits RACF through the SAF interface. The security functions provided cover user authentication, UNIX resource access control, and user privileges allocation (along with proper auditing).

A z/OS UNIX user is first an MVS user, in that the user is given an MVS user ID profile in RACF with an OMVS segment that contains a z/OS UNIX UID. Alternatively, an MVS user without an OMVS segment in the user profile can be given a default UID as defined in the BPX.DEFAULT.USER FACILITY profile.

Figure 3-1 illustrates the format of the USER and GROUP profiles in the RACF database, indicating the segments and fields containing the UNIX identity of these entities and the relationship between MVS and UNIX groups. Incidentally, this figure also shows miscellaneous profiles in the FACILITY class that are used on z/OS to bring additional security on top of the traditional UNIX Security model.

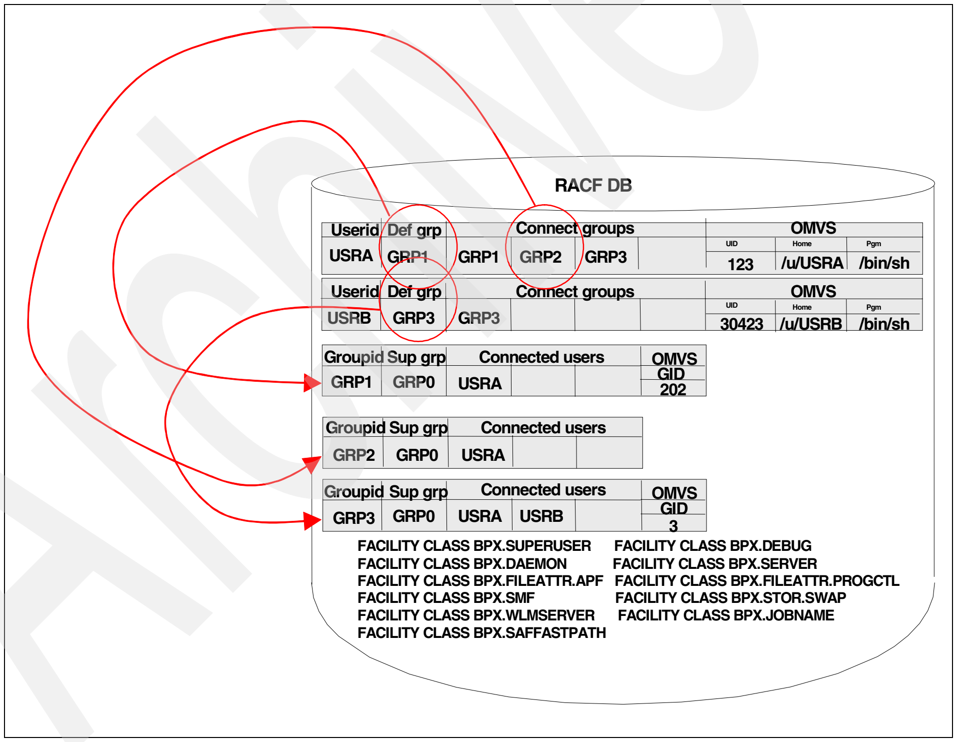


Figure 3-1 z/OS UNIX users and groups in the RACF database

In z/OS UNIX a user ID is identified to the system with a numerical value called a UID in the range from 0 through to 2,147,483,647. We recommend that all user

ID's have a unique UID value and the superuser value of UID(0) should only be assigned when absolutely necessary to system administrators and security personnel. Some z/OS UNIX applications may require their user ID to be given UID(0) to properly operate. This is expected to be explicitly required in the product documentation. If it is not then the above recommendation applies.

For example, the TCP/IP started task, with a name here of *TCPIP*, is given an MVS user ID of TCPUSER. The association between the profile for that started task and the user ID can be seen by listing the profile in the STARTED class. That is done by entering the command:

```
RLIST STARTED TCPIP.* STDATA NORACF
```

Example 3-1 STDATA information showing associated user ID

```
CLASS      NAME
-----      ----
STARTED    TCPIP.* (G)

STDATA INFORMATION
------------------
USER= TCPUSER
GROUP= OMVSGRP
TRUSTED= YES
PRIVILEGED= NO
TRACE= NO
```

Then, as required by the Communications Server installation documentation, the MVS user ID is given UID(0). You can then list the user ID TCPUSER to confirm what UID value it has with this command (Example 3-2):

```
LISTUSER TCPUSER OMVS NORACF
```

Example 3-2 User ID TCPUSER

```
USER=TCPUSER

OMVS INFORMATION
----------------
UID= 0000000000
```

The UNIX superuser

The value UID=0000000000 is the same as UID=0, and this indicates that the user ID has the SUPERUSER status within z/OS UNIX.

The UID(0) value allows the user ID full access to all the files and directories in the file system, as well as full access to define, modify, delete, and switch to any

UID. UNIX groups are also assigned a numerical value in the same range, but GID(0) has no special privileges like UID(0) has.

A UNIX superuser is also said to have *daemon authority*.

The dual identity

For organizational purposes and for ease of administration, users can be connected to groups. Groups are identified by a *group name* and are collections of users. A user who is connected to a group inherits the access rights that have been given to the group regarding the resource the user wants to access. Users can be connected to many groups that have access to the same resource. The user then benefits all of those various groups privileges at the same time when it comes to access this very resource. This function in RACF is called *list-of-groups checking*.

As soon as a task running under an MVS user ID is invoking a UNIX service, it then also takes the UNIX UID associated with the user. This process is called *dubbing* and is mandatory for the task to get access to the z/OS UNIX functions. From now on the task runs under the two identities: the MVS user name to be used for controlling access to MVS resources and the UID for controlling access to UNIX resources, as shown in Figure 3-2.

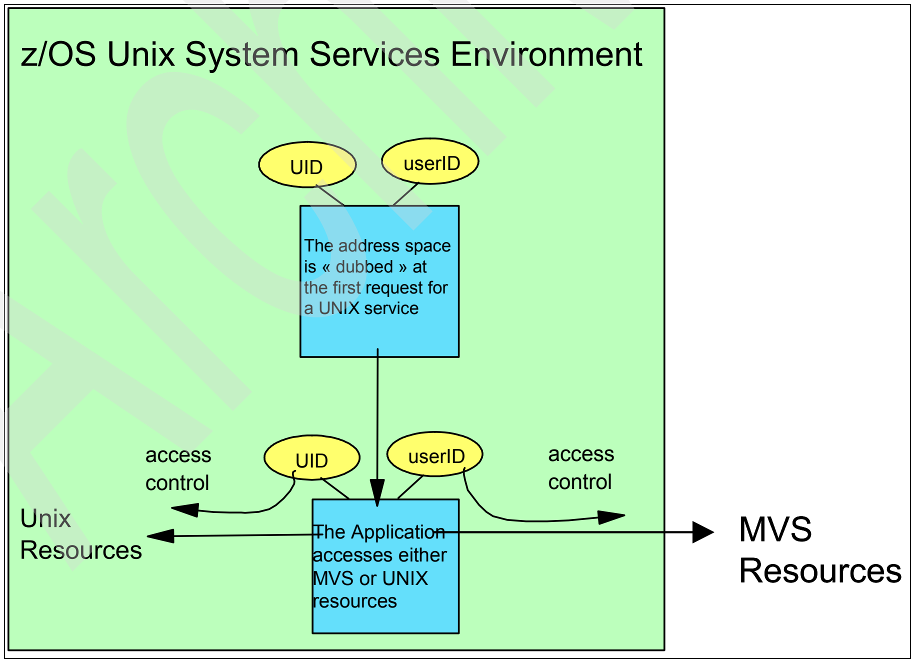

z/OS Unix System Services Environment
UID
userID
The address space is « dubbed » at the first request for a UNIX service
access control
UID
userID
access control
Unix Resources
The Application accesses either MVS or UNIX resources
MVS Resources

Figure 3-2 Dual access control to MVS and z/OS UNIX resources

3.1.1 Shared UID prevention

To prevent several users from being given the same UID number, the SHARED.IDS profile can be defined in the UNIXPRIV class. This profile acts as a system-wide switch to prevent assignment of a UID that is already in use.

The use of the SHARED.IDS profile requires having a RACF database at AIM stage 2 or 3 and is uniqueness guaranteed within the scope of the RACF database sharing.

To enable shared UID prevention, it is necessary to define a new SHARED.IDS profile in the UNIXPRIV class, as follows:

```
RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

This works for GIDs as well, as it will not allow a GID to be to shared between different groups.

Shared UID examples

In the follow example, we have created a user USER1 with UID(7). Then we tried to define a new user USER2 with the same UID(7). We received the following error message:

```
IRR52174I Incorrect UID 7.  This value is already in use by USER1.
```

You will get the following error message if you try to specify more than one user in an ADDUSER command request:

```
IRR52185I The same UID cannot be assigned to more than one user.
```

Existing shared UIDs

The use of this functionality does not affect pre-existing shared UIDs. They remain shared even after defining the SHARED.IDS profile. If you want to eliminate sharing of the same UID, you must clean them up separately. An IRRICE report is available that finds the shared UIDs.

Allowing shared UIDs

Even if the SHARED.IDS profile is defined, you may still require some UIDs to be shared and others not to be shared. For example, you may require multiple superusers with a UID(0). It is possible to do this using the new SHARED keyword in the OMVS segment of the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands.

To allow an administrator who is not SPECIAL to assign a non-unique UID or GID using the SHARED keyword, you must grant that administrator at least READ access to SHARED.IDS profile, as follows:

```
PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(ADMIN) ACCESS(READ)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

Once user ID ADMIN has at least READ access to the SHARED.IDS profile, ADMIN will be able to assign the same UID or GID to multiple users, using the SHARED KEYWORD, as follows:

```
ADDUSER (USERA USERB) OMVS(UID(7) SHARED)
```

3.1.2 Automatic UID/GID assignment

UIDs and GIDs can be assigned automatically by RACF to new users, making it easier to manage the process of assigning UIDs and GIDs to users.

Using the AUTOUID keyword with the ADDUSER and ALTUSER commands, an unused UID will be automatically assigned to the new or modified user. Using the AUTOGID keyword on ADDGROUP and ALTGROUP commands, a GID will be automatically assigned to the new or modified group.

The use of automatic UID/GID requires the following:

- A RACF Database at AIM stage 2 or 3. Otherwise, an IRR52182I message is issued and the automatic assignment attempt fails as follows:

  ```
  IRR52182I Automatic UID assignment requires application identity mapping to
  be implemented
  ```

- A SHARED.IDS profile must be defined. Otherwise, an IRR52183I message will be issued and the attempt fails as follows:

  ```
  IRR52183I Use of automatic UID assignment requires SHARED.IDS to be
  implemented
  ```

  The SHARED.IDS must be defined as follows:

  ```
  RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
  ```

  This command is explained in the previous section.

- The BPX.NEXT.USER facility class profile must be defined and RACLISTed. Otherwise, an IRR52179I message will be issued and the attempt fails as follows:

  ```
  IRR52179I The BPX.NEXT.USER profile must be defined before you can use
  automatic UID assignment.
  ```

The definition of the BPX.NEXT.USER FACILITY class profile has the following syntax:

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(UID/GID)
```

Where *APPLDATA* consists of two qualifiers separated by a forward slash (/). The qualifier on the left of the slash character specifies the starting UID value or range of UID values. The qualifier on the right of the slash character specifies the starting GID value or range of GID values. Qualifiers can be null or specified as ‘NOAUTO’ to prevent automatic assignment of UIDs or GIDs.

The starting value is the value RACF attempts to use in ID assignment, after determining that the ID is not in use. If it is in use, the value is incremented until an appropriate value is found.

The maximum value valid in the APPLDATA specification is 2,147,483,647. If this value is reached or a candidate UID/GID value has been exhausted for the specified range, subsequent automatic ID assignment attempts fail and message IRR52181I is issued.

Note: Be careful because APPLDATA is verified at the time of use, not when it is defined. If a syntax error is encountered at the moment of use of the auto assignment, an IRR52187I message is issued and the attempt fails.

Automatic assignment in an RRSF configuration

In an RRSF configuration, shown in Figure 3-3, in order to avoid UID and GID duplications, use non-overlapping APPLDATA ranges.

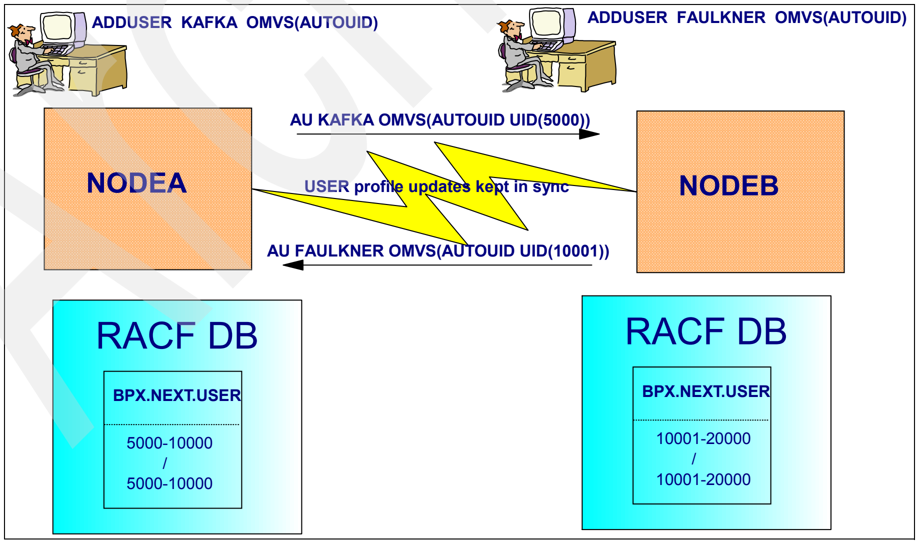

ADDUSER KAFKA OMVS(AUTOUID)
ADDUSER FAULKNER OMVS(AUTOUID)
AU KAFKA OMVS(AUTOUID UID(5000))
NODEA
USER profile updates kept in sync
NODEB
AU FAULKNER OMVS(AUTOUID UID(10001))
RACF DB
BPX.NEXT.USER
5000-10000
/
5000-10000
RACF DB
BPX.NEXT.USER
10001-20000
/
10001-20000

Figure 3-3 Automatic assignment in an RRSF configuration

An additional concern is to make RACF automatically suppress propagation of internal updates. This can be done by specifying the ONLYAT keyword to manage the BPX.NEXT.USER profile, as follows:

```
RDEFINE BPX.NEXT.USER APPLDATA('5000-10000/5000-10000') ONLYAT(NODEA.MYID)
RDEFINE BPX.NEXT.USER APPLDATA('10001-20000/10001-20000') ONLYAT(NODEB.MYID)
```

For more information about automatic assignment in a RRSF configuration refer to *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

3.2 z/OS UNIX Security

In a z/OS system, there is a system of checks and balances between security administrators (users with the RACF attribute SPECIAL), auditors (users with the RACF attribute AUDITOR), and other users such as system programmers. In this type of system, the implicit ability of a SUPERUSER to switch into the identity of any z/OS UNIX user is considered to be a potential security risk.

For safeguarding this environment, additional security functions were implemented in an earlier release of OS/390 UNIX System Services to ensure that overall system security is not lessened by the use of the basic UNIX security model. These additional security functions are optional, and therefore z/OS UNIX System Services supports two levels of system security:

- The UNIX level, which is the traditional UNIX Security model
- The z/OS UNIX level, which is the UNIX security model with, in addition, specific z/OS security functions

The documentation refers to two possible levels of activation of z/OS UNIX Security:

- The BPX.DAEMON profile is defined in the RACF FACILITY class of profiles.
- The system running with RACF enhanced program security, BPX.DAEMON and BPX.MAINCHECK, defined in the FACILITY class.

Both levels are explained below.

3.2.1 BPX.DAEMON FACILITY

If the BPX.DAEMON resource in the FACILITY class is defined, your system has z/OS UNIX security. Your system can exercise more control over your superusers. This level of security is for customers with stricter security requirements who need to have some superusers maintaining the file system but want to have greater control over the z/OS resources that these users can access. Although BPX.DAEMON provides some additional control over the capabilities of a superuser, a superuser should still be regarded as a privileged user because of the full range of privileges the superuser is granted.

The additional control that BPX.DAEMON provides involves the use of kernel services such as setuid() that change a caller's z/OS user identity.

Any user can issue a setuid(), which follows a successful __passwd() call to the same target user ID. However, a user with daemon authority (that is, a superuser) can issue the setuid() function, and therefore switch to another user identity, without providing the target user's password.

With BPX.DAEMON defined, a superuser process can freely switch to any new identity if the following statements are both true:

- The caller's user identity was permitted to BPX.DAEMON.
- All programs running in the address space have been loaded from a library that is controlled by a security product. An MVS library identified to RACF program control is an example. You can identify individual files as a controlled program by defining the file as a PROGRAM profile in RACF, for an MVS load module, or by giving the file the "p" extended attribute for a UNIX executable.

  If a service that changes a caller's z/OS user identity, such as setuid(), is used without presenting a password and the caller is not defined as a surrogate of the target UID, the kernel checks to see whether BPX.DAEMON has been defined. If it has, then the kernel checks whether all programs loaded into the address space have been defined to program control. If an uncontrolled program has been loaded, then the address space is marked dirty. In that case the following controlled functions will not be executed:

  - Setuid()
  - Spawn()
  - Pthread_security_np()
  - _login
  - su
  - setUID flag

This is illustrated in Figure 3-4, where the functions that imply switching an ID can be executed only if the *dirty bit* is off in the address space main TCB. If an application invokes one of these functions with the dirty bit on for the calling address space, then the system returns an EMVSERR return code and a JRENVIRTY reason code.

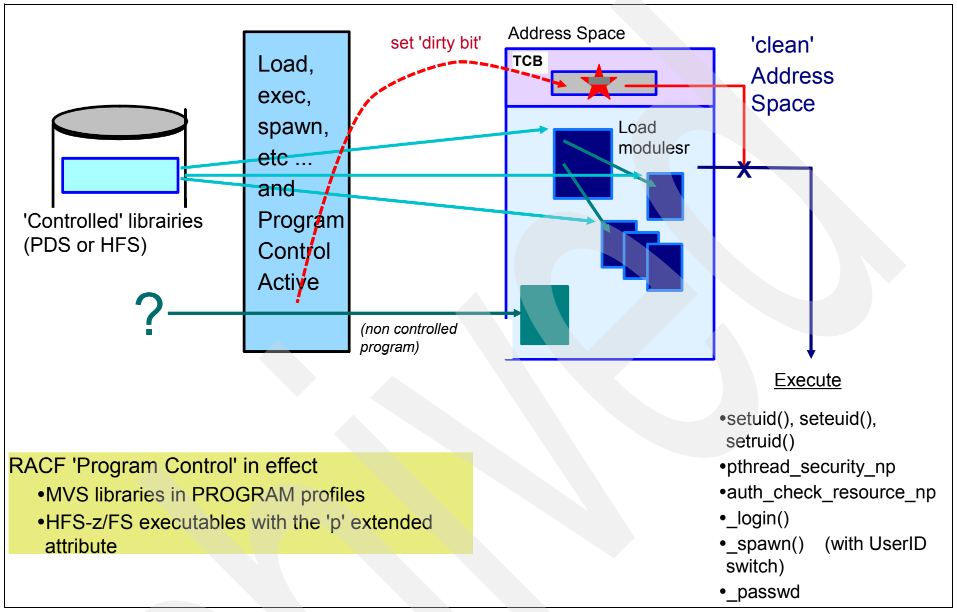

'Controlled' librairies (PDS or HFS)
Load, exec, spawn, etc ... and Program Control Active
set 'dirty bit'
Address Space
TCB
Load modulesr
'clean' Address Space
?
(non controlled program)
Execute
•setuid(), seteuid(), setruid()
•pthread_security_np
•auth_check_resource_np
•_login()
•_spawn() (with UserID switch)
•_passwd
RACF 'Program Control' in effect
•MVS libraries in PROGRAM profiles
•HFS-z/FS executables with the 'p' extended attribute

Figure 3-4 Controlled environment

However, this additional control does not guarantee that a daemon is well-behaved. It only ensures that programs come from protected sources.

3.2.2 RACF enhanced program security, BPX.MAINCHECK

RACF can operate in three modes regarding program protection:

- BASIC program security mode (default)
- ENHANCED program security mode
- ENHANCED-WARNING program security mode

The principles of operation of these modes and the related setup of RACF profiles are explained in detail in *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683, and *z/OS UNIX System Services Planning*, GA22-7800. Compared with BASIC mode, ENHANCED mode offers extra protection from hackers and other malicious users, but requires more work in setting up the PROGRAM profiles that control program protection. It also further restricts the environment in which users can make use of program access to data sets (PADS), program access to SERVAUTH resources, and execute-controlled programs. It optionally provides additional restrictions on the execution of UNIX servers and daemons.

If you choose to use the enhanced program security function, and you have daemons that use programs defined to RACF as execute-controlled programs, by having EXECUTE access to the RACF PROGRAM profile that defines the program, or EXECUTE access to the library containing the program, then you need to take some special actions to configure your daemons so that they will run properly.

In an environment with enhanced program security, and using execute-controlled programs, the initial program executed by a daemon must be defined to RACF with a profile in the PROGRAM class, and that profile must specify the MAIN option via the profile's APPLDATA. However, only programs loaded from an MVS library can be defined using the RACF PROGRAM class; you cannot define programs loaded from the z/OS UNIX file system. Therefore, if you have daemons that use execute-controlled programs, you need to move their initial program from the z/OS UNIX file system into an MVS library so that you can define it completely to RACF.

Additionally, you can choose whether to extend the enhanced program security protection to your UNIX daemons and servers that do not make use of RACF execute-controlled programs. You would enable this function by defining the profile BPX.MAINCHECK to RACF in the FACILITY class. Again, you would need to ensure that the initial program executed by your daemon or server resides in an MVS library and you would need to define it to RACF as a PROGRAM with the MAIN attribute.

3.2.3 More on UIDs and UID switching

To summarize: when a job starts or a user logs on to an application, the MVS user ID and password are verified by invoking existing z/OS and RACF functions. When the address space requests a z/OS UNIX function for the first time, RACF does the following:

- Verifies that the user is defined as a z/OS UNIX user
- Verifies that the user's current connect group is defined as a z/OS UNIX group
- Initializes the control blocks needed for subsequent security checks, both the MVS and the UNIX identities

A very important point to remember here is that a z/OS UNIX program authorized to switch to an alternate UID value on z/OS will also switch in most cases to the corresponding MVS user ID, in order to always conform to the dual identity as specified in the RACF USER profile.

Great care must be exercised when dealing with these UNIX programs since they can also access MVS resources. They are running on z/OS, with the switched-to MVS user ID. The use of BPX.DAEMON, as described above, is a primary step to protect against rogue UNIX programs with daemon authority.

As a reminder, Table 3-1 summarizes the conditions under which switching UID also implies switching the MVS user ID.

Table 3-1 UNIX UID and MVS user ID changes

USS function	Change effective UID	Change MVS user ID	Additional controls (RACF)[a]
setuid flag	Y	N	
su in shell	Y=0	N	User ID access to BPX.SUPERUSER
su with user ID and password	Y	Y	
su with user ID without password	Y	Y	User ID surrogate to target user ID
setuid() or _login with authentication of target ID	Y	Y	Clean address space
setuid() or _login without authentication of target ID	Y	Y	User ID surrogate of target or (user ID has UID(0) +access to BPX.DAEMON) + clean address space[b]
pthread_security_np with authentication of target ID	Y	Y	User ID permitted to BPX.SERVER + clean address space
pthread_security_np without authentication of target ID	Y	Y	User ID surrogate of target + permitted to BPX.SERVER + clean address space
_spawn() with identity change with authentication of target ID	Y	Y	
_spawn() with identity change without authentication of target ID	Y	Y	User ID surrogate of target or (user ID has UID(0) + access to BPX.DAEMON) + clean address space

a. Assumption: BPX.DAEMON is defined.
b. BPXROOT if target UID(0).

3.3 Limiting the amount of superusers

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

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

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

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

Superuser granularity

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

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

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

Table 3-2 shows the resource names that are used in the UNIXPRIV class of profiles and lists the privileges associated with each resource.

Note: We do not list all of the profiles in the UNIXPRIV class, only the ones that directly pertain to superuser granular privileges granted to regular users. For a complete description of all profiles in the UNIXPRIV class refer to *z/OS UNIX System Services Planning*, GA22-7800.

Table 3-2 Resource names in the UNIXPRIV class for z/OS UNIX privileges

Resource name	**z/OS UNIX privilege**	**Access required**
CHOWN.UNRESTRICTED	Allows all users to use the **chown** command to transfer ownership of their own files.	NONE
SUPERUSER.FILESYS.CHOWN	Allows users to use the **chown** command to change ownership of any file.	READ
SUPERUSER.FILESYS [1]	Allows users to read any HFS file and to read or search any HFS directory.	READ
	Allows users to write to any HFS file and includes privileges of READ access.	UPDATE
	Allows users to write to any HFS directory and includes privileges of UPDATE access.	CONTROL (or higher)
SUPERUSER.FILESYS. CHANGEPERMS	Allows users to use the **chmod** command to change the permission bits of any file and to use the **setfacl** command to manage access control lists for any file.	READ

Resource name	z/OS UNIX privilege	Access required
SUPERUSER.FILESYS.MOUNT	Allows users to issue the **`mount`** command with the nosetuid option and to unmount a file system mounted with the nosetuid option.	READ
	Allows users to issue the **`mount`** command with the setuid option and to unmount a file system mounted with the setuid option.	UPDATE
SUPERUSER.FILESYS.QUIESCE	Allows users to issue the **`quiesce`** and **`unquiesce`** commands for a file system mounted with the nosetuid option.	READ
	Allows users to issue the **`quiesce`** and **`unquiesce`** commands for a file system mounted with the setuid option.	UPDATE
SUPERUSER.FILESYS.PFSCTL	Allows users to use the pfsctl() callable service.	READ
SUPERUSER.FILESYS.VREGISTER [2]	Allows a server to use the vreg() callable service to register as a VFS file server.	READ
SUPERUSER.IPC.RMID	Allows users to issue the **`ipcrm`** command to release IPC resources.	READ
SUPERUSER.PROCESS.GETPSENT	Allows users to use the w_getpsent callable service to receive data for any process.	READ
SUPERUSER.PROCESS.KILL	Allows users to use the kill() callable service to send signals to any process.	READ

Resource name	z/OS UNIX privilege	Access required
SUPERUSER.PROCESS.PTRACE [3]	Allows users to use the ptrace() function through the dbx debugger to trace any process. Allows users of the **ps** command to output information about all processes. This is the default behavior of **ps** on most UNIX platforms.	READ
SUPERUSER.SETPRIORITY	Allows users to increase their own priority.	READ

Notes:

1. Authorization to the SUPERUSER.FILESYS resource provides privileges to access only local Hierarchical File System (HFS) files. No authorization to access Network File System (NFS) files is provided by access to this resource.
2. The SUPERUSER.FILESYS.VREGISTER resource authorizes only servers, such as NFS servers, to register as file servers. Users who connect as clients through file server systems, such as NFS, are not authorized through this resource.
3. Authorization to the resource BPX.DEBUG in the FACILITY class is also required to trace processes that run with APF authority or BPX.SERVER authority. For more information about administering BPX profiles, see *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800.

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

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

3.4 z/OS UNIX file systems: HFS and zFS

The comments in this section are included as a brief outline only. For a more complete description of these file structures, mounting, and other procedural concerns you must refer to the *z/OS UNIX System Services Planning* manual GA22-7800.

Note that HFS and zFS files can be shared between members of a Sysplex. This is further developed in 3.8, “The Sysplex shared file system” on page 93.

3.4.1 HFS basics

In this section we discuss HFS basics.

What the users see

The Hierarchical File System consists of the following:

- HFS files, which contain data or programs. A file containing a load module or shell script or REXX program is called an executable file. Files are kept in directories.
- Directories can contain files, other directories, or both. Directories are arranged hierarchically, in a structure that somewhat resembles an upsidedown tree. Using the tree as our symbol, the trunk is what we call the root directory, and that is at the top of our tree and the branches are at the bottom. The root is the first directory for the file system (at the top of the tree) and it is designated by a single slash (/) character.
- Additional local or remote file systems, which are mounted on directories of the root file system or of additional file systems, make up the remainder of the branches of the tree.

z/OS UNIX files are organized in a hierarchical file system (HFS), as in other UNIX systems. Files are members of a directory, and each directory is in turn a member of another directory at a higher level. The highest level of the hierarchy is the root directory. Each instance of the system contains only one root directory. Figure 3-5 shows the root and the directories that exist when the system is installed. In this example, under the /u directory are directories for UIDs, which are mounted automatically (in this example) using the AUTOMOUNT function.

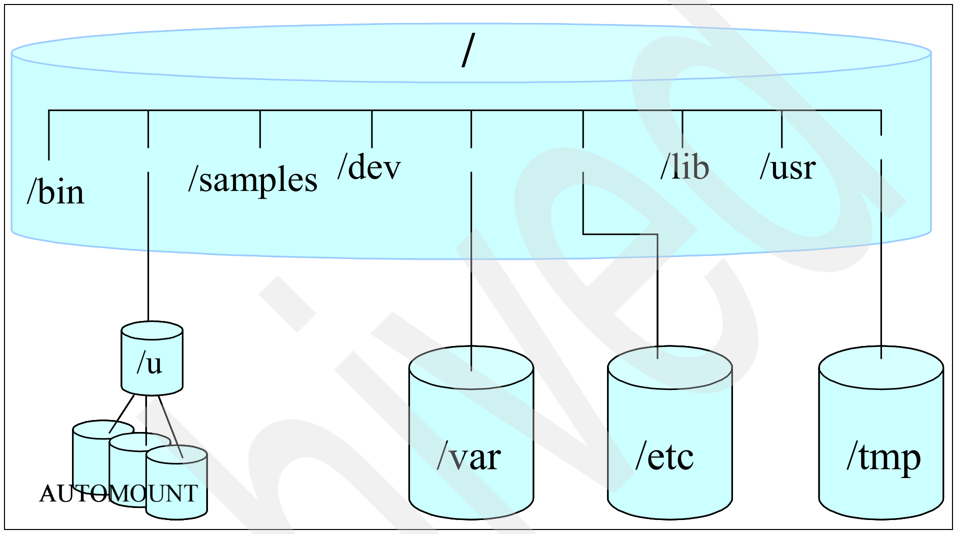


Figure 3-5 Example of a Hierarchical File System

How they are made up

HFS files and directories, along with metadata such as the File Security Packet, are kept in z/OS datasets allocated with a DSNTYPE of *HFS*.

The root file system is the starting point for the overall file structure. It contains the root directory and any related files or subdirectories. Figure 3-6 shows an example of a UNIX file tree contained in three z/OS HFS data sets. The logical connections between the HFS data sets are achieved by specifying *mount points*. Mounting a file system creates a binding for the duration of the mount. The binding is between a directory that is already in the file system hierarchy, called the mount point, and the entry point into the file system about to be mounted, called the root of this file system. The mount point directory and the root are connected until unmount time.

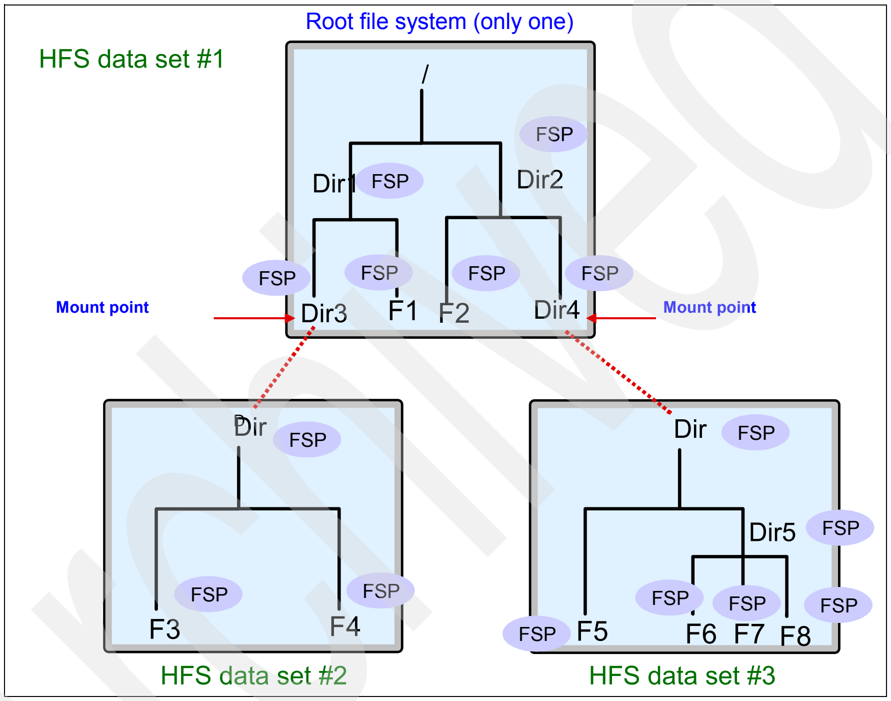


Figure 3-6 The z/OS UNIX HFS

During the first quarter of 2000, PTFs removed the requirement that file systems reside on SMS-managed volumes. File systems still need to be cataloged in the master or user catalog in order for the file systems to be mounted by z/OS UNIX.

Adding files to an existing HFS file system

If you need to add data to an existing HFS file system we recommend that you create separate MVS data sets for your HFS data and then mount them to your root file system or somewhere else in the hierarchy. How to do this is detailed in the *z/OS UNIX System Services Planning* manual GA22-7800, but before you begin it is important to state that the mount point should be an empty directory. If it is not, then its contents will be hidden for the duration of any subsequent mount.

In summary, you must perform the following steps to mount to a file system:

1. Identify the mount point for your data.
2. Allocate your additional HFS data set.
3. Free the data sets just created.
4. Logically mount the new file system in the directory previously identified by using the TSO/E MOUNT command under a user with mount authority.

Managing HFS

For users, you should logically mount HFS data sets on the root file system. You should also have your users place their directories and files in the mounted file systems. Separate user file systems offer several advantages:

- They improve storage management because the system administrator only needs to allocate data sets that are large enough to accommodate the needs of individual users.
- They enable failure isolation because the system administrator can unmount the user file system that caused an error without affecting other users' data or causing z/OS UNIX to fail.
- They relieve the contention for system resources that could occur by having multiple users in a single file system.

Name each user's home directory /u/*userid,* where *userid* is the RACF user ID in lowercase.

Keep system HFS data sets separate from user HFS data sets by segregating the HFS data sets on different volumes.

3.4.2 zFS basics

zFS is another UNIX file system that can be used in addition to HFS. It contains files and directories that can be accessed with Application Program Interfaces (APIs). They can also be mounted into the z/OS UNIX hierarchy along with other local or remote file systems types such as HFS, TFS, and NFS. Both zFS and HFS can be automounted, which means the automount facility automatically mounts file systems at the time they are accessed, and based on supplied criteria they are automatically unmounted when no longer in use.

zFS does not replace HFS. Rather, zFS is complementary to HFS.

How zFS differs from HFS

zFS and HFS are both UNIX file systems and both can participate in a shared file system. Although the application and command view of zFS is very similar to HFS, the administrator's view of zFS is different from HFS. zFS introduces the concept of an aggregate.

An aggregate is a container of zFS file systems. A zFS aggregate can contain one or more zFS file systems. An aggregate is a Virtual Storage Access Method Linear Data Set (VSAM LDS). Refer to *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394, for more information about defining VSAM Linear Data Sets. zFS aggregates and file systems are created using zFS utilities.

For zFS to be mounted like HFS datasets can be, only one file system may be defined inside the aggregate, and the file system's name must be the same as the aggregate name. This enables these file systems to be mounted without being attached because zFS can easily determine the data set name from the file system name. This is referred to as *HFS compatibility mode*.

For non-HFS compatible aggregates (that is, containing more than one file system), the aggregates must be *attached*. zFS aggregates are attached (that is, opened) when the zFS PFS is started. Attachment can also occur on the command **`zfsadm`**. The aggregates are listed in the IOEFSPRM member.

To exploit the new space sharing capability that zFS provides, you must use multi-file system aggregates. Multi-file system aggregates are created in a similar manner to compatibility mode aggregates except that no file systems are created when the aggregate is created. Space sharing allows a file system to use space that has been freed by another file system.

For example, if a file is removed from one file system, the other file system can create a file using that space. Each zFS file system in an aggregate has its own name (assigned when the file system is created).

Benefits of using a zFS

zFS provides the following features and benefits:

- Performance gains when accessing files approaching 8 K in size that are frequently accessed and updated. Access performance for smaller files is equivalent to that of HFS.
- Space sharing. Multiple zFS file systems can be defined in a single data set, enabling space that becomes available from erasing files in one file system to be available to other file systems in the same data set. This is an optional function that is available only in a non-shared environment.
- Read-only cloning of a file system in the same data set. The cloned file system can be made available to users to provide a read-only point-in-time

copy of a file system. This is an optional feature that is available only in a non-sysplex environment.

Security labels for zFS files and directories

The zFS file system is the only physical file system with full support for security labels in a multilevel-secure (MLS) environment. The HFS file system provides a very limited support for security labels in a multilevel-secure environment.

Therefore, if your sysplex has the need to implement MLS, then you must migrate your HFS system to a zFS system. Refer to *z/OS UNIX System Services Planning*, GA22-7800, which details how you can do that and which utilities IBM includes to perform and assist in that transition for you.

3.5 Access permission to UNIX files and directories

To be able to control access to UNIX files in z/OS, a UID has been assigned to the process or thread that tries to access the file. Typically the UID is taken from the OMVS segment of the RACF USER for the UNIX user that requests the access. Or this can be the UID of a default user defined in the BPX.DEFAULTUSER profile in the FACILITY class. Note that in this latter case the default UID is given only if the requestor's user profile does *not* have an OMVS segment. As already mentioned, the process is *dubbed* (that is, given a UID in addition to the MVS user ID) when it invokes a z/OS UNIX system service for the first time.

The contribution of RACF, at the time of file access, is as usual to provide an authorization advice that is to be enforced by the resource manager (that is, the UNIX System Service kernel for UNIX resources).

UNIX files are not protected by profiles in the RACF database. The permissions to access a file are metadata that are stored with the file. This is the File Security Packet (FSP). As shown in Figure 3-7, RACF is submitted by the FSP and the requesting user's User Security Packet (USP) and determines whether the access can be granted, returning the proper answer to the resource manager. RACF is also generating any appropriate auditing records.

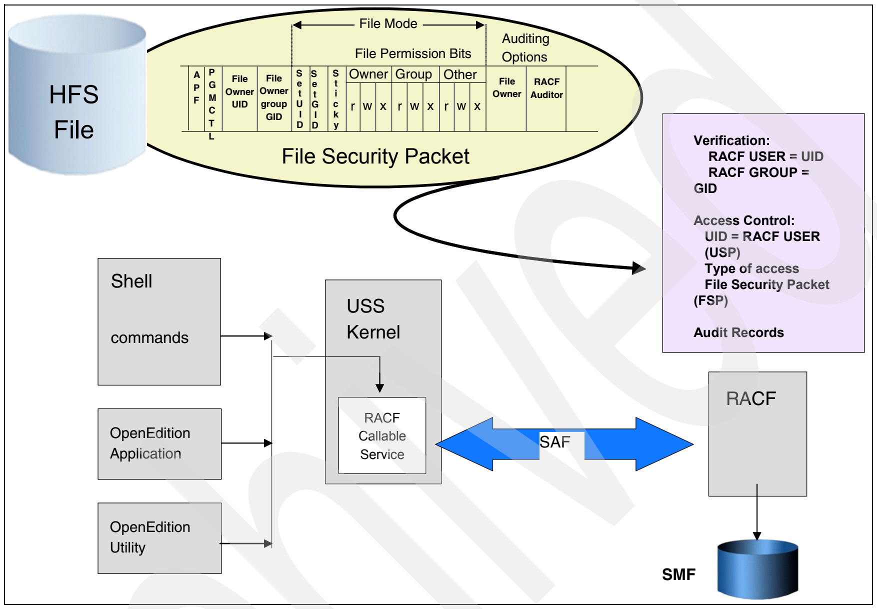


Figure 3-7 FSP and USP to RACF

The access control algorithm is based on the file permission bits, as shown in Figure 3-8, which are contained in the File Security Packet (FSP).

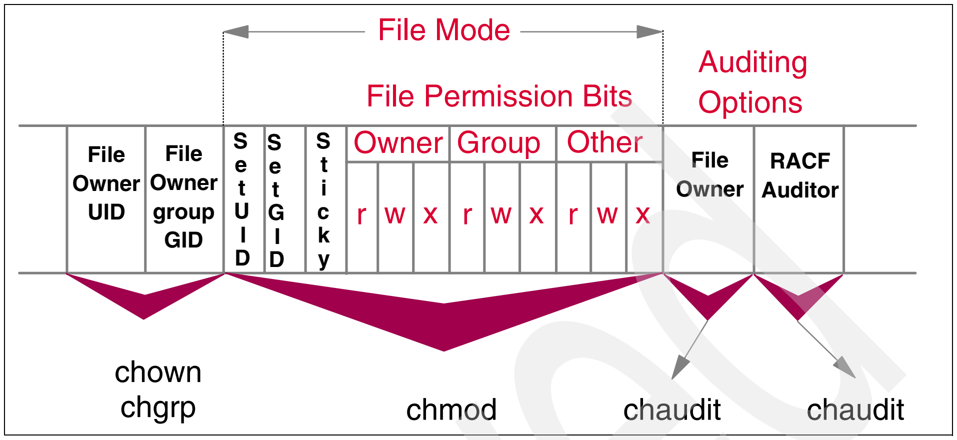


Figure 3-8   Hierarchical File System file security packet

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

- Owner

  Any processes with an effective UID that matches the UID of the file owner can access the file with the owner's defined permission.

  Note that the owner of a file may restrict himself to read and execute (r-x) access. A superuser can change the file owner UID in the FSP by issuing a **`chown`** shell command.

- Owning group

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

- Others

  This permission class is applied to any process that is not the file owner or a member of the owning group. It is commonly known as world access.

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

Table 3-3 is an overview of the types of access and the permissions granted by the access bits.

Table 3-3 z/OS UNIX files access permissions

Access type	Permission for file	Permission for directory
Read	Permission to read or copy the contents.	Permission to read, but not search, the contents.
Write	Permission to change, add to, or delete from the contents.	Permission to change, add, or delete directory entries (that is, HFS files, links, or subdirectories).
Execute	Permission to run the file. This permission is used for executable files.	Permission to search the directory.

Notes:

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

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

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

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

If RACF denies access, an error message will be logged to SYSLOG, as shown in Example 3-3.

Example 3-3 Resource access authorization error

```
ICH408I USER(IAN ) GROUP(SG247150) NAME(IAN HOLLAMBY IBM AUS)
  /u/boche/.sh_history
  CL(FSOBJ   ) FID(01D7C4C7D6C5F2000320000000260000)
  INSUFFICIENT AUTHORITY TO OPEN
  ACCESS INTENT(RW-)  ACCESS ALLOWED(GROUP ---)
```

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

- The first line identifies the user associated with the process or thread. USER is the RACF user ID, GROUP is the current connect group, and NAME is the user name in the user profile.
- The second line identifies the failing resource, in this case the full path name of an HFS file.
- In the third line, CL identifies the RACF class, in this case FSOBJ. This is one of the z/OS UNIX classes that is only used to hold auditing options, so do not look for any profiles in this class. FSOBJ means *file system object* (in other words, files). DIRACC would mean read/write access to a directory while DIRSRCH would appear if a directory search failed. Further details are provided in *z/OS Security Server RACF Security Administrator's Guide,* SA22-7683.
- The FID parameter is normally only of interest to the IBM service organization for debugging purposes.
- The fourth line contains the reason for the failure.
- In the fifth and last line, ACCESS INTENT, shows the access modes requested by the program. RW- in this case means that the program tried to open the file for both reading and writing. ACCESS ALLOWED shows the access level that would have been allowed. GROUP means that one of the groups the user is connected to matched the owning group, so the group permissions were used, and they were "---", which means no access was allowed.

3.6 HFS Access Control Lists (ACLs)

HFS files and directories are protected by POSIX permission bits, contained within the File Security Packet (FSP) also located in the HFS data set. That is to say that the permission bits are not kept in RACF profiles, and, incidentally, are transportable with the file or directory.

With the permission bits we cannot permit or restrict access to specific users or groups, which is felt to be a constraint when attempting to establish fine-grained access control.

The optional z/OS UNIX files' Access Control Lists are based on the traditional approach of associating to a resource a list of users and groups who can access the resource, in the mode (read, write, execute, or none) specified. As for the permission bits, they are contained within the file system, as opposed to being stored in RACF profiles.

The ACL, if present, is provided to RACF, along with the permission bits in the FSP, and are taken into consideration if the permission bits do not grant access. With ACLs, the access control algorithm behaves very much like the one RACF uses with the DATASET profiles access lists.

ACLs are managed using the z/OS UNIX specific commands **`setfacl`** and **`getfacl`**, but in order to use those one must either have UID(0), or be the file owner, or have at least read access to the SUPERUSER.FILESYS.CHANGEPERMS profile in the UNIXPRIV class (we explain below what the UNIXPRIV class of profiles is. These are actually the same requirements as for changing the permission bits.

ACLs can contain a maximum of 1024 entries, each composed of:

- A type: user or group
- An identifier: UID or GID
- The permissions to be given: read, write, or execute

ACL operational support

For ACL operational support:

- The ACLs support inheritance. We can establish a default or model ACL on a directory, which is automatically applied to new files and directories located in this directory. We can also have separate sets of defaults, one used for files and another used for directories,
- The ACL is automatically deleted if the file is removed.
- Several UNIX commands and ISHELL have been updated to support ACLs.

The reference book to use for the complete syntax of the UNIX System Services commands related to UNIX ACLs is *z/OS UNIX System Services Command Reference*, SA22-7802.

3.6.1 ACLs and RACF

RACF Access Checking with ACLs takes into account the base POSIX permission bit and the existing ACL. The full algorithm is rather complex and is shown in Figure 3-9. In any case, the book to refer to for additional information is *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683. Also refer to Appendix F, "Sysplex session-ID caching setup example" on page 257, which provides an extensive check list for identifying access problems.

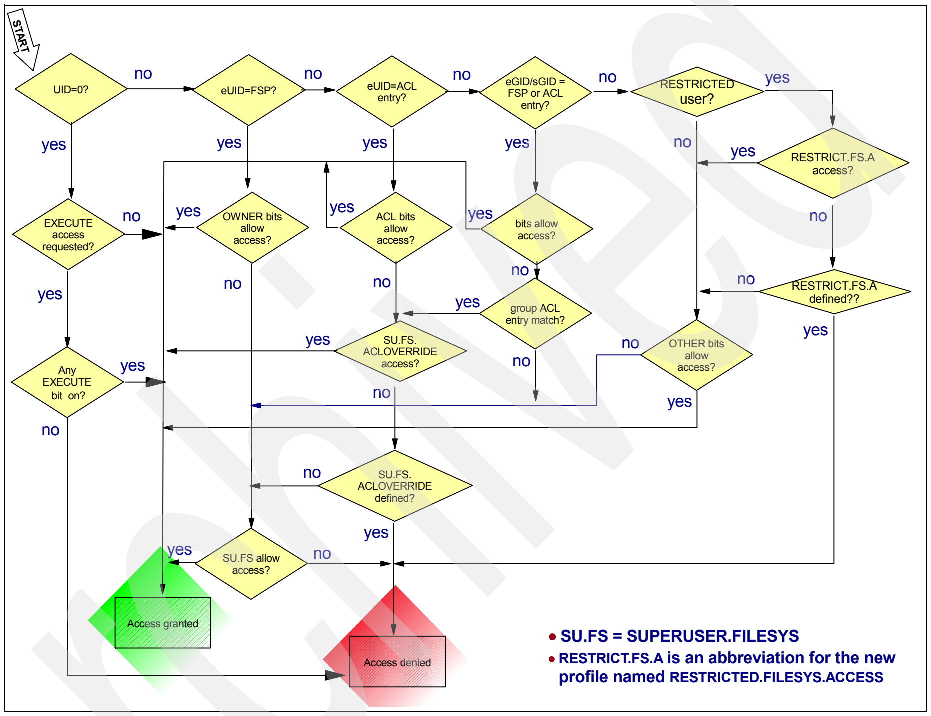


Figure 3-9 z/OS UNIX files Access Control Lists algorithm

The ACLs are used for access decisions only if the RACF FSSEC class is active, though setfacl can be used to create ACLs at any time, even when FSSEC is inactive.

Similar to RACF profile checking, user entry permissions take precedence over group entry permission, and if more than one group entry applies, the highest access is taken.

The basic file checking algorithm is used so each item below is checked in the specified order and checking stops at the first relevant item:

- Check *owner* bits.
- Check user ACL entries.
- Check union of *group* bits and group ACL entries.
- All entries are checked until a *single* entry grants the requested access.
- Check *other* bits.

Specific RACF profiles in the ACL decision flow

Specific RACF profiles influence the way access control decisions are made with ACLs. Below we provide a short explanation of these profiles.

SUPERUSER.FILESYS

SUPERUSER.FILESYS is a profile in the UNIXPRIV class to allow non UID(0) users to get extra privileges regarding handling of files and directories belonging to other users. Depending on the access a user is granted to on this profile:

- A user with READ access to the profile can read any local file and read or search any local directory.
- A user with UPDATE access can write to any local file, and includes in addition privileges of READ access.
- A user with CONTROL access can write to any local directory, and includes privileges of UPDATE access.

RESTRICTED.FILESYS.ACCESS

If the UNIXPRIV profile RESTRICTED.FILESYS.ACCESS is defined, then *other* bits are treated in the same way that UACC, GAT, and ID(*) are treated for users with the RESTRICTED attribute in their USER profile. That is, users with the RESTRICTED attribute cannot be granted file access via the other bits, whether or not an ACL exists.

If an exception to this rule has to be set for a specific RESTRICTED user, give a READ permission to RESTRICTED.FILESYS.ACCESS to this user or one of the groups it is connected to. Note that in this case the privilege, if any, provided by SUPERUSER.FILESYS in the UNIXPRIV class applies to this RESTRICTED user.

SUPERUSER.FILESYS.ACLOVERRIDE

If the UNIXPRIV profile SUPERUSER.FILESYS.ACLOVERRIDE is defined, then SUPERUSER.FILESYS authority is treated like the OPERATIONS attribute during RACF data set profile checking (that is, it prevents users from using their SUPERUSER.FILESYS authority to access file system resources if the users are specifically unauthorized to access through the ACL).

For exception cases, the user or group has to be permitted to SUPERUSER.FILESYS.ACLOVERRIDE with whatever access level is also required for SUPERUSER.FILESYS.

SUPERUSER.FILESYS.ACLOVERRIDE is checked only if an ACL entry match was found for the user or one of the groups the user is connected to, and no ACL entry granted the requested access.

Note: The SUPERUSER.FILESYS.ACLOVERRIDE profile introduces to the HFS files and directories access control an optional OPERATIONS-like behavior (that is reproducing RACF behavior when dealing with a requestor with the OPERATIONS attribute). As such, it only makes sense to check this profile if a *denying* ACL entry is encountered during ACL processing. If you simply define this profile with UACC(NONE), then you have created a system switch to enforce the OPERATIONS-like behavior. This can be overridden for individual users/groups by granting the access that would have been required for SUPERUSER.FILESYS.

If you do not wish to implement this behavior, then you have no need to define the SUPERUSER.FILESYS.OVERRIDE profile (and you will save some path length by not doing so).

3.6.2 ACL auditing and reporting

We can set LOGOPTIONS (ALWAYS) on the FSSEC class to audit on all ACL modifications. There are now two new SMF Type 80 event codes for setfacl (75) and delfacl (76). For setfacl one audit record is created for each added, modified, and deleted ACL entry. For delfacl only one record is created.

The ICH408I message and file access SMF record are updated to indicate if an ACL entry was used to determine permissions, failure due to inaccessibility of the ACL, or because a user is RESTRICTED.

3.7 z/OS UNIX System Services and Multilevel Security

In a multilevel-secure z/OS UNIX environment, authorization checks are performed for security labels in addition to POSIX permissions, to provide additional security. Refer to 2.4, “An overview of Multilevel Security (MLS)” on page 46, for an introductory overview of MLS.

3.7.1 Associating security labels with remote users

In a z/OS UNIX environment, a user might be entering the system from a remote IP address using an application such as rlogin. If the SECLABEL class is active, there are two ways that a security label can be associated with a remote user:

- The system attempts to derive a security label from the user's port of entry. The system determines the security label associated with the user's IP address, and whether the user is authorized to use the security label.
- If a port of entry is not available, the system uses the application's current security label to determine access for the remote user.

Profiles in the SERVAUTH class represent IP addresses and require security labels if the MLACTIVE option is active. Mandatory access checks for profiles in the SERVAUTH class are equivalence checks.

3.7.2 Assigning a home directory and initial program depending on security label

Because a user can use different security labels at different times, a z/OS UNIX user might need to have different home directories and initial programs, depending on the security label in use. The user's home directory and initial program are specified in the OMVS segment of the user's profile, in the HOME and PROGRAM fields. You can set up a special symbolic link and use it in the HOME and PROGRAM fields to provide different values depending on the user's current security label. One or more symbolic links of this type can reside in various directories, so that when they are encountered in path names the security label is dynamically substituted into the path name when it is resolved. The contents of these symbolic links must be in one of the following formats:

- $SYSSECA/ or $SYSSECA/pathname

  The symbol $SYSSECA indicates that the user's current security label should be substituted into the path name as an absolute directory name. This means that path name resolution continues at the ROOT with a directory name of the user's current security label.

  For example, assume that the OMVS segment of a user's profile defines the user's home directory to be /u/secsyma/ussuser, and /u/secsyma is a symbolic link defined to have the contents $SYSSECA/. If the user logs on with the security label ABCSEC, the user's home directory resolves to /ABCSEC/ussuser.

- $SYSSECR/ or $SYSSECR/pathname

  The symbol $SYSSECR indicates that the user's current security label should be substituted into the path name as a relative directory. This means that path name resolution continues in the directory in which the symbolic link is encountered, with a directory name of the user's current security label.

For example, assume that the OMVS segment of a user's profile defines the user's home directory to be /u/secsymr/ussuser, and /u/secsymr is a symbolic link defined to have the contents $SYSSECR/. If the user logs on with the security label ABCSEC, the user's home directory resolves to /u/ABCSEC/ussuser.

Note: The security label name is substituted in upper case. The directories in the file system must be defined with the security label in upper case, because directory names are case-sensitive.

If a user logs on without a security label, when security label substitution is performed it replaces the $SYSSECA/ and $SYSSECR/ symbols with a period (.), allowing path name resolution to proceed from the ROOT or from the current path name directory. For example:

- If /u/secsyma is a symbolic link defined to have the contents $SYSSECA/, and if the user's home directory is /u/secsyma/ussusr, if the user logs on without a security label, the home directory resolves to /./ussuser or /ussusr.
- If /u/secsymr is a symbolic link defined to have the contents $SYSSECR/, and if the user's home directory is /u/secsymr/ussuser, and the user logs on without a security label, the path resolves to u/./ussusr or /u/ussusr.

Security label substitution is performed only if the SECLABEL class is active. When the SECLABEL class is not active, the $SYSSECA/ and $SYSSECR/ symbols are not substituted, but instead are left in the path name as relative directory names.

Note that in order for a user's home directory to be protected by a security label and contain objects that have security labels, it must reside in a zFS file system. Exception: If the home directory is in an HFS file system mounted in read-only mode, in some cases the system assumes a security label for it.

3.7.3 Security labels for z/OS UNIX files and directories

z/OS UNIX files and directories can be protected by security labels. The security label for a file or directory is stored in the file security packet (FSP) for the file or directory.

The SETROPTS MLFSOBJ option controls whether security labels are required for z/OS UNIX files and directories.

The zSeries file system (zFS) and the hierarchical file system (HFS) are both z/OS UNIX file systems that can be used with security labels. However, there are restrictions for using security labels with HFS files in a multilevel-secure environment.

3.7.4 Assumed security labels

z/OS UNIX assumes a security label for a file system if the following conditions are met:

- A DATASET profile that specifies a security label protects the file system data set.
- You mount the file system in read-only mode.
- The RACF MLFSOBJ option is active when you mount the file system.
- The root directory of the file system does not already have a security label.

z/OS UNIX determines the assumed security label when the file system is mounted in read-only mode, from the security label specified in the data set profile, and assumes it only for the duration of the mount. After unmount, the file system again has no security label.

If you update the security label of the file system data set, the next time you mount the file system in read-only mode the assumed security label for the file system reflects the new security label for the data set. The assumed security label can vary upon mount according to the value of the security label specified in the profile for the containing aggregate. In contrast, a real security label assigned to a file system cannot be changed.

3.7.5 Security labels for zFS file systems and their contents

The zSeries file system (zFS) supports security labels. A zFS file system is contained in a VSAM linear data set. The security label of the root within each zFS file system data set is determined at the time the file system aggregate (container) is allocated, from the security label of the profile in the DATASET class that covers the aggregate.

If the SECLABEL class is active when allocating the aggregate, all file systems subsequently created within that aggregate contain a root with the security label that is specified in the profile for that aggregate. If no profile exists for the aggregate, or if it exists but does not specify a security label, then:

- If the MLFSOBJ option is not active the root for any file systems within that aggregate have no security label.
- If the MLFSOBJ option is active, requiring security labels on all file system objects, the user's security label is assigned to the root of the file system.

Important: If a zFS file system or aggregate has a security label, changing the security label specified in the data set profile does not change the security label of any zFS file systems contained within it. Once a file system is created with a security label it has that security label forever. (However, if a zFS file system does not have a security label and is mounted read-only, the security label that z/OS UNIX assumes for it can change).

If the SECLABEL class is active, when a new z/OS UNIX file or directory is created within a zFS file system, the system assigns it a security label as follows:

- If the parent directory has no security label, the new file or directory is not assigned a security label.
- If the security label of the parent directory is SYSMULTI, the security label of the new file or directory is set to the security label of the requesting address space (the user). If the user has no security label (this could occur only if the MLACTIVE option is not active), the new file or directory is not assigned a security label.
- If the security label of the parent directory is not SYSMULTI, the security label of the new file or directory is set to the security label of its parent directory.

If a z/OS UNIX file, directory, or symbolic link was created in a zFS file system without being assigned a security label (for example, if the SECLABEL class was not active when the file, directory, or symbolic link was created), the security administrator can assign a security label to it using the chlabel `shell` command.

Once a file or directory has been assigned a security label, there is no way to delete or change the security label assigned to it other than copying the file or directory to another directory with a different security label.

3.7.6 Security labels for HFS file systems

The hierarchical file system (HFS) does not fully support security labels and multilevel security.

Rule: In a multilevel-secure environment, use HFS file systems only in read-only mode. If you want to use files from an HFS file system in read-write mode, and use security labels in the file system, you must copy or move those files to a zFS file system.

3.8 The Sysplex shared file system

By establishing the shared file system environment, sysplex users can access data throughout a single shared file hierarchy from any member in the sysplex.

The sysplex shared HFS facility operates according to the *function shipping* model. That means one member in the sysplex is the owner of the file system to be shared between the sysplex members. As such, it is assigned as the mount coordinator for this particular file system, and in some cases, the system coordinating I/O for the file system. The file system owner is the only sysplex member to have direct access to the PFS. The other sysplex members are clients that are sharing the file system through XCF communication with the owner. Functional characteristics of the shared HFS implementation are:

- All mounted file systems are to be visible from all members in the sysplex, with the ability to read from and write to those file systems.
- The solution utilizes XCF services to transfer data to and from a file system owner, where the request can be processed.
- Dead system recovery is provided to dynamically recover file systems owned by a member that has left the sysplex for any reason. Those file systems will be moved to a surviving member of the sysplex, providing enhanced file system availability.

There needs to be one OMVS couple data set (BPXMCDS) defined for the sysplex. This data set is used as the sysplex-wide mount table. It contains information about all systems participating in file system sharing and records for all mounted file systems in the sysplex. This data is updated by members as they perform mount, unmount, quiesce, unquiesce, chmount (move file system), and dead system recovery requests.

All systems in the sysplex use this data set to keep in sync with file system status. As systems perform mounts, they add records to the couple data set, indicating the mount point, file system owner, and so forth. Other systems in the sysplex are then sent messages (via XCF services) telling them to check the couple data set for changes. Those changes are then processed on the other systems, allowing them to maintain a common file system hierarchy.

Figure 3-10 illustrates the implementation of the Shared HFS.

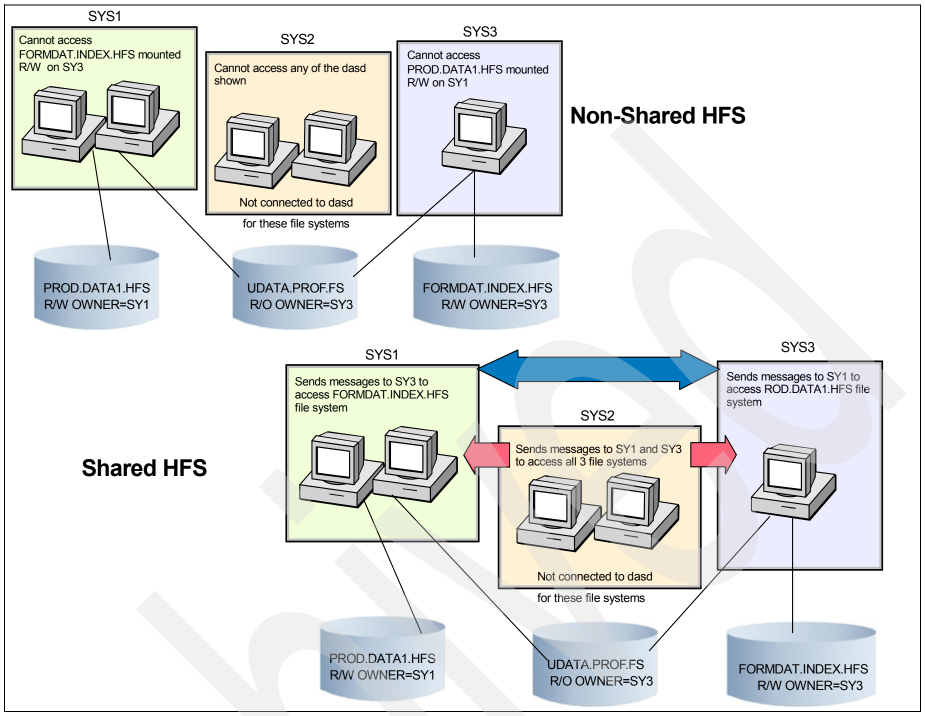

SYS1
Cannot access FORMDAT.INDEX.HFS mounted R/W on SY3
SYS2
Cannot access any of the dasd shown
Not connected to dasd for these file systems
SYS3
Cannot access PROD.DATA1.HFS mounted R/W on SY1
Non-Shared HFS
PROD.DATA1.HFS R/W OWNER=SY1
UDATA.PROF.FS R/O OWNER=SY3
FORMDAT.INDEX.HFS R/W OWNER=SY3
Shared HFS
SYS1
Sends messages to SY3 to access FORMDAT.INDEX.HFS file system
SYS2
Sends messages to SY1 and SY3 to access all 3 file systems
Not connected to dasd for these file systems
SYS3
Sends messages to SY1 to access ROD.DATA1.HFS file system
PROD.DATA1.HFS R/W OWNER=SY1
UDATA.PROF.FS R/O OWNER=SY3
FORMDAT.INDEX.HFS R/W OWNER=SY3

Figure 3-10 Sysplex Shared HFS.

Note: The Sysplex shared file system can comprise both HFS and z/FS files.

3.8.1 Procedures for establishing a shared file system in a sysplex

The following Web site provides an interesting animated overview of how to establish a shared file system in a sysplex:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/animations/ussanims.html`

3.8.2 The Shared HFS structure

The example supplied below refers to the latest structure of a shared HFS system available with z/OS V1R7. It uses a root file system called *version file system*.

The version file system is the IBM-supplied root file system. To avoid confusion with the sysplex root file system, *root file system* has been renamed to *version* *file system*. UNIX System Services creates a directory with the value *nnnn* specified on the VERSION parameter in the BPXPRMxx member and uses this directory as the mount point for the specified version data set.

Figure 3-11 shows a sample setup for two systems using the same release level of the HFS data set.

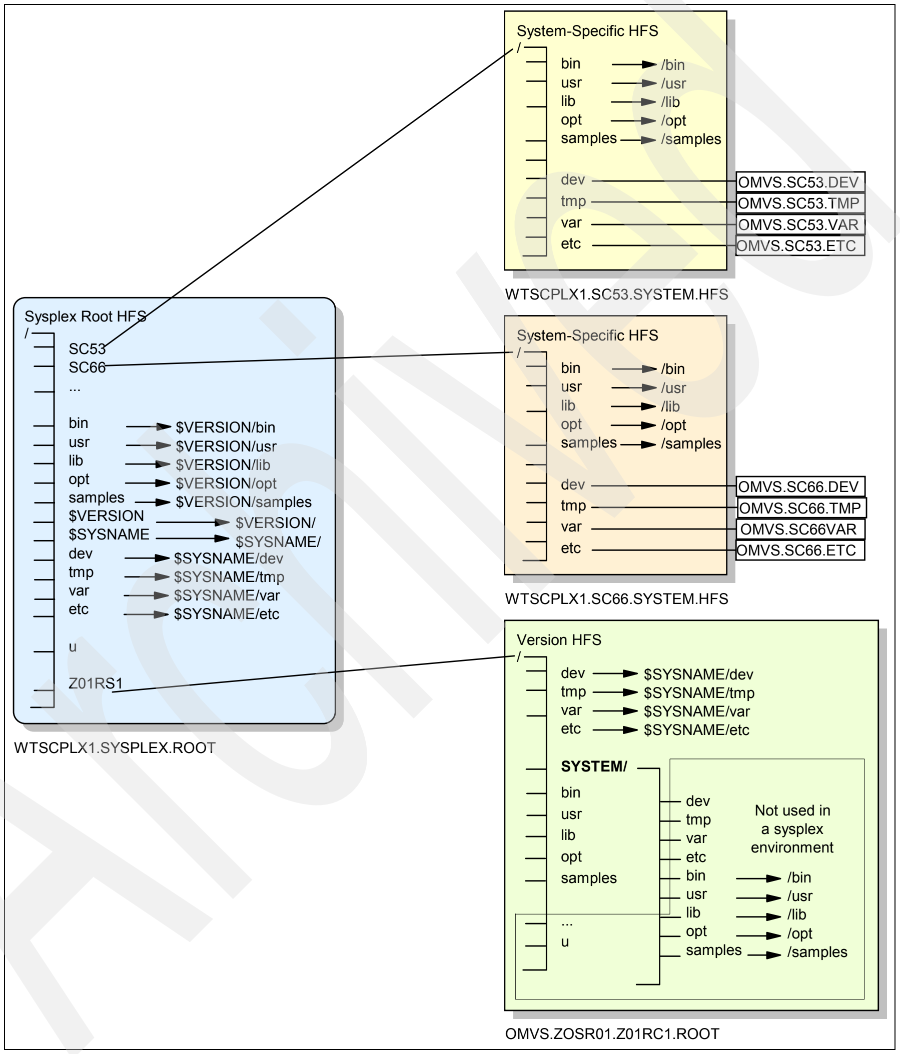


Figure 3-11 Shared HFS setup

Example 3-4 shows the changes to the BPXPRMxx member.

Example 3-4 Shared HFS setup in BPXPRMxx member with one version HFS for entire sysplex

```
FILESYSTYPE TYPE(HFS)
            ENTRYPOINT(GFUAINIT)
            PARM(' ')

VERSION('&SYSR1.')
SYSPLEX(YES)

.
.

ROOT  FILESYSTEM('&SYSPLEX..SYSPLEX.ROOT')
      TYPE(HFS) MODE(RDWR)

MOUNT FILESYSTEM('&SYSPLEX..&SYSNAME..SYSTEM.HFS')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('/&SYSNAME.')

MOUNT FILESYSTEM('OMVS.ZOSR01.&SYSR1..HFS')
      TYPE(HFS) MODE(READ)
      MOUNTPOINT('/$VERSION')

MOUNT FILESYSTEM('OMVS.&SYSNAME..DEV')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('&SYSNAME./dev')

MOUNT FILESYSTEM('OMVS.&SYSNAME..ETC')
      TYPE(HFS) MODE(RDWR)
      NOAUTOMOVE
      MOUNTPOINT('&SYSNAME./etc')

.
.
```

Additional information regarding the setup of the shared file system

Mount the version file system *read-only* in a sysplex environment, and designate it AUTOMOVE. The mount point for the version file system is dynamically created if the VERSION statement is used in BPXPRMxx.

Refer to *z/OS UNIX System Services Planning,* GA22-7800, for more information about how to set up UNIX System Services and shared HFS.

3.8.3 Security considerations with the shared HFS implementation

The following are some security considerations:

- Visibility of the common file system hierarchy and files by all contributing members. This is the objective when setting up a sysplex shared HFS, but the user has to know that there is no way to hide from the rest of the sysplex files or directories that are part of a mounted HFS, whatever the mount point is. The only access control left is the UNIX permission bits and the Access Control List. If it is an unacceptable security exposure to share a common file system hierarchy with a sysplex member that can be exposed from the Security standpoint (such as a sysplex member connected to a non-secure network), then there are several alternatives:
  - IPL the exposed member with SYSPLEX(NO) in BPXPRMxx, meaning this member cannot participate in HFS data sharing.
  - Use a configuration where the exposed system is not part of the secure sysplex. This approach of establishing sysplexes in different security zones is further discussed in Chapter 8, “Miscellaneous network-related considerations” on page 195.
- Ownership of file system. One may want to restrict file system ownership given to our exposed member so that a compromise of the exposed member would not adversely affect servicing access to the file system. This can be done at MOUNT time by mounting to a particular system using the SYSNAME option specifying a non-exposed member. However, the AUTOMOVE function, if enabled, may unexpectedly grant ownership to the exposed member. This case is developed in the next section.
- Because of the importance, not only of the root to the structure of the shared file system but also of the expanded file system through the links you establish with mounts, you must ensure the MVS datasets that store the shared file system files and directories are well protected via the RACF DATASET class from general users and the potential of accidental corruption.

Controlling who owns the file system

The SYSNAME option of the MOUNT command allows particular mounts to be owned by particular systems in your sysplex. You may want to mount a file system on the system where most of the users of that file system will be running to lessen the messaging traffic between systems, or you can specify the SYSNAME to control which member is initially getting ownership of the file system.

This ownership can be modified later on using the SETOMVS or **`chmount`** commands. However, since recovery of a file system whose owning system has left the sysplex will allow for higher levels of availability across the sysplex, you may wish to specify the AUTOMOVE option for a particular mounted file system, which tells whether automatic recovery should be performed for that file system if its owning system leaves the sysplex, for instance if it has crashed. The default for this option, in its various command-dependent forms, is AUTOMOVE (as opposed to NOAUTOMOVE) to recover the file system to a new owning system.

Shared file system environment and system-specific security labels

If you have a sysplex and are using shared file systems, you might get unexpected results if you define system-specific security labels and activate the RACF SECLBYSYSTEM option. Access checks might succeed or fail in ways that you do not expect if a user is running on a system other than the system where the data is owned, and either of the systems has security labels defined that are not defined on the other system.

If you want to use the SECLBYSYSTEM option in a shared file system environment, be aware of the following:

- Security checking for most operations occurs on the system where the user is running. For a shared file system client most checking is done on the client system. However, security checking for the name-hiding function for a command that lists the contents of a directory is done on the system that owns the data, and that might not be the system where the user is running.
- If you use AUTOMOUNT or AUTOMOVE, a file system might be moved to a system on which one or more of its security labels is not defined, causing data to not be available. You can use the AUTOMOVE system list (SYSLIST) to ensure that if the owning system leaves the sysplex, the file system is not moved to a system on which a security label contained in the file system is undefined. The SYSLIST option of AUTOMOVE is not supported for automounted file systems, but should not be necessary. AUTOMOUNT unmounts the file system if it is not in use from another system, or automatically moves it to a system where it is currently being used.
- If an object has a SYSMULTI security label, RACF grants a subject access to the object without doing further checks on the subject's security label, because SYSMULTI is equivalent to any other security label. As a result, in a shared file system environment with SECLBYSYSTEM active, if the user's security label is not defined on the system on which the access check is being done, RACF does not determine that the security label is undefined.

Guidelines

If you want to activate the RACF SECLBYSYSTEM option in a shared file system environment, you can minimize problems by doing the following:

- Mount a file system only on a system on which all security labels contained in the file system are defined. For example, if you have a sysplex with members SYSA, SYSB, and SYSC, and you have defined the security label •XYZ• to be valid only on SYSA and SYSB, do not mount a file system with the security label •XYZ• on SYSC or use SYSC as a backup for SYSA or SYSB.

- When mounting a file system, use the AUTOMOVE system list (SYSLIST) to ensure that if the owning system leaves the sysplex, the file system is not moved to a system on which a security label contained in the file system is undefined.
- Define system-specific security labels to be disjoint from security labels to be used on other systems, especially when the name-hiding function is in effect (the RACF MLNAMES option is active). Doing this ensures that the file system can be used from only one system, and will not automove.
- Do not use system-specific security labels (the RACF SECLBYSYSTEM option is active) for automount-managed file systems.
- If the RACF SECLBYSYSTEM option is active and you define security labels that are equivalent, define them to be active on the same systems.

3.8.4 The AUTOMOVE function

The AUTOMOVE / NOAUTOMOVE option of MOUNT is used to indicate what happens to the file system if the owner of that file system goes down.

- AUTOMOVE specifies that ownership of the file system is automatically moved to another system. It is the default.
- NOAUTOMOVE specifies that the file system will not to be moved if the owning system goes down, and the file system is not accessible until the failing owner recovers.

When mounting file systems in the sysplex, you can specify a prioritized AUTOMOVE system list to indicate what members the file system should or should not be moved to when the owning member leaves the sysplex. There are different ways to specify the AUTOMOVE system list. Some of them are:

- On the MOUNT statement in BPXPRMxx, specify the AUTOMOVE keyword, including the indicator and system list. in the format `(`*`indicator`*`,name1,name2,...,nameN)`, where *indicator* is either INCLUDE or EXCLUDE, which can also be abbreviated as I or E.

  For the TSO MOUNT command, specify the AUTOMOVE keyword, including the indicator and system list.

  ```
  AUTOMOVE  | NOAUTOMOVE  |  UNMOUNT |
  AUTOMOVE(indicator,name1,name2,...,nameN)
  ```

- Use the mount shell command:

  ```
  mount [-t fstype] [-rv] [-a yes | no | unmount | indicator,sysname1,...
  sysnameN]
  ```

You should define your sysplex root file system data as AUTOMOVE to ensure that the root is always available. You should also define your system-specific file systems as UNMOUNTED, so that the file system will be unmounted when the member leaves the sysplex.

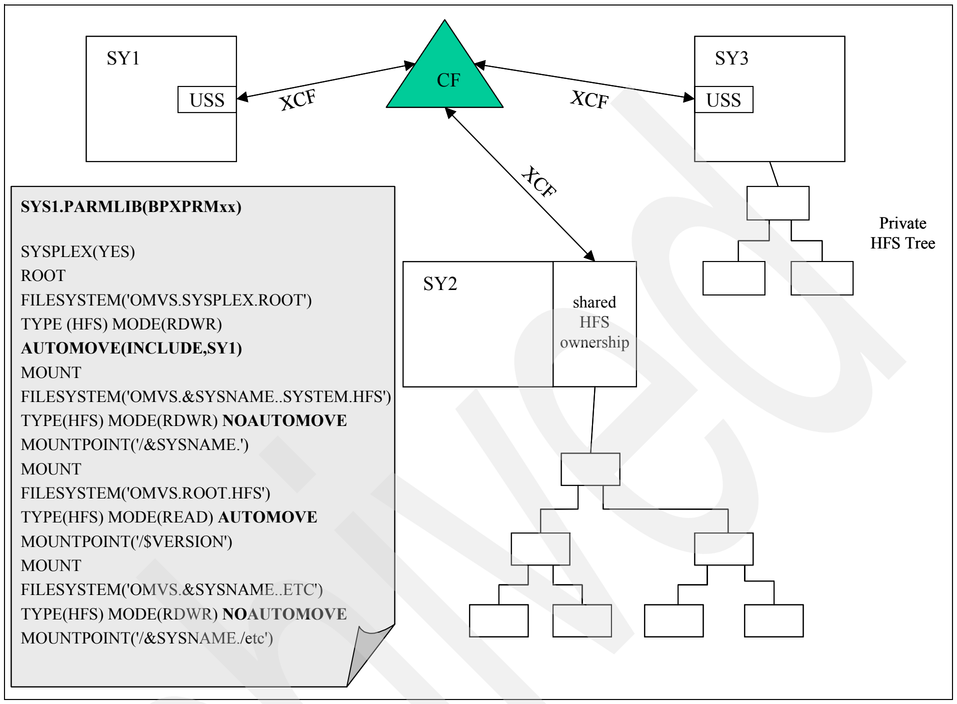
When AUTOMOVE is specified *without* the INCLUDE / EXCLUDE keyword, recovery of the file system will be performed to a randomly selected owner when the current owner fails.

AUTOMOVE with INCLUDE / EXCLUDE lists specifies managed recovery of the file system if the current owner fails.

- Use the EXCLUDE list to prevent recovery of a file system from transferring ownership to a particular member or a set of members in the participating group. When the current owner fails, recovery of the file system is performed by randomly selecting a new owner outside the EXCLUDE list.
- Use the INCLUDE list to ensure that recovery of a file system will transfer ownership only to a particular member or any member in a set of members in the participating group. Recovery of the file system is performed in priority order only by the list of systems specified in the INCLUDE list.

The systems specified in the INCLUDE list should have physical I/O paths to the volume where the file system resides. If not, the file system becomes unowned. The file system still exists in the file system hierarchy so that dependent file systems that are owned by another system are still usable. However, all file operations for the unowned file system will fail until a new owner is established.

Figure 3-12 describes a shared HFS implementation and the corresponding AUTOMOVE directives in the BPXPRMxx member.



```
SYS1.PARMLIB(BPXPRMxx)

SYSPLEX(YES)
ROOT
FILESYSTEM('OMVS.SYSPLEX.ROOT')
TYPE (HFS) MODE(RDWR)
AUTOMOVE(INCLUDE,SY1)
MOUNT
FILESYSTEM('OMVS.&SYSNAME..SYSTEM.HFS')
TYPE(HFS) MODE(RDWR) NOAUTOMOVE
MOUNTPOINT('/&SYSNAME.')
MOUNT
FILESYSTEM('OMVS.ROOT.HFS')
TYPE(HFS) MODE(READ) AUTOMOVE
MOUNTPOINT('/$VERSION')
MOUNT
FILESYSTEM('OMVS.&SYSNAME..ETC')
TYPE(HFS) MODE(RDWR) NOAUTOMOVE
MOUNTPOINT('/&SYSNAME./etc')
```

Figure 3-12 Sample AUTOMOVE configuration in BPXPRMxx

Using the asterisk as a wildcard

When you specify the LIST keyword with the AUTOMOVE operand, you can also use an asterisk (*) character to act as a wildcard instead of nominating a long list of systems.

Example: If you have a large number of systems in your sysplex, you can specify only the systems that should have priority and use a wildcard to indicate the rest of the systems.

```
AUTOMOVE(INCLUDE,SY1,SY2,...*)
```

At first glance, AUTOMOVE(INCLUDE,*) appears to work the same way as AUTOMOVE(YES) because all of the systems will try to take over the file system.

However, if none of the systems can take over the file system with AUTOM0VE(INCLUDE,*) coded, it will be unmounted. If AUTOMOVE(YES) is used, the file system will become unowned.

The following restrictions apply to the use of the wildcard:

- You can use the wildcard support on all methods of mounts, including the MOUNT statement in BPXPRMxx, the TSO MOUNT command, the mount shell command, the ISHELL MOUNT interface, the MNTE_SYSLIST variable for REXX, C program, and assembler program.
- The wildcard is only allowed in an INCLUDE list.
- The wildcard is not allowed in an EXCLUDE list.
- The wildcard must be the last item (or the only item) in the system list.
- Do not use an asterisk in a mixed SYSPLEX environment where any system is running below z/OS V1R6. Doing so will produce unpredictable results.
- Do not use the INCLUDE / EXCLUDE option in a shared file system environment with releases running below z/OS V1R4. Only use this option on mounts of file systems that are critical to operation across a subset of systems in the participating group, or when you do not want certain systems in the participating group to have ownership of the file system.

  If recovery processing fails to establish a new owner for the file system, the file system is unmounted, along with all the file systems mounted within it.

When specifying the AUTOMOUNT delay time in a shared file system, do not use the default AUTOMOUNT delay time of 0. Instead, specify a delay time of at least 10.

Benefits of using AUTOMOVE with a system list

Using an AUTOMOVE system list, you can:

- Provide predictability in which systems will take over a given file system. This feature provides you with additional control when you need to take into consideration Security, workload balancing, and performance issues with the remaining members of your sysplex.
- Be able to exclude certain systems from taking over a given file system.

Implications during system failures and recovery

File system recovery in a shared file system environment takes into consideration file system specifications such as AUTOMOVE, NOAUTOMOVE, UNMOUNT, and whether or not the file system is mounted read-only or read-write.

Generally, when an owning system fails, ownership over its automove-mounted file system is moved to another system and the file is usable. However, if a file system is mounted read-write and the owning system fails, then all file system operations for files in that file system will fail. This happens because data integrity is lost when the file system owner fails.

File systems that are mounted NOAUTOMOVE will become unowned when the file system owner exits the sysplex. The file system will remain unowned until the original owning system restarts or until the unowned file system is unmounted. Note that since the file system still exists in the file system hierarchy, the file system mount point is still in use.

File systems that are mounted below a NOAUTOMOVE file system will not be accessible via path name when the NOAUTOMOVE file system becomes available.

3.8.5 Root isolation

Shared HFS provides a sysplex-wide root HFS. No files or code reside in the sysplex root data set. It consists of directories and symbolic links only.

Each system in a shared HFS environment still has its own common HFS data sets for /etc, /tmp, /var, and /dev. The system-specific HFS data set contains the directory entries for the /etc, /tmp, /var, and /dev mount points.

The sysplex root is a file system that is used as the sysplex-wide root. This file system must be mounted read-write and designated AUTOMOVE. Only one sysplex root is allowed for all systems participating in a shared file system.

3.9 Additional BPX.* FACILITY class profile

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

- BPX.CONSOLE

  Allows a permitted user the ability to use the _console() or _console2() services.
- BPX.DAEMON.HFSCTL

  Controls which users with daemon authority are allowed to load uncontrolled programs from MVS libraries into their address space.
- BPX.DEBUG

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

  This controls which users are allowed to set the APF authorized attribute for an HFS file. This authority allows the user to create a program that will run

APF authorized. This is similar to the authority of allowing a programmer to update SYS1.LINKLIB or SYS1.LPALIB.

- BPX.FILEATTR.PROGCTL

  This controls which users are allowed to set the program-controlled attribute for an HFS file. Programs marked with this attribute are recognized as program controlled by the system when loaded into storage.

- BPX.FILEATTR.SHARELIB

  Shared libraries allow multiple processes to share subroutines in object libraries more efficiently.

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

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

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

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

- BPX.SERVER

  Restricts the use of the pthread_security_np() service. A user with at least READ or WRITE access to the BPX.SERVER FACILITY class profile can use this service. It creates or deletes the security environment for the caller's thread.

  This profile is also used to restrict the use of the BPX1ACK service, which determines access authority to z/OS resources. Servers with authority to BPX.SERVER must run in a clean program-controlled environment. z/OS

UNIX will verify that the address space has not loaded any executables that are uncontrolled before it allows any of the following services that are controlled by z/OS UNIX to succeed:

- seteuid
- setuid
- setreuid
- pthread_security_np()
- auth_check_resource_np()
- _login()
- _spawn() with user ID change
- _password()

- BPX.SMF

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

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

- BPX.STOR.SWAP

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

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

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

- BPX.WLMSERVER

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

  - QuerySchEnv()
  - CheckSchEnv()

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

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

Note: The full set of BPX profiles is described in *z/OS UNIX System Services Planning*, GA22-7800.

4

Sysplex Workload Management and Security

In this chapter we review basic concepts of the Workload Manager and the advantages of running WLM in a sysplex environment. Although not directly addressing the security setup of a sysplex, WLM can become a major player in maintaining the system's availability under attack conditions such as a denial of service attempt launched against a sysplex member. (Refer to 1.1, "Security" on page 2 for an explanation of the denial of service attack.)

In 5.4.2, "How WLM can limit the resources used by TCP/IP" on page 140, we further develop specific WLM considerations regarding sysplex security.

Detailed information about all the functions of the Workload Manager can be found in:

- *z/OS MVS Planning; Workload Management,* SA22-7602
- *System Programmer's Guide to: Workload Manager*, SG24-6472

4.1 Basic concepts of WLM

One of the strengths of the System z platform and the z/OS operating system is the ability to run multiple workloads at the same time within one z/OS operating system image or across multiple images that share the same physical processors. The function that makes it possible is dynamic workload management, which is implemented in the Workload Manager component of the z/OS operating system.

The z/OS Workload Manager (WLM) component introduces the capability of dynamically allocating or re-distributing server resources to competing workloads based on user-defined goals and the resources they require within a z/OS image.

The z/OS Workload Manager is able to perform this function also across multiple images of z/OS, Linux, or VM operating systems hosted in the same physical System z.

Another function of WLM is to assist routing components and products to dynamically direct work requests associated with a multisystem workload to run on a z/OS image within a Parallel Sysplex that has sufficient resources to meet the customer defined goals.

WLM constantly monitors the systems and has an overall view of the performance of the workload running in the parallel sysplex. Using this information, WLM advises and assists the subsystem with making decisions about which system the work will be routed to.

Workload management is a combined cooperation of various subsystems, such as CICS, IMS, JES, TSO, DB2, and so on, that interface with WLM to automatically balance tasks across all the resources of the Parallel Sysplex. The work can come from batch, SNA, TCP/IP, DRDA®, or WebSphere® MQ. When a failure occurs, the workload will be automatically redistributed to the remaining available resources.

For batch workload, WLM can start additional initiators if they are needed to obtain the performance goals. WLM balances the WLM-managed initiators between the members of a sysplex. On highly utilized machines, the number of initiators is reduced, while more are started on low utilization systems. WLM attempts to distribute the initiators across all members of the sysplex to balance the utilization of the systems while taking care that the jobs with affinities to specific systems are not hurt by its decision. Initiators are stopped on systems that are utilized over 95% when another system in the sysplex offers the required capacity for starting new initiators.

With WLM, you can manage the performance and number of servant regions in WebSphere Application Server for z/OS. WLM manages the response time and throughput of WebSphere transactions according to their assigned service class, the associated performance objectives, and the availability of system resources. To meet these goals, WLM sometimes needs to control or override the number of servant regions that are active. A minimum or maximum value or number of servant regions can be specified in the WebSphere Application Server to control the number of started regions.

Workload Manager information is used by the Sysplex Distributor to distribute incoming work according to varying LPAR capacities, sending work where there is more capacity. A parameter on the VIPADISTRIBUTE statement may be used to assign incoming connections for the Distributed Dynamic VIPA (DDVIPA) among available server instances in a round-robin method.

WLM can also be used to limit the amount of processing capacity available to some work, and to set a minimum processing capacity for some work in the event that the work is not achieving its goals. For most systems, you can let workload management decide how to manage the resources in the sysplex and not using these limits. However, there can be some special case where you want to enforce these limits. One of these cases would be precisely to use this capability as a protection not to reach a denial of service condition.

WLM is a policy-driven task, and there are often different policies in effect at various times of the day. For example, there may be a daytime policy where online tasks have priority, and a nighttime policy where batch jobs are more important. There is only one WLM policy active at a time in the sysplex.

The collection of the different polices, with the definition of the kinds of workloads and the corresponding goals, are contained in the service definition. It describes the performance goals for the workloads with a sysplex scope.

The service definition concept is presented graphically in Figure 4-1. All work running in the installation is divided into *workloads*. A workload is a group of work that is meaningful for an installation to monitor, like CICS, DB2, Batch, and so on. Within a workload, you group work with similar performance characteristics into *service classes*. You create service classes for a group of work with similar:

- Performance goals (response time, execution velocity, or discretionary)
- Resource requirements
- Business importance to the installation

To use the set of definition specified in a policy, that policy has to be activated.

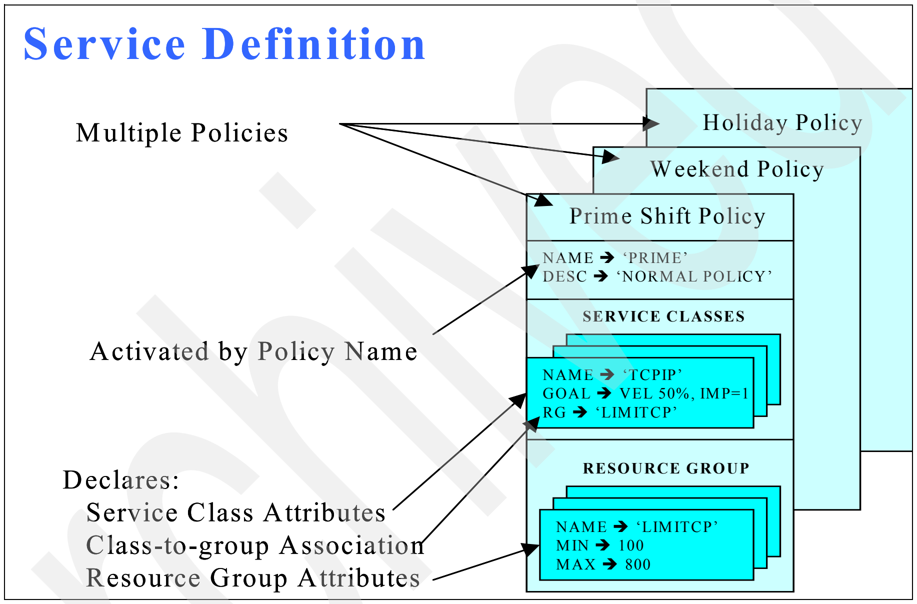


Figure 4-1 Concept of WLM service definition

The explicit definition of workloads and performance goals is called a *service definition*. The service definition contains all the information about the installation needed for workload management processing. There is one service definition for the entire sysplex.

A service definition contains:

- One or more *service policies*, which are named sets of overrides to the goals in the service definition. When a policy is activated, the overrides are merged with the service definition. You can have different policies to specify goals intended for different times. Service policies are activated by an operator command, or through the ISPF administrative application utility function.

- *Workloads*, which aggregate a set of service classes together for reporting purposes.
- *Service classes*, which are subdivided into periods, group work with similar performance goals, business importance, and resource requirements for management and reporting purposes. You assign performance goals to the periods within a service class.
- *Report classes*, which group work for reporting purposes. They are commonly used to provide more granular reporting for subsets of work within a single service class.
- *Resource groups*, which define processor capacity boundaries across the sysplex. You can assign a minimum and maximum amount of CPU service units per second to work by assigning a service class to a resource group.
- *Classification rules*, which determine how to assign incoming work to a service class and report class.

4.2 Limiting resources consumption with WLM

Resource groups are a way of limiting or guaranteeing resource capacity.

Work can be grouped into a service class based on resource requirements. If you have a group of batch work that can consume vast amounts of resources, and you want to limit it, you can define a service class and assign it to a resource group with a maximum amount of capacity specified. If the work exceeds that capacity, workload management slows the execution rate of the work classified in the associated service class. Also, if a certain group of work needs a minimum amount of processor capacity, you can set up a service class and assign it to a resource group, where a minimum value is specified.

You can define up to 32 resource groups per service definition.

You cannot assign a resource group to service classes representing transaction-oriented work, such as CICS or IMS transactions. The ISPF application notifies you with an error message if you attempt to define one. If you want to assign a minimum or a maximum capacity to CICS or IMS work, you can do so by assigning a resource group to their regions.

In defining additional service class, you have to keep in mind that the intrinsic performance of WLM is affected by the number of service classes to manage. You have to find a balance, and to group workload as much as possible in the service classes.

For most systems, you can let WLM decide how to manage the resources in the sysplex and not use resource groups. You set performance goals for work and let workload management adjust to meet the goals.

4

HMC and Sysplex Timer

In this chapter we address miscellaneous Security-related items for the Systems Hardware Management Console (HMC) and the Sysplex Timer®, such as:

- Physical security
- Access control
- Network security

4.1 HMC

A Hardware Management Console (HMC) is connected via a network to the Support Elements (SE) of the Central Processor Complexes (CPC) that it manages. Tasks selected at the HMC are forwarded to the corresponding SEs for execution.

Typically there are multiple HMCs and SEs connected through a LAN, and one single HMC can access one or more CPCs.

The HMC and the Support Element delivered with the z9™ systems have a new implementation: they operate as a closed platform, which means that other applications or user programs may not be added to their imbedded software. This enhances their security and integrity. As this also prohibits the installing the HMC ESCON® Director Control programs and Sysplex Timer controls, separate PCs are now mandatory for controlling ESCON Directors and sysplex timers.

The new HMCs delivered with System z9™ (FC 0079) are connected to the SEs in the CPCs only using TCP/IP. There is no longer any SNA and NetBios support in this HMC model. As a consequence, it does not support the 9472 systems G1, G2, G3, or G4.

The following systems are supported, but they must be upgraded to a new AROM level:

- z990 and z890 - with Driver 55
- z900 and z800 - with Driver 3G
- 9472 G5 and G6 - with Driver 26
- M3000 - with Driver 24

An overview of the System z9-109 HMC is given in the IBM Redbook *IBM System z9 Enterprise Class Technical Guide*, SG24-7124.

HMC access control to SEs

The *Domain Security* task on the HMC provides a method to maintain security of a processor complex by controlling the access of the HMCs to the SEs. The HMC can only communicate with an SE that have the same domain name and domain password as the Hardware Management Console. Assigning a unique domain name and domain password to a Hardware Management Console and the CPC's SE that are defined to it will isolate those CPCs from any other Hardware Management Console connected to the same Local Area Network.

4.1.1 HMC Physical Access Security

Control of physical access to the HMC is a customer responsibility. Basically the requirement is to keep the HMC in a locked room. In addition, one can also use the physical lock on the PC and protect it with a power-on password.

4.1.2 Logical Access Security

User authority for the HMC is determined by user roles that are associated with tasks. Each user role is actually a collection of authorizations. Membership to a user role is protected by a password.

Following the installation of the CPCs, HMCs, and SEs in your configuration, the access administrator should change the default logon passwords at the HMCs and SEs.

You can use the following enable/disable controls to help you control access and provide focal point capabilities:

- Licensed Internal Code (LIC) update (change management focal point)
- Remote service support
- Remote customer access
- Remote service access
- Auto-answer of the modem

We recommend that you log off each Hardware Management Console when it is not in use. The HMC provides a status bar capable of displaying status colors (or grey patterns) to alert you when operator activity is needed, even when no one is logged on.

Additional security consideration for the pre-z9 HMCs include:

- Use the OS/2® lockup function to lock the HMC keyboard after a predetermined period of keyboard inactivity has expired.
- Customize the HMC’s secure desktop setting to control wether users have access to the applications and objects on the console’s OS/2 desktop.

4.1.3 Network Access Security

To enhance its security, the HMC delivered with the System z9 is configured with a built-in firewall to limit network inbound and outbound accesses. By default, no connections are allowed through the firewall. As objects are defined to the HMC application, the necessary firewall rules are added to allow for communications to and from these objects. Firewall rules are also added to allow external user connections, access by product engineering, and the customization of network settings to allow specific applications.

You should consider these security recommendations:

- Create a private LAN to interconnect the HMCs with the controlled SEs. Using a private LAN for your configuration offers several security, availability, and performance advantages as follows:
  - Direct access to the LAN is limited to the Hardware Management Consoles, SEs, CPCs, and control units attached to it. Outsiders cannot connect to it.
  - Traffic disruption due to temporary outages on the LAN is reduced, including disruptions caused by plugging in and powering on new devices on the LAN (minor) to LAN adapters being run at the wrong speed (catastrophic).
  - LAN traffic is minimized, reducing the possibility of delays at the Hardware Management Console/SE user interface.

  Additionally, HMC and SE activity, if occurring on a non-private LAN, could potentially disrupt other LAN activity.

  If using a private LAN is not practical, isolate the LAN used by the HMCs, SEs, and CPCs by providing a LAN bridge or router between the isolated LAN and the backbone LAN to provide an intermediate level of isolation.

  For the pre-z9 HMC, if the LAN bridge exists between a HMC and the SEs, the LAN bridge must allow SNA, TCP/IP, and NetBios traffic to flow across the bridge for automatic discovery of SEs by the HMC.
- Assign a unique domain name that includes all the CPCs controlled from one or more Hardware Management Consoles.
- If a remote console is used for remote operations access, assign a secure logon password.

Remote access

Each HMC contains a Web server that can be configured to allow remote access for a specified set of users.

- HMCs ordered with System z9

  A Web browser can connect to the local HMC using either a LAN TCP/IP connection or a switched, dial, network PPP TCP/IP connection. Both types of connections can only use encrypted (HTTPS) protocols, as configured in the local HMC. If a PPP connection is used, the PPP password must be configured in the local HMC and in the remote browser system. Logon security for a Web browser is provided by the local HMC user logon procedures. Certificates for secure communications are provided, and can be changed by the user.

- pre-z9 HMC
  - The HMC may be connected using a Token Ring or Ethernet network, using both the TCP/IP and SNA protocols.

    Access control to the remote HMC is provided by the HMC user logon requirement, the secure transmissions between the HMC and SEs, and domain security controls.
  - A Web browser can be used.

    Security for a browser connection is provided by the HMC enablement functions, HMC user logon controls, and customer network access controls. Encryption of the browser data may be added by selection **Enable Secure only**.

Remote Support Facility (RSF)

The Hardware Management Console provides Remote Support Facility to aid in the service and maintenance of your system. As general rules:

- Remote support connections for service are always initiated by the customer machine (HMC) to IBM Retain. IBM never initiates a call into a customer system.
- An RSF connection can be made via a phone connection (via the HMC integrated modem) or an Internet connection (via the enterprise LAN).
  - If it is via a phone connection, the protocol used is PPP and TCP/IP. All data transfer is done using SSL encryption.

    Phone connections are made using AT&T as the global service provider. Provision is made for the use of local phone connections where available.
  - If it is via an Internet connection, the protocol used is TCP/IP and all data transfer is done using SSL encryption.
  - Internet connections are assumed to go through a customer firewall system before entering the global Internet.
- On either style connection, the IBM Retain system validates that the incoming requesting system is known and authorized, or the connection is dropped.
- All data transferred to IBM is for service use only. No customer data is transferred.
- All data transferred from IBM to the customer system is encrypted and checked prior to use.
- All the data exchange protocols are proprietary to IBM service and not published.

- The HMC, by default, will not answer an incoming call via the modem. This can be changed by the customer if desired to allow remote backup operations.

4.2 Sysplex Timer

In this section we discuss the Sysplex Timer.

4.2.1 Sysplex Timer (9037-002)

The Sysplex Timer is the unit used for the time synchronization of the different CPC and CFs connected to it. Note that as of the writing of this book the Sysplex Timer is intended to be replaced by a new time synchronization feature called STP (for Server Time protocol). STP is addressed in more details in 4.2.2, “Server Time Protocol” on page 119.

The Sysplex Timer is required in a parallel sysplex environment that involves multiple physical CPCs or when there is a physically separated system that hosts the Coupling Facility function. There are Sysplex Timer links between each physical CPC and the Sysplex Timer device, known as the External Time Reference (ETR) links. There are also links between redundant Sysplex Timer devices, referred to as the Control Link Oscillator (CLO) links.

The Sysplex Timer can be controlled from a dedicated console, or from an HMC running the application for this control. The HMC delivered for System z9 support does not allow you to install this application. The Sysplex Timer console connects to the Sysplex Timer through a Token Ring LAN using the TCP/IP protocol.

We recommend for security reasons that you use a dedicated PC to control the Sysplex Timer and to put it, along with the Sysplex Timer device itself, in a locked room.

The Sysplex Timer console functions are protected by a password. There are three levels of password protected functions:

- Administrative
- Maintenance
- Operator

You may elect to have the Sysplex Timer itself synchronizing to an external reference source, such as the NIST Automated Computer Time Service (ACTS). The sysplex timer unit has to be connected to a modem. To dial out the sysplex Timer must have been set up with a number to dial. This setup is protected by the administrative functions password.

4.2.2 Server Time Protocol

IBM plans to make available a new synchronization feature, STP, which is designed to provide the capability for multiple System z9 and zSeries servers to maintain time synchronization with each other. STP is planned to be the follow-on to the Sysplex Timer (9037-002). The Sysplex Timer and the STP are designed to allow events occurring in different System z9, zSeries, and S/390 servers to be properly sequenced in time.

STP is designed for severs that have been configured to be in a Parallel Sysplex or a sysplex (without a Coupling Facility), as well as servers that are not in a sysplex but that need to be time synchronized. STP is designed to allow timing information to be sent between servers and CFs over the InterSystem Channel-3 (ISC-3) links configured in peer mode Integrated Cluster Bus-3 (ICB-3) links or Integrated Cluster Bus-4 (ICB-4) links.

STP is designed to:

- Allow clock synchronization for the z9-109, z900, and z890 servers and Coupling Facilities without requiring the Sysplex Timer boxes.
- Support a multisite timing network of up a 100 km (62 miles) over fiber optic cabling, allowing a Parallel Sysplex to span these distances.
- Potentially reduce the cross-site connectivity required for a multisite Parallel Sysplex.
- Coexist with an ETR network.

STP is planned to be available as a feature on the z9-109, z990, and z890 and to be supported by z/OS V1.7 (PTFs will be required to enable STP support).

Hardware Cryptography in a Sysplex

This chapter describes the new feature of the HCR7730 level of ICSF which provides CKDS support in a Sysplex environment.

Note: ICSF can now be downloaded, in an SMP/E installable format, from the Web site:

`http://www.ibm.com/servers/eserver/zseries/zos/downloads/`

Along with the installable code comes a program directory that contains miscellaneous information including the prerequisite z/OS level.

As of the writing of this book, the latest available level of ICSF is the FMID HCR7730, which is required to be at least at z/OS V1R6.

In this chapter we address the following items:

- Configuration of Sysplex updating facilities
- Description of the serialization mechanisms
- Explanation of data space updating mechanisms
- Running multiple CKDS datasets in a sysplex
- New messages
- New abend and reason codes

4.1 Background

ICSF had not been providing full sysplex support, as the CKDS data set can be shared between different ICSF instances but the in-storage copies of the data set are not automatically synchronized when one ICSF instance was adding, modifying, or deleting a key in the shared dataset. Synchronization has then to rely on manual processes.

New functions have been added into ICSF HCR7730 to maintain the coherency of the multiple in-storage copies of the CKDS in a sysplex configuration. Before we proceed and describe these new functions we explain in more detail how CKDS sharing used to be achieved before the availability of ICSF HCR7730.

4.1.1 Sysplex support prior to ICSF HCR7730

There are two key datasets used by ICSF. These are the Cryptographic Key Data Set (CKDS) and Public Key Data Set (PKDS). The CKDS is used to hold DES kinds of keys used in symmetric encryption and MAC algorithms, while the PKDS is used to hold asymmetric keys (that is, RSA keys with the Crypto Express 2 coprocessor).

The CKDS dataset is the dataset in which content is updated and accessed more frequently, due to the relatively ephemeral nature of the symmetric keys. Asymmetric keys are mostly long-living keys, and a decision was made by the ICSF laboratory to focus on developing full sysplex support for the CKDS only in HCR7730.

The CKDS can be defined using the following IDCAMS statements:

```
DEFINE CLUSTER (NAME(LENNIE.CSFCKDS)      -
                RECORDS(100 50)           -
                RECORDSIZE(252,252)       -
                KEYS(72 0)                -
                FREESPACE(10,10)          -
                SHAREOPTIONS(2)) -
          DATA (NAME(LENNIE.CSFCKDS.DATA) -
                BUFFERSPACE(100000)       -
                ERASE                     -
                WRITECHECK)               -
         INDEX (NAME(LENNIE.CSFCKDS.INDEX))
```

The SHAREOPTIONS above are those used for datasets that are shared and managed by the application. ICSF makes use of system scope ENQ macros to serialize access to the CKDS so that read integrity is maintained when the CKDS is updated by the ICSF address space as long as only one system is accessing the dataset for updates. However, read integrity is not maintained if KGUP updates the CKDS.

The entire CKDS is read into a dataspace named CSFDS001 when ICSF starts. This in-storage copy of the CKDS is managed by ICSF as follows:

- Updates that are made to keys using services such as CSNBKRC (Key Record Create) and CSNBKRW (Key Record Write), CSNBKRD (Key Record Delete) and CSNBKPI (Key Part Import), or any service that invokes these implicitly are made to the CKDS dataspace and are also written to the CKDS.
- However, READ activity using the CSNBKRR (Key Record read) or any service that invokes it implicitly is always directed only to the dataspace.

The KGUP utility can be used to add, modify, and delete keys in the CKDS, but this is done *without* updating the CKDS in-storage copy. That is the data space and the CKDS can become de-synchronized as the DASD version of the CKDS is being updated by the local KGUP.

This is also true for updates done to the shared CKDS by remote instances of ICSF or KGUP, as none of them will be reflected in the local in-storage copy of the CKDS.

The solution to put the CKDS dataset and its in-storage copy back in synchronization again is then to use the CSFEUTIL program or the REFRESH facility from the ICSF ISPF panels. Either one of these methods will refresh the local in-storage copy of the CKDS from the CKDS dataset.

The mode of operation when not in full sysplex support (that is, with ICSF levels previous to HCR7730) is illustrated in Figure 4-1 and Figure 4-2.

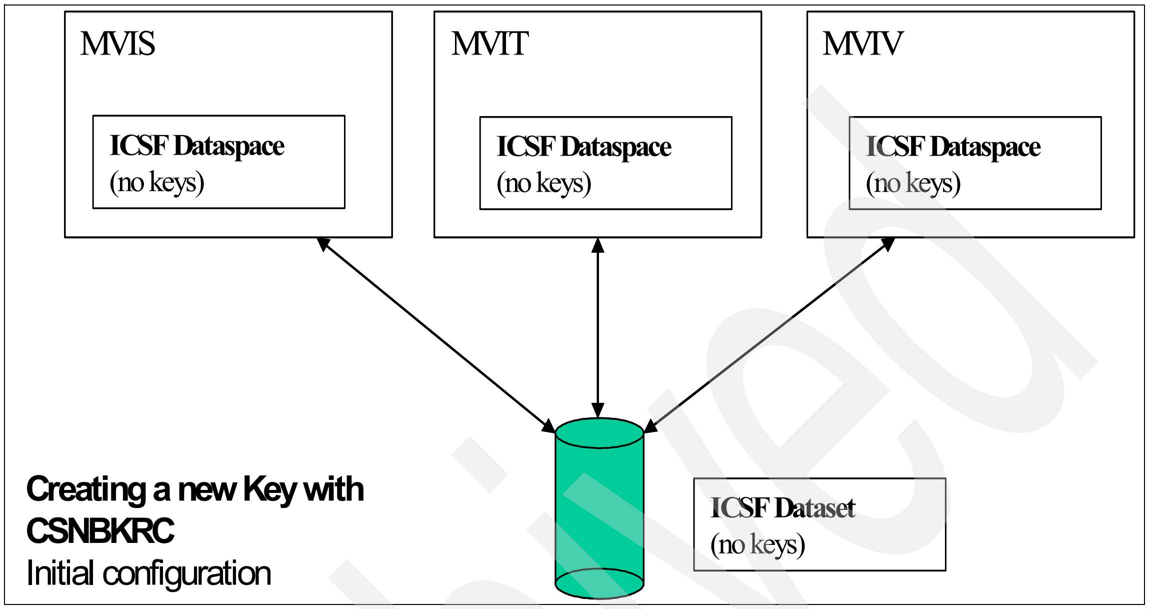


Figure 4-1 Pre-HCR7730 Sysplex configuration - Key Record Create part 1 of 2

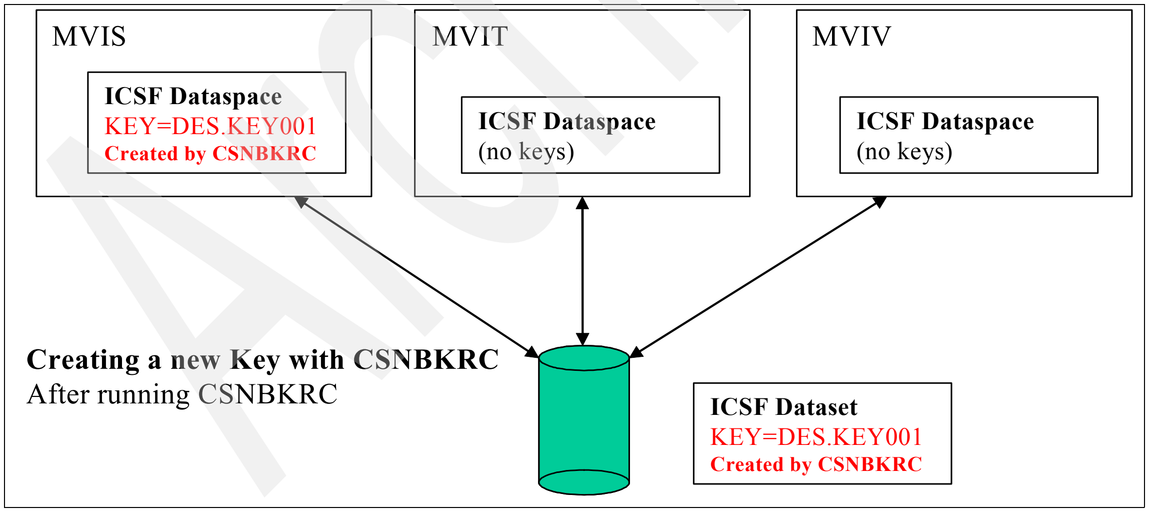


Figure 4-2 Pre-HCR7730 Sysplex configuration - Key Record Create part 2 of 2

It is the lack of data space synchronization and dataset cross-system serialization that is addressed in the HCR7730 version of ICSF.

4.2 Sysplex support with ICSF HCR7730

ICSF HCR7730 provides a mechanism to insure CKDS dataset and in-storage copy synchronization in a sysplex environment.

The options dataset SYSPLEXCKDS keyword

The SYSPLEXCKDS keyword is new with ICSF level HCR7730. Its syntax is:

```
SYSPLEXCKDS(YES|NO,FAIL(YES|NO))
```

Note that the FAIL parameter is only applicable when the first parameter is YES.

SYSPLEXCKDS(NO,FAIL(NO)) is the default setting. The first NO indicates that no sysplex sharing protocol is to be used. This is equivalent to the serialization mechanism used prior to ICSF HCR7730.

SYSPLEXCKDS(NO,FAIL(YES)) is the same as SYSPLEXCKDS(NO,FAIL(NO)).

SYSPLEXCKDS(YES,FAIL(NO)) and SYSPLEXCKDS(NO,FAIL(YES)) will cause the new CKDS sharing protocol to be used when accessing the CKDS within a sysplex.

The two options FAIL(YES) and FAIL(NO) indicate what action should be taken when an attempt to join the sysplex group by ICSF fails. If FAIL(YES) is specified, the ICSF initialization will fail if the attempt to join the sysplex group fails. If FAIL(NO) is specified, then ICSF initialization will continue but the sysplex sharing protocol will not be used. Updates to the CKDS will not be received from other systems and updates performed will not be notified to other systems, just as though SYSPLEXCKDS(NO,FAIL(YES|NO)) had been coded.

Once we have established a CKDS shared across a sysplex, we wanted it to continue to be maintained correctly, so we generally specified SYSPLEXCKDS(YES,FAIL(YES)) in our testing.

4.2.1 How the new CKDS Sysplex sharing works

It is mandatory that all systems sharing the CKDS are using HCR7730. Systems using lower levels of ICSF will not be able to take part in the CKDS sharing process.

Sysplex XCF messages are sent to the sysplex members in the ICSF group. At any given time the messages may originate from any one system, reflecting an update done to the CKDS dataset from that system. The systems receiving messages take appropriate actions to update their CKDS in-storage copy. The XCF group name is fixed to SYSICSF by design.

The messages are sent when one system updates or creates a CKDS record using one of the following ICSF services,

- Key Record Create (CSNBKRC)
- Key Record Delete (CSNBKRD)
- Key Record Write (CSNBKRW)
- Key Part Import (CSNBKPI)

If the CKDS is updated as a result of using the KGUP utility or by performing a CKDS re-encipher, then the updates will *not* be propagated to the other systems, except if the CKDS update is the completion step of a Load Operational Key process started at the TKE. In this latter case the CKDS update driven either by KGUP or from the ICSF ISPF panel is propagated to the in-storage copy of all the ICSF instances sharing the CKDS.

Access to the entire CKDS dataset is serialized using ENQ macros with a Qname of SYSZCKT and an Rname of ckdsname. This ENQ is used to ensure serialization between the ICSF address spaces and block writers of the CKDS that write using CSNBKRC, CSNBKRD, or CSNBKRW. But KGUP is not prevented to write to the CKDS.

In addition to the above level of serialization, all ICSF address space I/O to the CKDS dataset and updates of in-storage copies of the CKDS are serialized using an ENQ with a Qname of SYSZCKDS and an Rname of .ckdsname. However, this ENQ is subject to a time-out to allow for situations where one system is holding the ENQ for an excessive time. Note also that XCF messages are broadcast requesting other ICSF address spaces to release the ENQ. The time-out is designed to handle software or hardware failures that may cause WAITs.

Each of the ENQs above is issued with a scope SYSTEMS and so will be propagated to the whole sysplex (and potentially beyond if the GRSplex includes systems outside the sysplex).

Internally to the ICSF address space there are 512 latches, which are used to serialize access to particular key records within the CKDS. This ensures that the same key record cannot be updated simultaneously by requests from two independent processes making requests of ICSF. A hashing method is used to select a particular latch to correspond with a given key record. A latch is obtained whenever a key record is being read or written.

Creating a record

When a new record is created, via CSNBKRC, the local in-storage copy of the CKDS is updated along with the CKDS dataset itself. In the same time frame a signal is broadcast via the XCF links to instruct the other systems that a new record is available to be read into the CKDS in-storage copy. The other systems will then read the record from the CKDS dataset into their local in-storage copy.

Updating a record

When a CKDS record is updated via CSNBKRW or CSNBKPI, the local in-storage copy of the CKDS is updated along with the CKDS dataset itself with the new record contents. In the same time frame a signal is sent via the sysplex XCF links to instruct the other systems to invalidate the record in local CKDS in-storage copy and to read the updated version from the CKDS dataset.

Deleting a record

When a CKDS record is deleted via CSNBKRD, the local CKDS in-storage copy along with the CKDS dataset itself is updated. In the same time frame a signal is broadcast via the XCF links to the other systems to invalidate the record in their local in-storage copy. The other ICSF instances then delete that record from their in-storage copy.

The operation of ICSF in this new mode is shown in Figure 4-3 and Figure 4-4 on page 128.

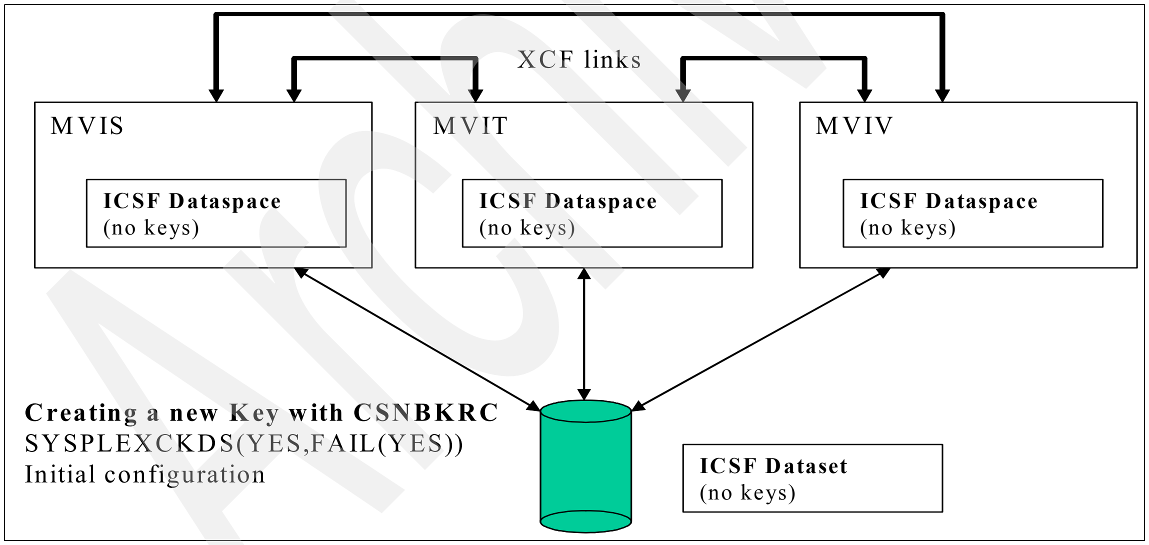


Figure 4-3 HCR7730 Sysplex configuration - Key Record Create part 1 of 2

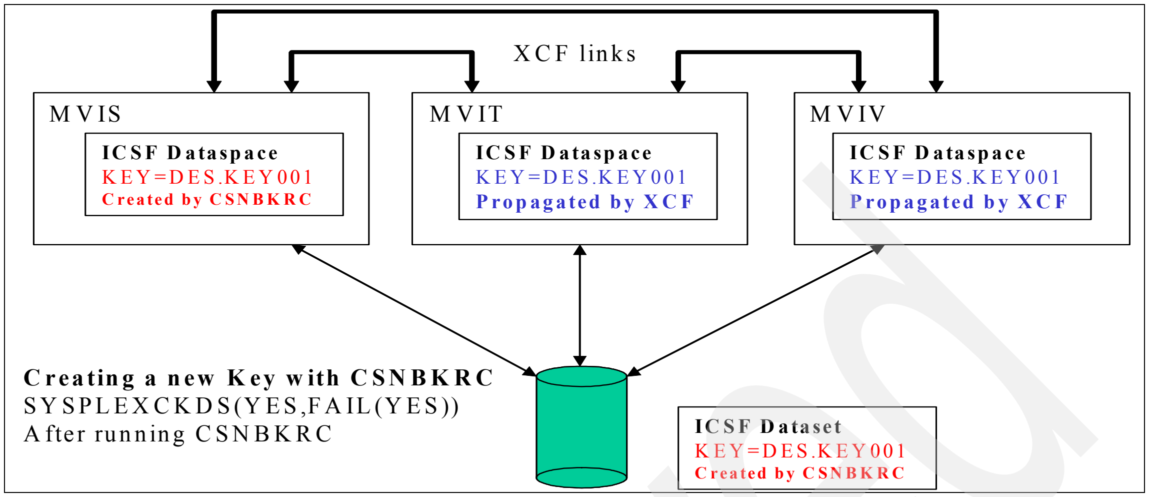


Figure 4-4 HCR7730 Sysplex configuration - Key Record Create part 2 of 2

Part 2

One Sysplex member with network connectivity

Part 1 has been focusing on protecting the sysplex-specific resources without considering the potential security exposures induced by a network connection.

In this part we address the security exposures that a sysplex configuration with only one member connected to a non-secure network may have to deal with.

This is different from what is addressed in Part 3, “All Sysplex members with network connectivity” on page 175, in that here we assume that the network communications are not propagated from the connected member to other members in the sysplex.

Note, however, that the protection mechanisms and rules addressed in this part are fully re-usable in Part 3, “All Sysplex members with network connectivity” on page 175.

5

Protecting the network connection

This chapter describes the security implications when connecting one member of a sysplex to a non-secure TCP/IP network. It assumes that the security considerations discussed in Part 1, “Basic Parallel Sysplex security” on page 15, have already been taken carefully into account, as it would not be of any sense to try establish network level security if the basic platform security itself were questionable.

The reference books to use when designing and setting up the protections described in this section are:

- *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

5.1 General discussion on non-secure network threats

A lot has already been written on the threats one is exposed to when connecting to the Internet. All of these threats are carried by the same (and only possible) vector: the *IP datagram*. This is why it is so important to be able to apply the ISO 7498-2 principles to each datagram arriving from a non-secure network. The threats can roughly be categorized based on the levels at which they occur:

- At the TCP/IP stack level. The TCP/IP host itself can be harmed by the contents of the IP datagram, be it the header part or the carried data. This is possible because of a design mistake in the stack code, which does not handle certain configurations in the IP datagram correctly. These vulnerabilities are therefore vendor, and even code-level dependant, and the information is usually widely spread over the Internet. Hence, the very strong recommendation, for all platforms, to always apply the latest service updates to the TCP/IP stack code.
- The stack cannot handle the flow of communications, either because of the volume of packets or because of connection sequences violating the TCP/IP rules. These threats can be grouped roughly into two types: *intrusion* and *denial of service* attacks.
  - Generally speaking, intrusion threats consist of the capability to acquire undue privileges in the system in which the TCP/IP stack is running because of weaknesses in the stack code itself, usually compounded by weaknesses in the platform security products. Intrusion is generally the first step to causing more damage to user data or to the operation of the system itself.
  - In the Denial of Service (DoS) attack, the attacker manages to prevent the receiving TCP/IP stack from providing the expected service. A denial of service can be instigated in the system itself as the consequence of an intrusion, but it can also be initiated from the network. DoS attacks initiated from the network fall into these two categories:
    - The single packet DoS, where the IP datagram carries lethal contents to the receiving TCIP/IP stack. This is the typical consequence of a design or coding mistake in the TCP/IP stack code, which ends up crashing the stack.
    - The multiple packet DoS, where an attacker manages to flood the receiving TCP/IP stack with illegitimate communications, thereby preventing the overloaded stack from providing the expected services to legitimate clients. These attacks can be conducted by generating a very high volume of faked client requests or by exploiting open weaknesses in the TCP/IP protocol itself. There is not too much that can be done to protect against these attacks at the TCP/IP stack itself; minimally, the stack and the operating system should be designed not

to overly consume system resources during the overload period. The problem is rendered even more challenging in that it is often very difficult to detect, at first, whether the stack is under a DoS or is just facing high traffic from the clients. Installations usually take a holistic approach to preventing DoS attacks by installing software probes (*Intrusion Detection* probes) at several locations in their network, in which detected events are collected and correlated at a central point in order to provide fast and overall accurate alerts.

- At the application level, where the IP datagram went safely through the stack and the extracted data is a danger to the application. Here again we deal with design or coding mistakes, but at the application level this time. And again, what one can do to prevent this is to perform proper application code maintenance and, if possible, up-front filtering of the received IP packets on the basis of authentication of the sender and integrity checking of the received data.

Note: A clear-cut classification of all Internet threats would be impossible because there are so many ways of conducting attacks and exploiting TCP/IP weaknesses.

Two final points complete our discussion on Internet threats:

- Do not expect a TCP/IP stack to provide the required protection by itself, even for very strong stack codes. You will always need additional protection mechanisms such as firewalls or intrusion detection. Among the reasons for this are the possibility of administration errors that weakened your TCP/IP or system setup, and that denial of service attacks are quite difficult to discriminate from regular requests during their build-up phase.
- Again, whatever the security devices or products you put in place, the inherent security of the operating system is the first line of defense that all your security mechanisms will rely on.

5.2 TCP/IP oriented additional RACF protections

We have already mentioned the measures that you should have taken to protect your stand-alone sysplex environment. This takes care of protecting the fundamental MVS and UNIX resources of the system. The additional threats that connecting to a non-secure network brings require additional protections that RACF (or another external security manager that uses the SAF interface) can partially provide.

We recommend that you consider putting in place the following RACF protections.

5.2.1 TCP/IP resources as protected by RACF

These are resources that can be protected by profiles in the SERVAUTH class, the resource manager in charge being the z/OS TCP/IP stack. In this section we explain how to control TCP/IP application access to the TCP/IP stacks, ports, and networks.

Network access

The z/OS TCP/IP stack has an option to activate the SAF protection of a resource that identifies an IP network or IP address. In other words, RACF, or any other external security manager, can be used to control applications' access to and from a specific IP address. The resource itself is defined by a statement within the TCP/IP profile data set. This statement defines a resource name and maps it to a specific IP address or network.

The best way to illustrate this is by the example shown in Example 5-1.

Example 5-1 NETACCESS example

```
NETACCESS INBOUND OUTBOUND
    9.100.199.162     WORKSTN
    DEFAULT           OTHERNW
ENDNETACCESS
```

In this example, two resources have been defined via the NETACCESS statement in the TCPIP.PROFILE data set: WORKSTN represents IP address 9.100.199.162, while OTHERNW applies to all other IP addresses, also known as the *default zone*. In this example the controlled resources include both inbound and outbound traffic.

If the RACF SERVAUTH class is active, and the resource profile has been defined with UACC(NONE), any user ID attempting access to a network defined within the NETACCESS statement must have been permitted READ access to the resource. For example, in order to communicate with 9.100.199.162 on z/OS system MVSIV using the TCP/IP stack started as TCPIP2, the user ID would need permission to the following resource:

```
EZB.NETACCESS.MVSIV.TCPIP2.WORKSTN
```

The resource name has a predefined syntax with the following qualifiers:

- EZB.NETACCESS are fixed qualifiers.
- The system ID where the protection is to operate. MVSIV in this example.
- The TCP/IP stack started task where the protection is to operate. TCPIP2 in this example.
- The resource name that is defined in the NETACCESS statement in the PROFILE.TCPIP data set. WORKSTN in this example.

Notes: This protection enforces the access control for local applications to open an outbound TCP/IP connection or to receive from an inbound connection. It is not a direct protection against external threats as IP filtering could provide. Rather, it enforces local applications' authorization to use TCP/IP resources. Typically, if this protection is set up, a trojan horse kind of program, which we can expect not to have been authorized to the resource by the RACF administrator, would be prevented from communicating with network entities.

Port access

Similar to network access, port access implements access control for TCP/IP ports using a resource name of the following form:

```
EZB.PORTACCESS.MVSIV.TCPIP2.SERVER1
```

Again, the first two qualifiers are predefined.

Corresponding to this statement, the PORT reservation statements in the PROFILE.TCPIP data set would have to specify the resource name such as that shown in Example 5-2.

Example 5-2 Port statement using SAF

```
PORT
    21 TCP * SAF SERVER1
    23 TCP * SAF SERVER1
```

In order for any task to use TCP ports 21 or 23, permission to the above resource profile is required. The TCP/IP port SAF resource protection applies both to UDP and TCP communications.

Tip: You may want also to consider the following options that do not use SAF protection of resources but are fully performed by the TCP/IP stack:

- For ports numbered 1024 and lower, their use can be controlled directly via the TCPCONFIG RESTRICTLOWPORTS and UDPCONFIG RESTRICTLOWPORTS statements within the PROFILE.TCPIP data set. In addition, the PORTRANGE and PORT statements can also be used to restrict application access to ports.
- The usage of the BIND option in the port statement should also be considered as a security option as the server will be listening only to a specific IP address and not to all IP addresses the stack is bound to.
- For the ports that are not intended to be used, the RESERVED keyword can be used to block any access to them by applications.

Stack access

Continuing the pattern already established, the following syntax can be used when identifying a TCP/IP stack as a SAF resource (remember that it can be up to eight TCP/IP stacks executing concurrently in a single z/OS image):

```
EZB.STACKACCESS.MVSIV.TCPIP2
```

With this resource defined to the SAF product with a UACC of NONE, all users that need access to any IP resource would require being explicitly permitted.

Tip: You can achieve a simple network segregation setup in your z/OS image by starting several TCP/IP stacks (for a maximum of eight stacks concurrently running in a single z/OS instance), each stack being connected to a different network.

TCP/IP command control

There are several commands that display or control the TCP/IP resources' status that should be restricted to a limited set of users. The following commands are the most significant TCP/IP-related commands that should be reviewed in a security enforcement context:

VARY	The resource MVS.VARY.TCPIP.<command_name> can be defined in the OPERCMDS class and can be used to restrict access to the quite powerful VARY command.
NETSTAT	Access to the NETSTAT command is protected with the profile

EZB.NETSTAT.<system_ID>.<tcpip_procname>.<netstat_option> in the SERVAUTH class.

Note that the NESTAT DRop or -D command is controlled by MVS.VARY.TCPIP.DROP in the OPERCMDS class.

> **Note:** Information about local adapter addresses, routing information, and active connection is displayed using the MVS or UNIX System Services `netstat` command. Such information is unlikely to be needed by general users, and can be exploited for malicious purposes. We therefore strongly recommend that you protect these `netstat` options with the appropriate profiles in the SERVAUTH class.

PING and TRACEROUTE	The TSO PING and TACERTE commands must be listed as an AUTHCMD in IKJTSOxx. The UNIX System Services `ping` and `traceroute` commands require superuser authority. These commands could be categorized as a medium exposure, as they disclose information about hosts and routers in the network and access must be granted accordingly. Both commands use the Internet Control Message Protocol (ICMP). If you plan on also exploiting the IP filtering function of the z/OS TCP/IP stack, see also 6.2, "IP security filtering" on page 149, for information about filtering ICMP packets.
ipsec	The UNIX System Services `ipsec` command controls the use of the integrated IP filtering and IPSec VPN functions in the TCP/IP stack. Usage of the `ipsec` command is controlled using the SERVAUTH facility profile EZB.IPSECCMD.<system_ID>.<tcpip_procname>.<command_type>. Usage of this command and its options should be carefully controlled.
pasearch	Access to the `pasearch` command is granted to superusers. For non-superusers its access is controlled by the SERVAUTH class using resource EZB.PAGENT.<system_ID>.<tcpip_procname>.<policy_type>. The `pasearch` command cannot be used to change policy information, but we recommend that displaying policy information be restricted.

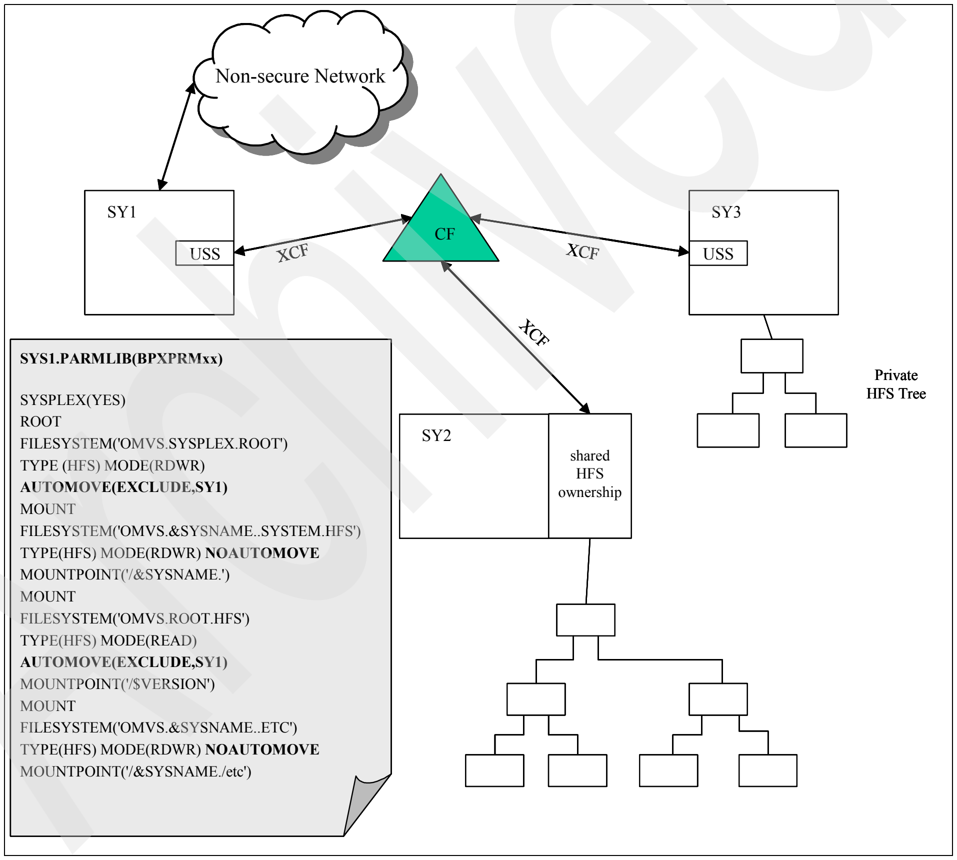
5.3 UNIX System Services security considerations

In this section we discuss UNIX System Services security considerations.

5.3.1 AUTOMOVE of the shared HFS owner

As we have stated previously, the AUTOMOVE option of the MOUNT command is used to give directive on what sysplex members should do to the file system if the system that owns it fails.

Figure 5-1 describes a Sysplex configuration with one member now connected to a non-secure network.



```
SYS1.PARMLIB(BPXPRMxx)

SYSPLEX(YES)
ROOT
FILESYSTEM('OMVS.SYSPLEX.ROOT')
TYPE (HFS) MODE(RDWR)
AUTOMOVE(EXCLUDE,SY1)
MOUNT
FILESYSTEM('OMVS.&SYSNAME..SYSTEM.HFS')
TYPE(HFS) MODE(RDWR) NOAUTOMOVE
MOUNTPOINT('/&SYSNAME.')
MOUNT
FILESYSTEM('OMVS.ROOT.HFS')
TYPE(HFS) MODE(READ)
AUTOMOVE(EXCLUDE,SY1)
MOUNTPOINT('/$VERSION')
MOUNT
FILESYSTEM('OMVS.&SYSNAME..ETC')
TYPE(HFS) MODE(RDWR) NOAUTOMOVE
MOUNTPOINT('/&SYSNAME./etc')
```

Figure 5-1 Changed AUTOMOVE settings when connected to non-secure network

The concern we raise in this scenario is that if member SY2 fails and member SY1 became the owner of the shared HFS through an AUTOMOVE action, then you have created a potential risk, as the shared HFS owner is now the one probably the most exposed to various threats.

In order to avoid SY1 from being a target for this automated move, should the owner of the shared HFS fail, the SYS1.PARMLIB(BPXPRMxx) member that we show specifies moving the root system of your shared HFS to the alternate system SY3 if SY2 fails by excluding SY1 as a possible AUTOMOVE target.

You may want to use these INCLUDE or EXCLUDE system lists when you feel that there is a high enough exposure giving the shared HFS ownership to the network connected system. This is, however, tied to the availability of at least two non-exposed systems as potential shared HFS owners.

5.4 Considerations on WLM participation to Security

In Chapter 4, “Sysplex Workload Management and Security” on page 107, we explained the main concepts of WLM including the use of resource groups. This can be applied to the TCP/IP address space by establishing limits to its use of system resources, the purpose of this being to mitigate the effect of a multi-packet denial of service attack aimed at overloading the networking resources of the z/OS image. By capping the resources granted to the TCP/IP address space, other workloads still benefit from system resources preserved for their own use.

The reference document to use here is *z/OS MVS Planning: Workload Management*, SA22-7602.

Important: What we describe here is a possible approach to protect against TCP/IP denial of service attacks that we feel have to be addressed in this book. However, we recommend as a z/OS protection against TCP/IP denial of service attack the use of the traffic regulation feature imbedded in the z/OS Intrusion Detection Services (IDS). We describe IDS in 6.7.4, “The z/OS Intrusion Detection Services” on page 170.

Attention: The possible use of WLM that we explain here might not fit your current installation requirements or may induce undesirable secondary effects. We strongly recommend that any setup decision be made with proper support of a WLM expert.

5.4.1 WLM and the TCP/IP address space

The TCP/IP address space is classified under subsystem type STC (started task). If you do not classify TCP/IP, it is, by default, associated with the SYSSTC service class, which is a system-supplied service class. Address spaces in the SYSSTC service class are kept at a very high dispatching priority.

You can limit the amount of processing capacity at the service class level only. Also, if you limit the amount of processing capacity for a service class containing other work than the one you target, the performance of the other workload can be influenced. Therefore you need to define a service class for the specific workload you want to tightly control.

5.4.2 How WLM can limit the resources used by TCP/IP

You can assign a minimum and maximum amount of processing capacity available to work by assigning a resource group to a service class. A single resource group can be assigned to one or more service classes.

If work in a resource group is consuming resources above the specified maximum capacity, the system throttles the associated work to slow down the rate of resource consumption. The system may use several mechanisms to slow down the rate of resource consumption, including swapping the address spaces, changing its dispatching priority, and capping the amount of service that can be consumed.

Attention: A resource group tracks resource consumption in a given service class at the sysplex level, meaning that WLM decisions have a sysplex-wide effect regarding the workloads that are running in this service class.

Proper differentiation in service class names has to be made at the members level if it is desired that the resource group algorithm applies to only a subset of the sysplex members.

Note: A resource group cannot be associated with the SYSSTC service class. Make sure that the TCP/IP address space goes into a user-defined service class.

In order to assign a resource group to a service class especially defined for TCP/IP, perform the following steps:

1. Optionally, define a TCP/IP workload, if not done yet, to WLM.

2. Calculate the maximum value for the usage of the CP resources as explained in 5.4.3, “Calculation of the usage limit values” on page 142
3. Define a resource group and specify the calculated maximum value.
4. Define a service class:
   a. Specify the workload name.
   b. Specify the resource group.
   c. Set CPU Critical to YES.
   d. Insert the performance goals:
      i. Specify the execution velocity goal for the TCP/IP workload. A typical value would be about 50–60%. Execution velocity goals define how fast work should run when ready, without being delayed for processor, storage, I/O access, and queue delay. Execution velocity goals are intended for started tasks or long-running batch work.
      ii. Specify an importance of 1.
5. Modify the classification rules for assigning the TCP/IP workload to the defined service class. TCP/IP belongs to the STCs, and use TN (transaction name) as the work qualifier.

You can integrate these definitions in your current WLM environment by a dynamic modification of the current policy. The definitions will become active with an activation of the modified policy.

Before deciding on the maximum limit value in the resource group, you must have a good understanding of the current TCP/IP environment. You can obtain performance information by first specifying a report class for this workload:

1. Define a report class to WLM.
2. Assign this report class to the TCP/IP workload by adding this report class in the corresponding field on the screen of the classification rules.

When you know how many resources are needed by TCP/IP during its normal peak periods, then you can decide on what can be a reasonable resource group limit (however, still being above the normal peak requirement). The objective is not to limit the resources to TCP/IP unless there is an abnormal demand situation.

Note: The setting described above is not guarantying the dispatching priority, as it is dynamically established among workloads in the same importance category, continuously, depending on the resource shares already obtained by these workloads.

5.4.3 Calculation of the usage limit values

With resource groups you specify either a minimum or a maximum amount of sysplex capacity, expressed in unweighted CPU service units per second. (*Unweighted* means that the SRM constant coefficient is not used.) The CPU service units per second depend on the type of the machine. A list of the CPU service units per second can be found on the following Web site:

`http://www-03.ibm.com/servers/eserver/zseries/srm/`

They can also be found in the manual *z/OS MVS Planning: Workload Management,* SA22-7602.

The Web site is updated whenever a new machine is announced.

For example, we want to limit the maximum resources for the considered resource group to 15% of a z9 with 10 processors:

- In the table we can find the value of 23391.8129 for the CPU service units per second for a 2094-710.
- The total amount of capacity for this machine is 233918.129 SU/sec:

```
(#processors) * (CPU service units/second)
10 * 23391.8129
```

- When multiple systems are involved, the resulting capacity is the sum of each system's capacity.
- For the limit of 15% of the machine, a value of 35088 SU/sec (=233918.129*0.15) is calculated.

5.5 SYSLOGD protection considerations

The SyslogD UNIX application is used to collect the messages issued by the TCP/IP integrated security services and, depending on the directives set up in its configuration file, it collects and sends messages to the MVS console, log files, SMF, other machines, or users.

Note: The SyslogD process creates a UDP socket and binds to port 514 during initialization. This port is reserved to OMVS in the TCP/IP stack profile and is also set aside in the /etc/services file. Local processes create an AF_UNIX socket to communicate with SyslogD for logging purposes. An AF_INET socket is used for logging from and to remote servers. If it is not necessary that SyslogD receive messages from remote servers, which constitutes a security exposure, the SyslogD **`start`** command must be issued with the -i parameter, as explained in "Configuring and starting syslogd" on page 144.

The SyslogD has predefined *facility* names, which can be used for making decisions about where the received messages should go. Messages are also given a *priority code*, which SyslogD can also use to make the decision. What to do with the received message, based on facility name and priority code, is indicated in the SyslogD configuration file.

For instance, in the case of the z/OS Intrusion Detection Services, messages issued by Traffic Regulation Monitor Daemon (TRMD) will be associated with the facility name *daemon,* will be prefixed with *TRMD,* and may have a priority code of:

- 0 EMER
- 1 ALERT
- 2 CRITICAL
- 3 ERROR
- 4 WARNING
- 5 NOTICE
- 6 INFO used by IDS statistics
- 7 DEBUG

The message priority code is set by the application logging into SyslogD and is expected to abide with the above meanings. That is, an application enabled to deliver warning and error messages will log messages with the 4 and 3 priority codes.

There are other considerations pertaining to SyslogD security (for example, in a situation where log data is received from other remote syslogd servers). See the IBM Redbook *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383, for more information. The syslogd configuration file may indicate different destinations for messages based on their facility name or priority code, or both.

Configuring and starting syslogd

The syslogd is started by the **syslogd** shell command:

```
syslogd [-f confile] [-i][-u[-c[-d][-m interval] [-p logpath]
```

Where -f indicates where the syslogd configuration file is located. Each statement in the configuration file is of the form:

```
[userid.jobname.] facility.priority
```

The [userid.jobname] parameters are optional. They are part of the *SyslogD isolation* capability described later. In the lab we used the syslogd configuration file shown in Example 5-3, which tells SyslogD that all messages issued by the user FWKERN, whatever the job name, facility name, and priority are, should be added to file /var/ICAPLOG/icaplog.log. Messages issued by user CSUSER in job TRMD1 go into file /var/TRMLOG/trmlog.log. All other messages are *caught* by the last configuration line and go into file /var/SYSLOG/syslog.log.

Example 5-3 Our SyslogD configuration file

```
FWKERN.*.*.*  /var/ICAPLOG/icaplog.log
CSUSER.TRMD1.*.*  /var/TRMLOG/trmlog.log
*.*      /var/SYSLOG/syslog.log
```

SyslogD can also be started with the PROC shown in Example 5-4.

Example 5-4 SyslogD Started Procedure

```
//SYSLOGD PROC PARMS='-f /etc/syslog.conf -u -i',
// MODULE=SYSLOGD
//SYSLOGD EXEC PGM=&MODULE,REGION=30M,TIME=NOLIMIT,
// PARM=('POSIX(ON) ALL31(ON)',
// '/&PARMS')
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSOUT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
//*
```

SyslogD isolation

The options (-i, -u) that can be specified when initializing SyslogD have the following meanings:

-i	Do not receive messages from the IP network.
-u	For records received over the AF_UNIX socket (most messages generated on the local system), include the user ID and job name in the record.

Using these options permits you to isolate messages into separate logs on the basis of user IDs and job names and to restrict communication with remote syslog servers more easily. This is given the generic term *SyslogD isolation*.

5.6 Overview of z/OS network level security

In this section we describe the z/OS Integrated IP Security functions. Note that how much an organization chooses to exploit of these functions depends upon that organization's security policies and relevant risk assessment.

5.6.1 Protection from network intrusions

It is expected that there will always be at least one firewall device between the z/OS host and the Internet. However, attacks can come from within a firewall, and firewalls can occasionally be compromised. An organization may find it beneficial to improve the attack resistance and detection capabilities of z/OS itself. In 6.7.1, "Intrusion detection overview" on page 168, a discussion of how to improve TCP/IP's security against intrusions can be found.

5.6.2 Controlling the contents of communications

Earlier in this chapter, RACF was discussed as a method for controlling access to the TCP/IP resources. In addition, the IP filtering capabilities of the z/OS TCP/IP stack can be used to control the contents of the communications that flow once access to the resources is granted.

5.6.3 Transport Layer Security and Virtual Private Networks

When a single image of the sysplex is opened to a non-secure network, a secondary problem is exposed: what happens if data classified as secure needs to cross this non-secure network? The z/OS Communications Server provides IPSec Virtual Private Networks (VPNs). Furthermore, the z/OS Communications Server provides, beginning with z/OS V1R7, a variation of Transaction Layer Security (TLS) implementation called Application Transparent TLS, or AT-TLS. This options is discussed in detail in Chapter 6, "Security at the network level" on page 147.

5.7 Controlling TCP/IP address space storage consumption

The TCP/IP address space has to allocate memory for:

- ► Data buffers for transferring data between the network interfaces and applications
- ► Control blocks to map the network entities, connection, and so on
- ► Policies
- ► TN3270 application when it running in the stack

If the system is being attacked, or even during an unexpected peak, the TCP/IP could overuse the system storage and cause availability problems.

The usage of memory should be controlled to prevent the overuse of system storage, specially for the common area. One way to achieve this control could be to monitor the usage of memory by the TCP/IP and Communication Storage Manager (CSM) during peak periods and give 20–50% of allocatable memory above what has been observed.

The usage of storage by the TCP/IP address space can be controlled in two ways:

- ► With parameters in the IVTPRM00 parmlib member
- ► By using the GLOBALCONFIG statement in the PROFILE.TCPIP data set

6

Security at the network level

In this chapter we address at a high level the miscellaneous Security functions in z/OS that can contribute to the Security of the TCP/IP traffic and the protection of z/OS itself as a potentially exposed TCP/IP host.

We explain the following best practices and TCP/IP mechanisms that aim at providing proper network level security:

- TCP/IP *stack hardening*
- IP filtering
- IPSec virtual private networks (VPNs)
- z/OS V1R7 Communications Server Application Transparent TLS (AT-TLS)

Connecting a TCP/IP host to a non-secure network also brings consideration about proper protection if this host is also to be connected to a more secure network such as an intranet. This is where the concept of DeMilitarized Zone (DMZ) comes to play. In this chapter we make the assumption, most probably a theoretical one, that there is no connection with another network of higher security and therefore we do not address the DMZ concept, which is discussed in Part 3, “All Sysplex members with network connectivity” on page 175.

6.1 z/OS TCP/IP stack hardening

Hardening is a term that describes what should be done, in terms of operating system and communications setup, on a system that is going to be connected to a non-secure network. Although some systems propose hardening tools (that is, automated or semi-automated ways to achieve this kind of setup), hardening can generally be seen as a set of best practices intended to secure a TCP/IP host.

Below we provide a non-comprehensive list of these best practices, focusing on the most important ones and providing information about what they translate into in the z/OS platform.

Selecting which services are to be provided by the hardened system, the objectives are:

- To keep the system at the bare minimum of required interactions with the network (that is, strictly disable all services that are not required). Typically, on z/OS and other platforms, it is achieved by controlling who can start and who can use these services, along with access control of the TCP/IP resources such as ports and stacks, which are to be used to propose the services to the outside world.
- To pay particular attention to the services provided that are known, because of their design or reference implementation, to be possible vectors of attacks against the system itself or to other connected systems (for example, routing, DNS, FTP, and so on). Ancillary services that may establish TCP/IP connections (for example, Syslogd, ping, traceroute) during their normal operations are also to be looked at.

  A reference implementation is an early implementation of a standard, usually focusing on the functional aspect and leaving other non-directly relevant aspects, such as Security can be, incompletely implemented. Experience shows that follow-on implementations by vendors are strongly inspired by the reference one, up to the point where its flaws can be duly reproduced and carried over.

We have already discussed in 5.2.1, “TCP/IP resources as protected by RACF” on page 134, the capability of locking or controlling access to TCP/IP resources by local applications. In the following sections we focus on security mechanisms pertaining to the data *on the wire*. The protection mechanisms that are described here are:

- IP filtering
- IPSec Virtual Private Networks
- Transaction Layer Security (TLS) and the z/OS Application Transparent TLS (AT-TLS)

6.2 IP security filtering

In this section we discuss IP security filtering.

6.2.1 The relevance of IP filtering in our configuration

In this section we address the case where only one member of the Sysplex has access to a non-secure network, and we therefore want to make sure that there will be no IP traffic flowing from this member to other members in the Sysplex. What we want to achieve is shown in Figure 6-1.

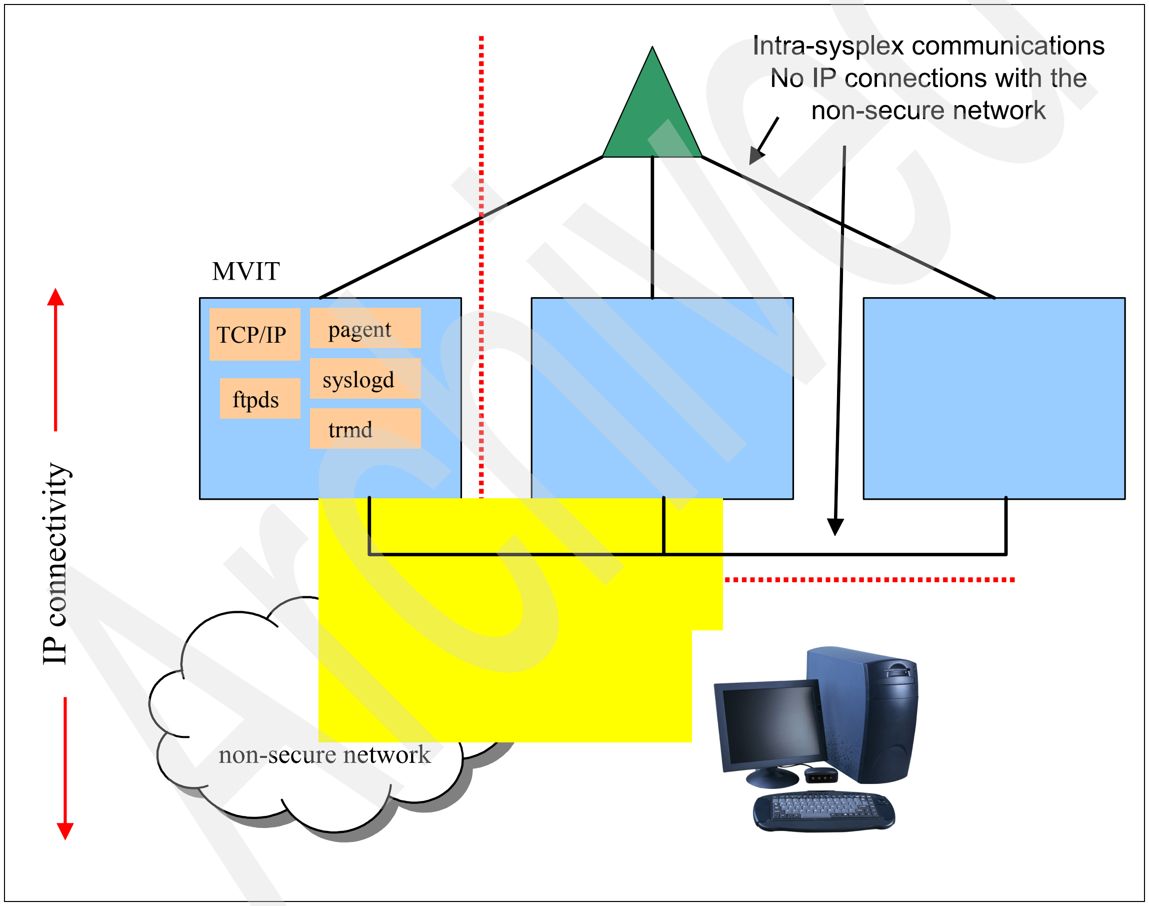

Intra-sysplex communications
No IP connections with the non-secure network
MVIT
TCP/IP
pagent
syslogd
ftpds
trmd
IP connectivity
non-secure network

Figure 6-1 Only one image connected to the network scenario

Note: An example of IP filtering setup is given in Appendix C, “IP filtering implementation and management example” on page 221.

6.2.2 IP Filtering concepts

From the Security assessment standpoint networks usually fall into one of the following categories:

- Internet - Any uncontrolled entity can be an Internet host.
- Extranet - TCP/IP hosts are owned (not implying that they are controlled) by business partners and customers.
- Intranet - The TCP/IP hosts belong to the corporate network, and as such they are expected to be under some level of control.
- Demilitarized Zone (DMZ) - The network segment is deliberately exposed to partners hosted in a less secure environment (actually a fully non-secure environment when the DMZ is facing the Internet).

One of the network security best practices is to consider any type of network as non-secure and to implement in a cost-effective manner (that is, with proper relevance to the evaluated risks) security mechanisms. Even an intranet has different requirements for security for different users. Each department, branch offices, and sometimes customers, will have different security requirements, and they can be enforced by implementing IP filtering functions and policies.

IP Filtering controls whether IP packets received, or intended to be sent, by a TCP/IP stack should flow as intended or should be rejected by the stack. The filtering decision, enforced by the TCP/IP stack internal logic, is based on a security policy or a set of rules that establish the intended Security level of a network environment. The rules state whether the packet should flow as intended or must be stopped right at the stack based on parameters such as the packet origin or destination addresses/ports, the protocol the packet carries, whether the packet final destination is the stack’s own IP address or if it should be routed by the stack, and so on.

When an IP packet matches the conditions described in a rule, a relevant action is performed by the filtering mechanism that can be:

- Permitting the packet to go into or out from the stack.
- Deny the packet, which means not letting it go (implying by the same token not telling the source of the packet about what has happened).
- If the packet carries the IPSec protocol, trigger the authentication of the packet and/or encryption/decryption of the payload.

- Log the event and the resulting decision as specified.

From the implementation standpoint the filtering rules are applied as the packets arrive or leave a stack interface, as shown in Figure 6-2.

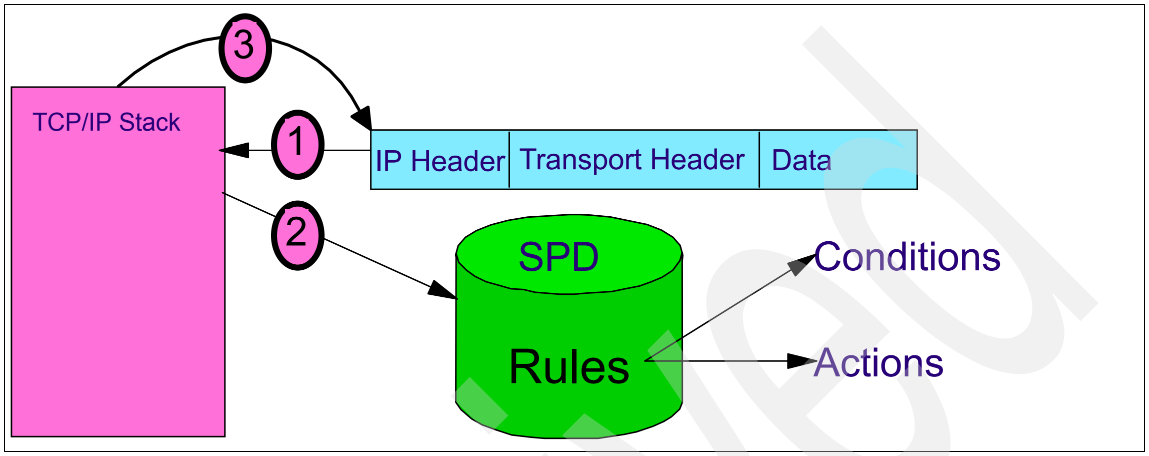


Figure 6-2 IP filtering at work

When a packet arrives at a stack network interface the IP filtering code running in the TCP/IP stack searches for rules into the security policy database (SPD) that match the packet characteristics and then performs proper actions.

In order to create the IP filtering policy (that is, the filtering rules) we have to know what resources the network is made of.

The network resources that are taken into consideration are:

- The TCP/IP stack with its network interfaces and their respective IP address.
- The local servers or clients that are using the stack through a TCP/IP port.
- The network communication flow itself, with its inbound or outbound direction and which end initiated the communication.

The network resources external to the z/OS TCP/IP stack are:

- The known clients and servers that are expected to connect to the z/OS TCP/IP host, specified by their corresponding remote port numbers and IP addresses
- The IP networks and subnets that are hosting potentially acceptable or unacceptable partners

The IP filtering policy is the set of statements that establish relationships between all these resources for the TCP/IP stack to make the decision to accept or deny letting an IP packet flow over the network connection.

This is an example of an IP filtering rule in common language: permits host A to the service B on host C. This indicates that packets with an origin address A and a destination port B at address C are to flow freely, implicitly stating that other packets will be denied. Using more specific TCP/IP technology terms, a rule would read:

```
permit 1.1.1.1 to 2.2.2.2 tcp 23
```

This means that we permit the host 1.1.1.1 to access the server listening on port 23, which can be a Telnet server, on the host 2.2.2.2.

Note: From the terminology standpoint the firewall functions are directly driven by *rules*. These rules are in firewall function internal syntax and are automatically derived from a *policy*. The policy format lends itself to an easier syntax understanding and handling by a security administrator and a simplified update and maintenance process.

Furthermore, a policy can reside in a shared repository and be exploited by several TCP/IP stacks with a common policy need.

The amount of rules an IP filtering function is to consult depends on the number of TCP/IP resources that need to be considered and their combinations with the decision condition.

The z/OS Communications Server as a firewall device

Be aware that the IP filtering function in the z/OS Communication Server is intended to be a built-in protection against moderately hostile environments. It is not intended to catch up with the very sophisticated, and frequently updated, technologies that you want to use to directly protect your installation from a very hostile environment such as the Internet. However, it can provide an efficient combination when the z/OS host is in a network that has already reached a certain security level due to another IP filtering solution, like a dedicated firewall device, already operating. In this case the z/OS Communications Server IP filtering can be used as a second line of defense, and is well fitted to the specific filtering needs that can arise in environments such as an intranet.

Tip: We recommend that the IP filtering function available on the z/OS Communication Server be used to control and protect the resources located in the local z/OS image or the Sysplex, as opposed to using this function in an enterprise firewall or Security gateway scheme, where the z/OS Communications Server would just filter communications to be routed to other networks. A routing firewall would probably be more efficiently implemented using a *lighter* dedicated device or host in that case.

6.3 Protecting the data on the wire

In this section we introduce the use of the IPSec VPNs or the SSL/TLS technologies to protect the data flowing over the network.

6.3.1 Introduction to IPSec VPNs and AT-TLS

The TCP/IP stack has a layered architecture. Each layer interfaces with the layer below it. At the top of the layers are IP applications, for example, a Web browser. A graphical representation of the layered architecture can be seen in Figure 6-3.

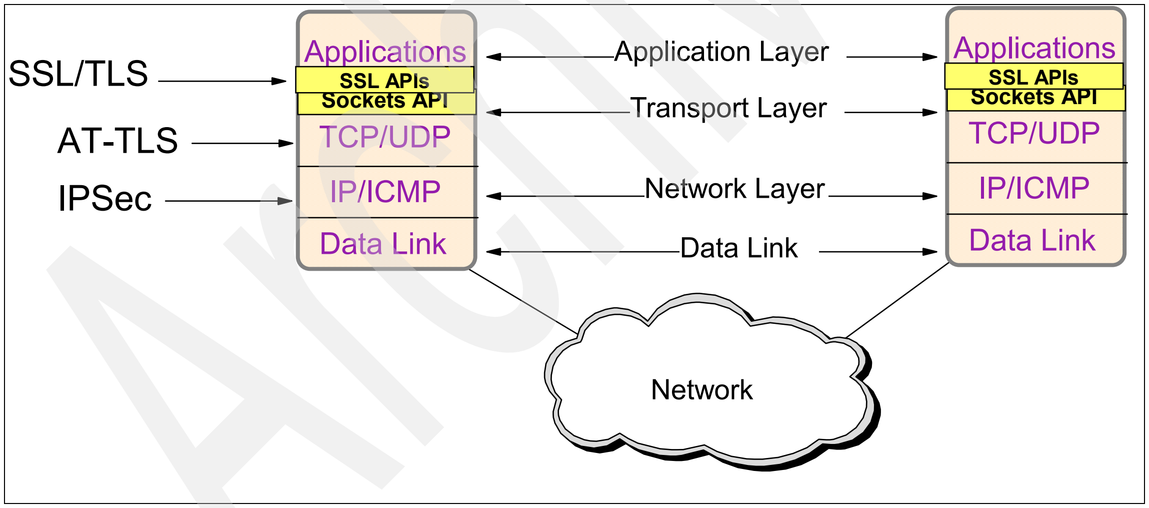


Figure 6-3 Network layers and secure protocols

In z/OS V1R7 the following protections are available in the Communications Server for the flow of data over the network:

- IPsec Virtual Private Networks (VPN) - VPNs are handled by the TCP/IP stack itself at the network layer.
- Application Transparent TLS (AT-TLS) - Transport Layer Security (TLS) is usually being handled by the application code itself, without any involvement of the TCP/IP stack. SSL/TLS application encrypted packets are just regular TCP packets for the TCP/IP stack.

  Beginning with z/OS V1R7, the SSL protocol and cryptographic workload can de directly driven by the z/OS TCP/IP stack without the application being involved. Application Transparent TLS is precisely intended for those applications of which code has not been designed or cannot be upgraded to support TLS.

We explain below the principles of operation of these facilities and provide setup examples in Appendix E, “AT-TLS implementation” on page 253, and Appendix G, “VPN setup” on page 269.

There is one layer left in Figure 6-3 on page 153 that we did not comment yet: the application layer with its SSL/TLS support. This is what we explained above for AT-TLS: it is expected that the TLS protocol and cryptographic processes are being taken care of by the application code. TLS is today superseding Secure Socket Layer (SSL).

We also see that the z/OS V1R7 AT-TLS implementation does not prevent us from continuing to use applications that come with their own SSL/TLS support.

The z/OS VPNs and AT-TLS are activated using policies that are made available to the stack with a specific utility called the Policy Agent (PAGENT). The Policy Agent is explained later in this chapter as a preamble to VPNs and AT6TLS setup information.

Important: These *data on the wire* protections embedded in the z/OS Communications Server apply to all TCP/IP applications ending at the z/OS TCP/IP stack, whatever the physical media is. That is, they apply to IP over XCF, hipersockets, or regular network communications.

6.3.2 z/OS V1R7 Integrated IP Security as compared to the z/OS Firewall Technologies

Important: As of the editing of this book, it has been officially announced that the z/OS Firewall Technologies are discontinued at z/OS V1R8.

Before z/OS 1.7 the IP Filtering and VPN support was spread among several z/OS components. The management and configuration functions were provided by the z/OS Firewall Technologies, which were part of the z/OS Security Server until z/OS 1.5, and then became part of the so-called Integrated Security Services.

Figure 6-4 describes the functional layout of the z/OS Firewall Technologies.

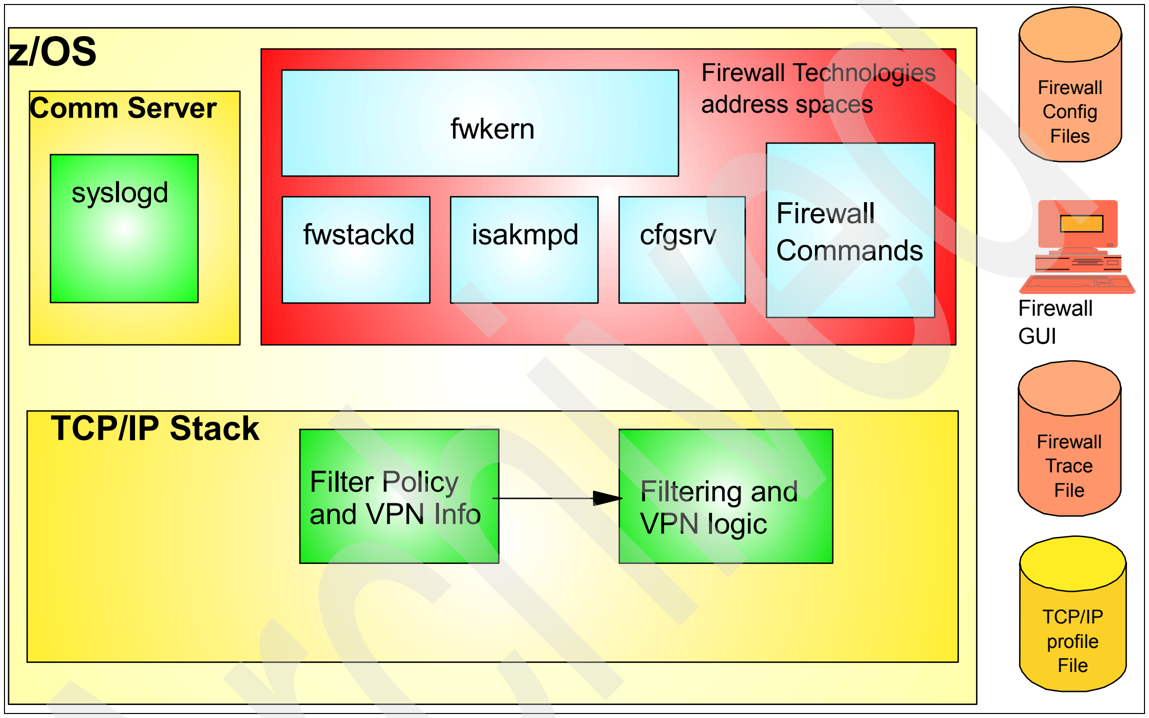


Figure 6-4 IP Filtering and VPN support components in the z/OS Firewall Technologies

This has to be compared to the new IP Security implementation at z/OS V1R7, which is shown in Figure 6-5. Note the higher level of integration, which also shows that the functions are now fully integrated in the z/OS Communications Server package.

Attention: The z/OS Firewall Technologies were providing other functions in addition to IP Filtering and VPNs, which are:

- FTP proxy
- Socks server
- Network Address Translation (NAT)

These functions are no longer available in the new z/OS IP Security functions.

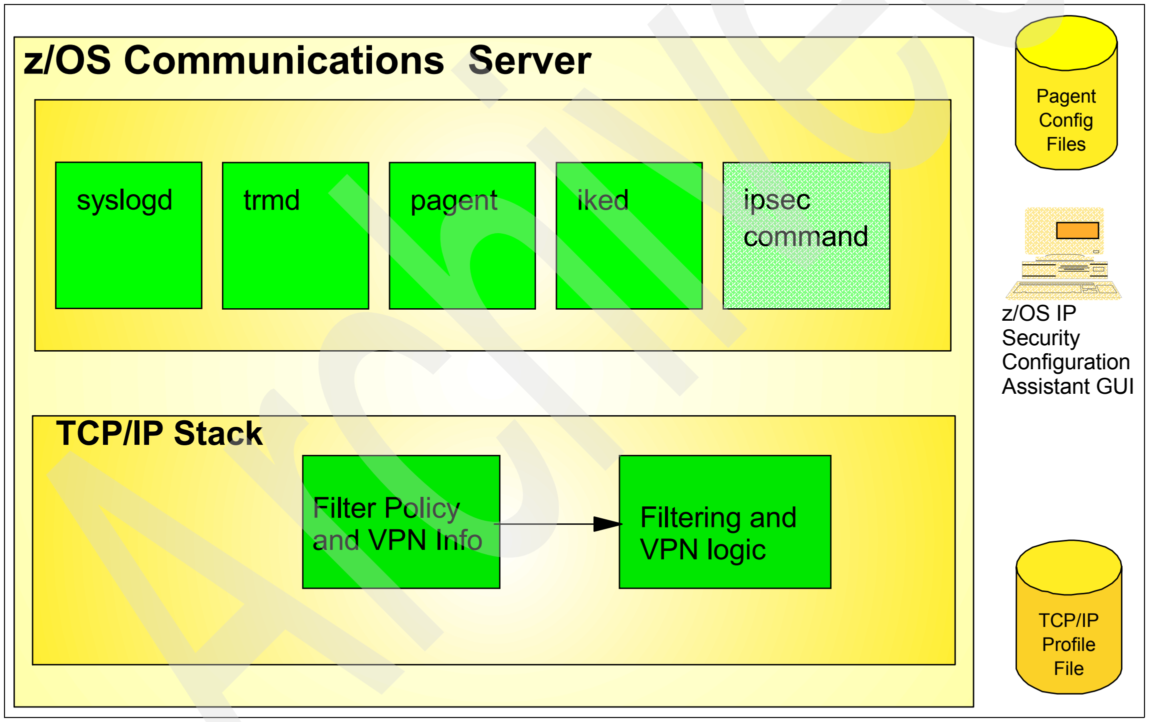


Figure 6-5 New Integrated IP Security components at z/OS 1.7

The z/OS V1R7 IP Security services are under the control of the policies installed in the TCP/IP stacks by the Policy Agent utility. The new z/OS UNIX `ipsec` command is used to manage and monitor the IP filtering and VPN policies.

The Communications Server provides integrated functions to support the IP filtering, the IPSec VPNs, and the Internet Key Exchange (IKE) daemon. This implementation, compared to the previous Firewall technologies, is intended to provide:

- Easier configuration
- Greater scalability
- Improved performance
- Enhanced serviceability

6.4 The z/OS Communications Server Policy Agent

This section discusses the Policy Agent (PAGENT) utility, which is required for setting up the VPN, AT-TLS, IP filtering, and IDS. The Policy Agent fetches relevant data in the policies pertaining to these functions and installs this data, in internal code format, in the designated TCP/IP stacks.

Once the policies are installed in a z/OS stack, the Policy Agent can be invoked by the z/OS UNIX command **`pasearch`**. Using the **`pasearch`** command, a system operator can quickly confirm the status of the policies that have been installed in the TCP/IP stacks.

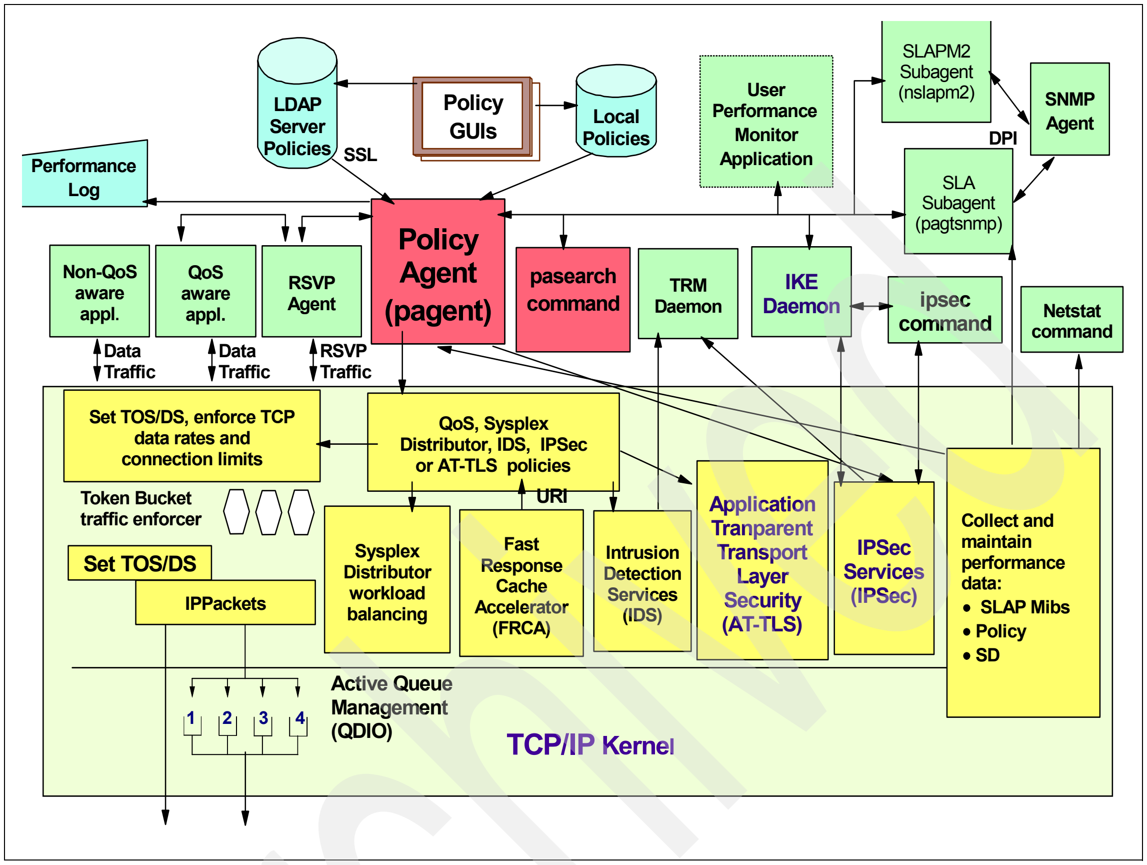

LDAP Server Policies
Policy GUIs
Local Policies
User Performance Monitor Application
SLAPM2 Subagent (nslapm2)
SNMP Agent
DPI
SLA Subagent (pagtsnmp)
SSL
Performance Log
Non-QoS aware appl.
QoS aware appl.
RSVP Agent
Policy Agent (pagent)
pasearch command
TRM Daemon
IKE Daemon
ipsec command
Netstat command
Data Traffic
Data Traffic
RSVP Traffic
Set TOS/DS, enforce TCP data rates and connection limits
QoS, Sysplex Distributor, IDS, IPSec or AT-TLS policies
URI
Token Bucket traffic enforcer
Set TOS/DS
IPPackets
Sysplex Distributor workload balancing
Fast Response Cache Accelerator (FRCA)
Intrusion Detection Services (IDS)
Application Tranparent Transport Layer Security (AT-TLS)
IPSec Services (IPSec)
Collect and maintain performance data:
• SLAP Mibs
• Policy
• SD
1
2
3
4
Active Queue Management (QDIO)
TCP/IP Kernel

Figure 6-6 z/OS Communications Server Policy Agent

As shown in Figure 6-6, quite a large number of components of a TCP/IP environment are linked to the agent.

There are two sources of policy configuration data shown above the policy agent itself. The policy agent can read data either from an LDAP server, for the IDS (Intrusion Detection Services) and QoS (Quality of Service) policies, or from flat files for all policies except the IDS policy, which still requires, as of the writing of this book, to be fetched from an LDAP directory.

To the right of the policy agent is the **pasearch** command. As mentioned before, the **pasearch** command is used to query information about the policies that have been read into a stack by the agent.

Of note also is the box titled IPSec Services. The IPsec services is the service that implement the IPsec VPN. As noted before, the VPN is controlled by policy statements. The z/OS UNIX **ipsec** command allows policy information to be queried and changed. We discuss this further in “A quick look around - pasearch and ipsec” on page 218.

Two other partners of the policy agent warrant a comment here. The first is the Traffic Regulation Monitoring daemon (TRMD). TRMD can be viewed simply as a message and report writer, sending them to Syslogd. TRMD is in charge of event recording for Intrusion Detection Services (IDS), IPsec services, and traffic regulation itself (traffic regulation is a mechanism imbedded in z/OS that can limit the amount of TCP or UDP requests that the stack has to process, by thus limiting the effects of Denial of Service attacks.

The z/OS Network Configuration Assistant is a graphical user interface (GUI) that can be used to define AT-TLS and IPsec, VPN and IKE configuration policies. Once defined at the Configuration Assistant, the policy files and other relevant parameters or JCL files are sent to the z/OS TCP/IP host using FTP. The GUI can be downloaded from the following URL:

`http://www.ibm.com/support/search.wss?tc=SSSN3L&rs=852&rank=&dc=D400&dtm`

The use of the Network Configuration Assistant is documented in “z/OS Network Configuration Assistant” on page 270.

Figure 6-6 on page 158 also shows the IKE daemon (IKED). IKE stands for Internet Key Exchange, a standardized protocol defined in RFC 2409. The IKE daemon facilitates negotiation and exchange of keys to be used for encryption in a VPN. As such, IKED is really a helper application to make management of encryption keys for a VPN an automated process.

Note: An example of a Policy Agent setup is given in Appendix B, “TCP/IP configuration information” on page 215.

6.5 Virtual Private Network (VPN)

Connecting a sysplex member to a non-secure network can introduce the need for securing data that travels over the network. A VPN is one of the solutions available on z/OS to improve security of data on the wire. In this part of the book we consider that only one member of the sysplex is connected to the network. In Part 3, “All Sysplex members with network connectivity” on page 175, we address the situation where all sysplex members have network access, and we see the additional considerations this brings to the utilization of the VPN technology.

Clarifying the terminology

The world of technology has plenty of confusing acronyms. It is appropriate here to make sure that the reader knows what is meant by what term.

A VPN (also called a tunnel) is cryptographic protection applied to one or more logical (virtual) connections between one or more pairs of endpoints on an IP network. The traffic between these two endpoints is authenticated and encrypted. The effect is to have data being carried in a secure fashion over what is classified as a non-secure network.

The term *IPsec* (IP Security) is used to describe the standardized protocol that makes a VPN exist and be exploited. The IPsec standard is documented by the Internet Engineering Task Force (IETF) in documents referred to as RFCs (Request For Comments). The IPsec protocol is complex and is described over very many RFCs. However, the primary document is RFC 2401, which may be found at:

`http://www.ietf.org`

In z/OS V1R7, the replacement mechanisms for the Firewall Technologies fall in the functions category called Communications Server IP Security, and requires, to be enabled, the new statement IPSEC in the TCP/IP profile data set. This IPSEC statement also controls the IP filtering functions built in the stacks at z/OS V1R7.

In addition, VPN, the IP filtering options, and more are controlled by a new z/OS UNIX command called **`ipsec`**.

To summarize:

IPsec	The public standard for VPN communications
IPSEC	The statement in the z/OS PROFILE.TCP/IP data set that enables IP filtering or the IPSec VPNs
ipsec	A z/OS UNIX command used to configure and display filtering and VPN settings

6.5.1 ISAKMP and IKE

An IPsec VPN is to provide encryption, integrity, and authentication, or, to put it into different terms, privacy and integrity of data and identity verification of the communicating partner. Keep in mind, however, that with VPNs the partners, or *endpoints*, are TCP/IP hosts, not applications as would be the case with SSL/TLS.

In order to encrypt data, a VPN uses a symmetric encryption key at each endpoint. There are two ways in which this encryption key can be established:

- Manually, by manually installing the secret keys at both endpoints
- Dynamically, using the Internet Security Association and Key Management Protocol along with the Internet Key Exchange protocol

The manual method has obvious limitations: the sharing of the key must be done by some secure method, like placing it onto a floppy disk and copying it to each endpoint. Mistakes can be made when typing, and a refresh of the key, a mandatory practice to protect against compromising of the key, must also be done manually. And, if the endpoints of the VPN are very far apart, securely transporting the key can be a difficult undertaking.

Although z/OS V1R7 supports manual VPNs, they will not be discussed in this book because dynamic VPNs are considered to consist of a superior method of key exchange.

The Internet Security Association and Key Management protocol (ISAKMP) and its subsidiary, Internet Key Exchange Protocol (IKE), comprise the public standard, defined in RFCs 2408 and 2409, respectively, for the dynamic setting of a VPN key, by providing a secure automated method of VPN endpoints authentication and exchange of the secret encryption keys over a non-secure network.

6.5.2 Security Associations (SA) and VPN

A security association is a one-way, or simplex, specification of the cryptographic algorithms, keys, and processes that the VPN endpoints should adhere to. Since an SA pertains to only one direction of the flow of data inside the VPN, there are two security associations required to identify what processes have to be run at both ends of the tunnel.

Figure 6-7 is a graphical representation of SAs and their contents.

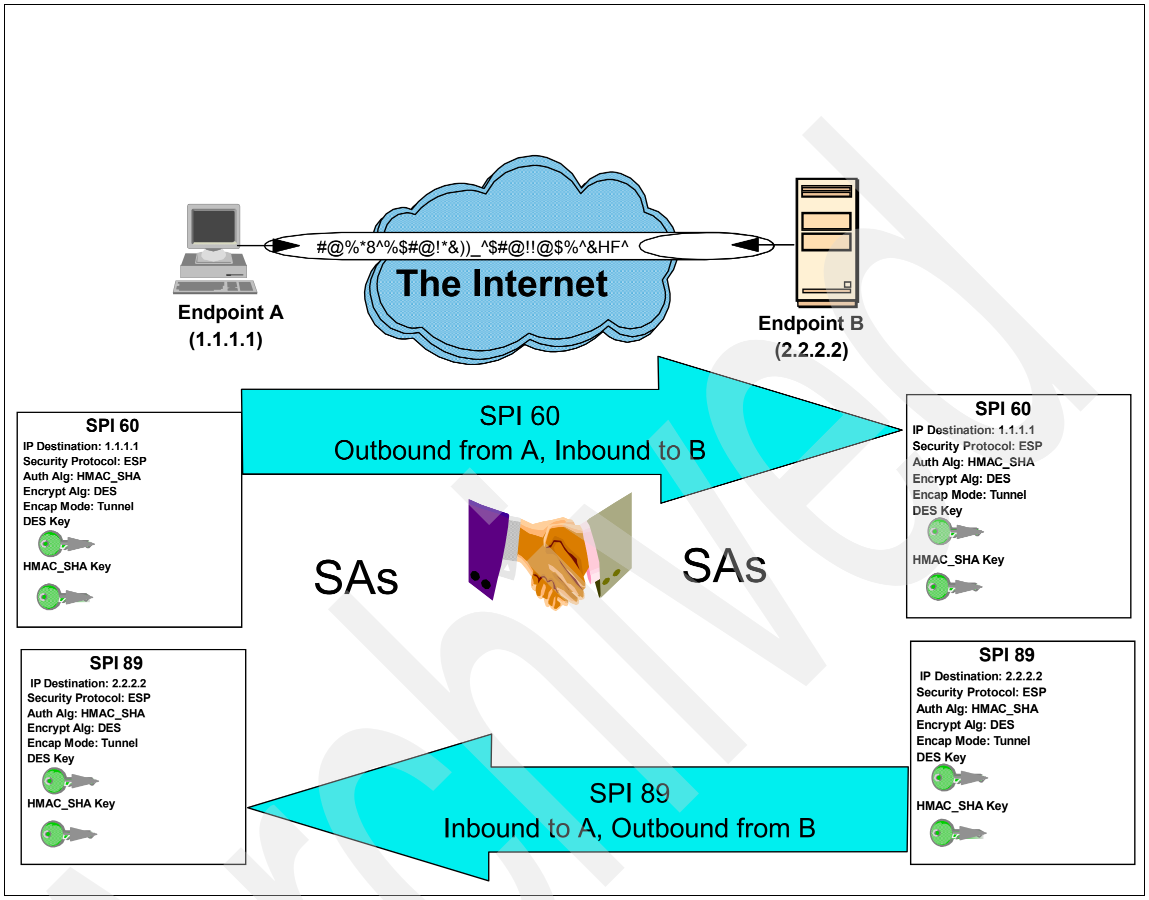


Figure 6-7 Security associations

The security associations are automatically established using the IKE protocols. Once established, SAs are identified by a number, called the Security Parameter Index (SPI).

An SA consists of the following information:

IP destination	The IP address, inbound and outbound of the VPN endpoint.
Security protocol	IPsec allows options as to whether data should be encrypted (Encapsulating Security Payload, or ESP) or protected only from modification (Authentication Header, or AH). AH does not provide any privacy, but the involved

	cryptographic processes are requiring less resources than the ESP protection.
Authentication algorithm	The scope of this text does not include a discussion of different authentication algorithms. A successful negotiation ends up with both ends using the same algorithm to authenticate.
Encryption algorithm	Again, encryption algorithms are outside the scope of this text. A successful negotiation ends up with both ends using the same algorithm to encrypt and decrypt.
Encapsulation mode	In *tunnel mode*, the entire IP datagram (that is, header and data) is encapsulated as a new data field and is given a new IP header. The new source and destination IP addresses are the endpoints of the VPN. In *transport mode*, the original IP header is left intact, implying that the original source and destination addresses are the VPN endpoints.
Keys	The actual encryption keys to use are also part of the SA.

For more details, RFC 2401 provides good background information about all of these terms.

Important: There are known incompatibilities between VPN traffic and the Network Address Translation (NAT) function, leading in some cases to the impossibility of properly establishing VPNs if a NAT device is located between the two endpoints. You must be aware of any NAT device that could be traversed by the potential tunnels, and then refer to the VPN implementation reference documentation to determine whether you are facing impossibility to properly operate the tunnel. For z/OS the document to refer to is *z/OS Communications Server IP Configuration Guide*, SC31-8775.

The z/OS V1R7 IPsec VPN technology supports the NAT Traversal protocol that can help in solving some of these cases.

6.5.3 VPNs and certificates

The dynamic tunnel method of establishing a VPN can use several options to authenticate both IKE daemons (that is, both endpoints of the VPN). One of these options involves an exchange of digital certificates. It is outside of the scope of this text to go into details about how certificates work. There are numerous, and lengthy, texts on this topic. The digital certificate format is the x.509 V3 format.

Each endpoint sends a certificate request to the other end with information describing the certificate authority certificate that it will accept. In turn, the individual certificate signed by this authority is sent out in response.

Note: Examples of VPN Implementation are given in Appendix G, “VPN setup” on page 269.

6.6 Application Transparent TLS (AT-TLS)

In this section we address the AT-TLS implementation provided in z/OS V1R7. With AT-TLS the z/OS TCP/IP stack can provide TLS support to applications that do not have TLS support implemented in their own code.

Note: Secure Socket Layer (SSL) was developed by Netscape and the current version, Version 3.0, has been available since 1996. Transport Layer Security (TLS) is a secure protocol that supersedes SSL.

TLS is an IETF standard protocol described in RFC 2246, and although based on SSL V3.0, it is not interoperable with SSL. However, experience shows that current products that support TLS can fall back to SSL communication should the partner not support TLS.

SSL/TLS - why and how

When the application is using a non-secure network to send and receive data, some mechanism of protection is recommended. SSL/TLS is a secure client/server protocol that is used by the application to preserve the confidentiality of data with encryption, to authenticate the server it is connected to, and optionally to have the client authenticating using a digital certificate.

The SSL/TLS protocol uses well-known encryption and hashing protocols like:

- Encryption Symmetric Key algorithms: DES, T-DES, RC4, AES
- Hashing algorithms: MD5 and SHA-1
- Encryption Asymmetric Key algorithms: RSA, Diffie-Hellman and DSA

The protection provided by SSL/TLS differs from the IPSec VPN in that:

- It is intended for application-to-application communication, as opposed to a VPN that protects a TCP/IP host-to-host communication.
- The SSL/TLS flow does appear as normal TCP communication to the network entities. SSL/TLS is not a TCP/IP registered protocol as TCP, UDP, and

ICMP are. IPSec is a registered protocol (protocol 50 and 51) so that network entities know the packet is transporting IPSec protected data and can make decisions based on this.

Note that SSL/TLS can flow within a VPN. The VPN ends at the TCP/IP stack network layer, while the SSL/TLS communication ends at the application layer.

z/OS V1R7 offers the new AT-TLS function. It is intended for those client/server applications not supporting SSL/TLS, whatever the reason is, for which you wish to protect the TCP/IP socket communication with TLS. With AT-TLS the TLS protocol is being handled by the TCP/IP stack itself.

6.6.1 AT-TLS concepts

z/OS has been providing, and is still providing, the System SSL API for applications that do not have their own SSL/TLS code but still have to support the protocol. System SSL is widely used by SSL/TLS clients and servers provided with z/OS. System SSL has two specific features:

- It transparently invokes the hardware cryptographic coprocessors, if in operation in the system, to get hardware assistance for the SSL/TLS cryptographic processes.
- Its API is intended for the C/C++ language.

Applications running on z/OS can still have their own SSL/TLS support code. However, if not using the System SSL API, it will not get the benefit of the implicit cryptographic coprocessor hardware assist.

With AT-TLS an application that does not support SSL/TLS, or has the support but is not using the z/OS System SSL API, can transparently get access to System SSL via the TCP/IP stack.

Note that AT-TLS, with its setup parameters, allows a modular involvement of the application with the on-going TLS communication. The application can completely ignore that TLS is being performed, and is therefore not involved at all (that would be the case for applications without any SSL/TLS support, but also for applications with their own support that for diverse reasons we do not want to use), or the application can intervene in the protocol execution sequence to grab, for instance, the digital certificate provided by the partner.

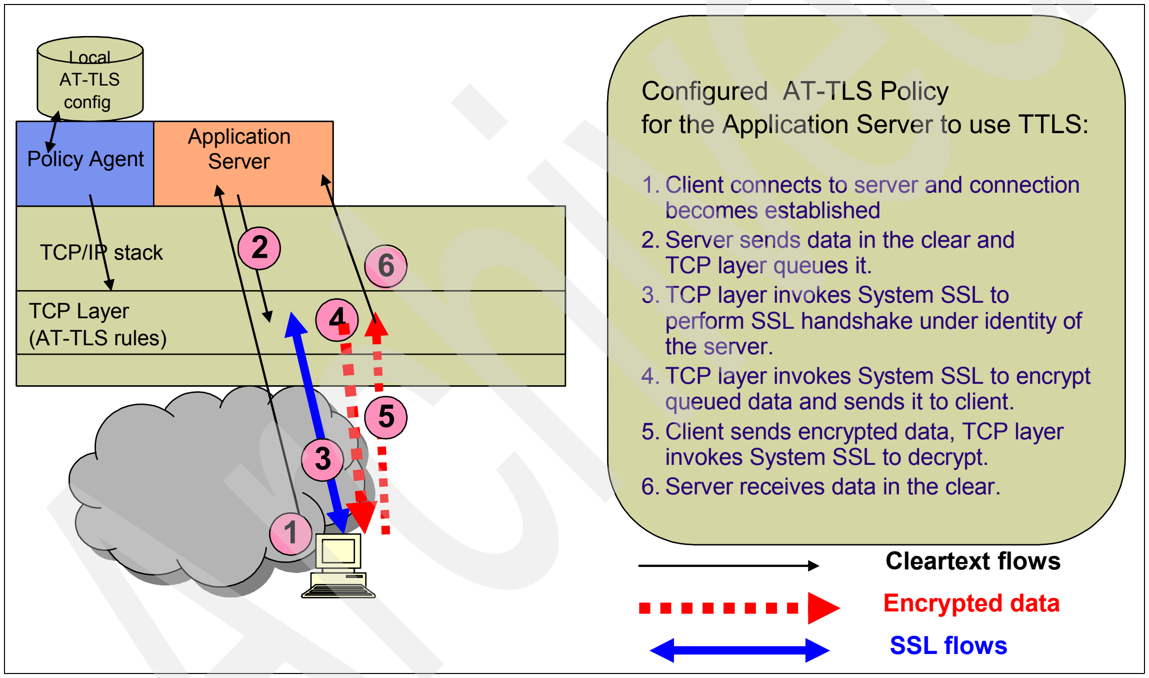
The AT/TLS provides a transparent layer where all the SSL/TLS functions are implemented and defined outside the application in the TCP/IP address space.

6.6.2 AT-TLS z/OS implementation

The z/OS V1R7 AT-TLS function involves several components:

- The TCP/IP stack address space
- The PAGENT utility
- System SSL

The TCP/IP address space is responsible for handling all of the TLS-related functions like the handshake, authentication, and encryption and decryption of the data and session key management via calls to System SSL. All functions are performed based on user-defined policies and loaded into the TCP/IP stack by the PAGENT. A high-level description of the AT-TLS implementation and data flow is given in Figure 6-8.



Configured AT-TLS Policy
for the Application Server to use TTLS:

1. Client connects to server and connection becomes established
2. Server sends data in the clear and TCP layer queues it.
3. TCP layer invokes System SSL to perform SSL handshake under identity of the server.
4. TCP layer invokes System SSL to encrypt queued data and sends it to client.
5. Client sends encrypted data, TCP layer invokes System SSL to decrypt.
6. Server receives data in the clear.

Figure 6-8 AT-TLS flow

In Figure 6-8 the application server does not have to know anything about the SSL/TLS protocol. Every function required by the TSL protocol will be performed transparently by System SSL upon requests by the TCP/IP stack address space. For complete information about the AT-TLS configuration and implementation see the following manuals: *IP Configuration Reference*, SC31-8776, and *IP Configuration Guide*, SC31-8775.

The TCP/IP stack address space intercepts any connection being made from the z/OS image or from the network and makes a decision whether the connection should use TLS based on the AT-TLS policy.

Note that the client application shown in Figure 6-8 on page 166 should support TLS, or it can be a non-TLS-enabled application that runs in z/OS V1R7 and is also taking advantage of AT-TLS.

For the sake of Security, applications may have to make decisions based on the current characteristics of the TLS session and take appropriate action. This is possible with AT-TLS. The degree of involvement of the applications with the AT-TLS operations is shown in Figure 6-9.

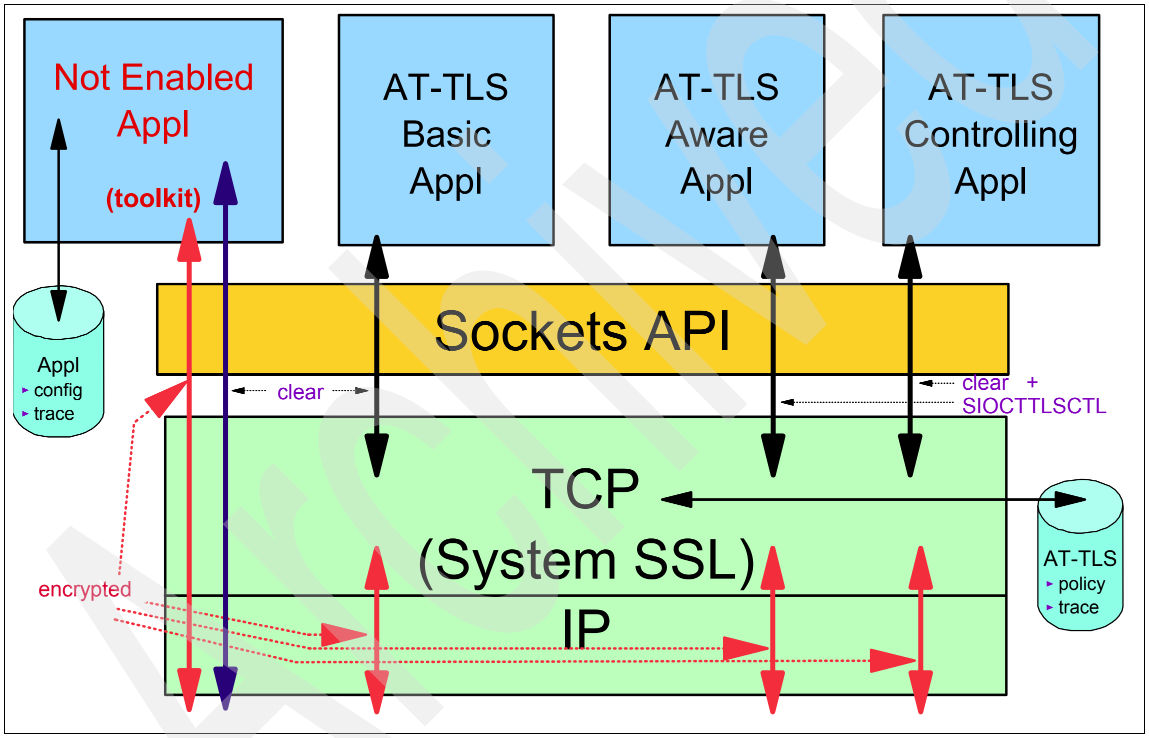


Figure 6-9 AT-TLS application types

The application categories are:

- Not-enabled applications, that is, applications that cannot be assisted by AT-TLS:
  - Applications written in the Pascal API or Web servers using the Fast Response Cache Accelerator feature.

  - No policy is defined to them or a policy explicitly disables the usage of AT-TLS.
  - Applications already using the System SSL API.
- Basic applications. These are applications that can be transparently assisted by AT-TLS:
  - There is a policy specifying that they are enabled for AT-TLS usage.
  - They are unchanged and unaware of AT-TLS.
  - The application protocol is not affected by the usage of AT-TLS.
- Aware applications. These are applications that are getting information from the TLS context and protocol data:
  - There is a policy specifying that they are enabled for AT-TLS usage.
  - The application is changed to use a socket API called SIOCTTLSCTL to extract information from AT-TLS like the policy status, the negotiated SSL/TLS version and cipher specifications, and the partner's certificate.
- Controlling applications. These are applications that are participating in some way to the execution of the TLS protocol:
  - There is a policy specifying that they are enabled for AT-TLS usage and to control the usage of AT-TLS on the session.
  - They can start the session in clear and later decide to start the SSL/TLS session.
  - The SIOCTTLSCTL is used to extract information and to control the AT-TLS like starting a secure session and resetting the cipher or the session.

Note: An example of AT-TLS implementation is given in Appendix E, "AT-TLS implementation" on page 253.

6.7 z/OS Intrusion Detection Services

In this section we discuss z/OS Intrusion Detection Services.

6.7.1 Intrusion detection overview

In network security terminology, *intrusion* globally designates anomalous, and potentially malicious, activities. The objective of an intrusion may be to acquire information that a person is not authorized to have. It may be to gain unauthorized use of a system as a stepping stone for further intrusions elsewhere. Or it may be to cause business harm by rendering a network, system, or application unusable. Most intrusions follow a pattern of information gathering, attempted access, and then destructive attacks.

An intrusion detection system (IDS) can be network-based, in that it analyzes the data flowing over a segment of the network, or it can be host-based when performing the analysis within the TCP/IP stack of a network host system. It can also be a mix of both technologies, comprising sensors located in network segments and on the host's TCP/IP stacks.

6.7.2 Network-based intrusion detection

Sensors, also called *IDS probes*, analyze the network data against known *signatures*, meaning IP datagram headers or data patterns known to be used for intrusion. Because the probes are scattered over the network, and a single probe view is usually not enough to assess the real danger of the observed IP traffic, network-based IDS also involves the use of correlating devices, such as the Tivoli® Risk Manager. These devices raise an alert if the observed traffic is deemed to be a real alarm (not a *false-positive*, in IDS terminology) based on information collected from several IDS probes.

6.7.3 Host-based intrusion detection

Host-based intrusion detection relies on additional capabilities on the host TCP/IP stack to analyze the received IP packets against known intrusion characteristic patterns. Host-based IDS presents the following advantages when compared to network-based IDS:

- The ability to evaluate inbound IPSec data from when the data is first decrypted in the target stack before intrusion analysis.
- The overhead of per-packet evaluation against a table of known attacks is avoided since the target stack can internally detect the anomalous condition and then make a decision based on the IDS policy it has to apply.
- Statistical anomalies can be determined based on the target stack internal thresholds or state data.
- Prevention methods can be applied by the target stack as per the provided IDS policy.
- Globally speaking, there are also fewer false positives with host-based IDS.

But, again, an installation will probably want to take advantage of both implementations by integrating network-based and host-based IDS in their network layout.

6.7.4 The z/OS Intrusion Detection Services

z/OS IDS is an enhanced real-time host-based Intrusion Detection Service (IDS) using policy control to identify, alert, and document suspicious events and assist in their analysis. Traffic regulation is provided for TCP and UDP traffic via the Traffic regulation Management Daemon (TRMD). Messages about possible security violations can be sent to a log file and also sent directly to the console. In summary, the z/OS IDS is used to:

- Detect and record scanning, common attacks, and flooding.
- Record suspicious events to syslog, console, or trace files.
- Set up policies to define preventative measures (meaning queue and connection limits) against attacks and floods, such as multi-packet Denial-of-Service attacks.
- Gather data for possible legal actions.

The z/OS IDS layout in an installation is shown in Figure 6-10. The IDS is configured with an IDS policy that defines the intrusion events to monitor along with the actions to take. The IDS policy can only be stored in an LDAP directory and the Policy Agent (PAGENT) reads the policy from the LDAP server. The policy definitions are processed by the Policy Agent and installed in the TCP/IP stack.

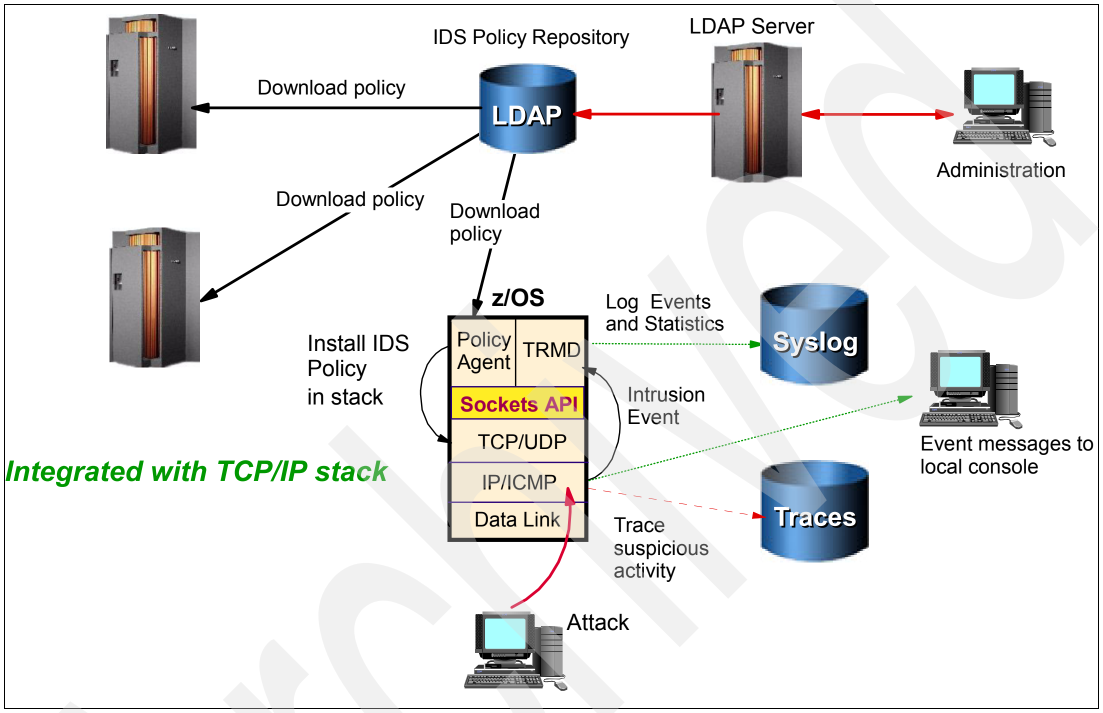


Figure 6-10 Overview of the z/OS IDS infrastructure

6.7.5 The z/OS IDS policy

The IDS policy is a set of definitions, entered as attributes in an LDAP directory, that describe to the TCP/IP stacks what event types to monitor, which criteria to apply for issuing alerts, and when to trigger logging and preventive actions.

The supported intrusion event types are described in this section.

Scanning

The intent of scanning is to map the target of the attack by collecting information about subnet structure, addresses, masks, addresses in-use, system type, operating system, application ports available, and software release levels.

Attack

The intent of an attack is to crash or hang the system by sending single malformed packets in an attempt to exploit known weaknesses in the target's system TCP/IP stack or by flooding the target system with multiple packets sent in high volume.

Traffic regulation for TCP connections and UDP receive queues

This is actually a preventive measure in case of a high volume of connection requests arriving at the target host, which could be intended to flood a system or could just be an unexpected peak in valid requests.

It is possible to log suspicious events to the MVS console or syslogd. Statistics are logged to syslogd by the enhanced Traffic Regulation Management Daemon (TRMD). The TRMDSTAT command can be used to summarize and display syslogd messages. It is also possible to trace suspicious packets to the new SYSTCPIS component trace log. You must use IPCS to format the SYSTCPIS trace. PAGENT also logs its activity via SyslogD.

Figure 6-11 shows an overview of the interactions between the z/OS IDS components.

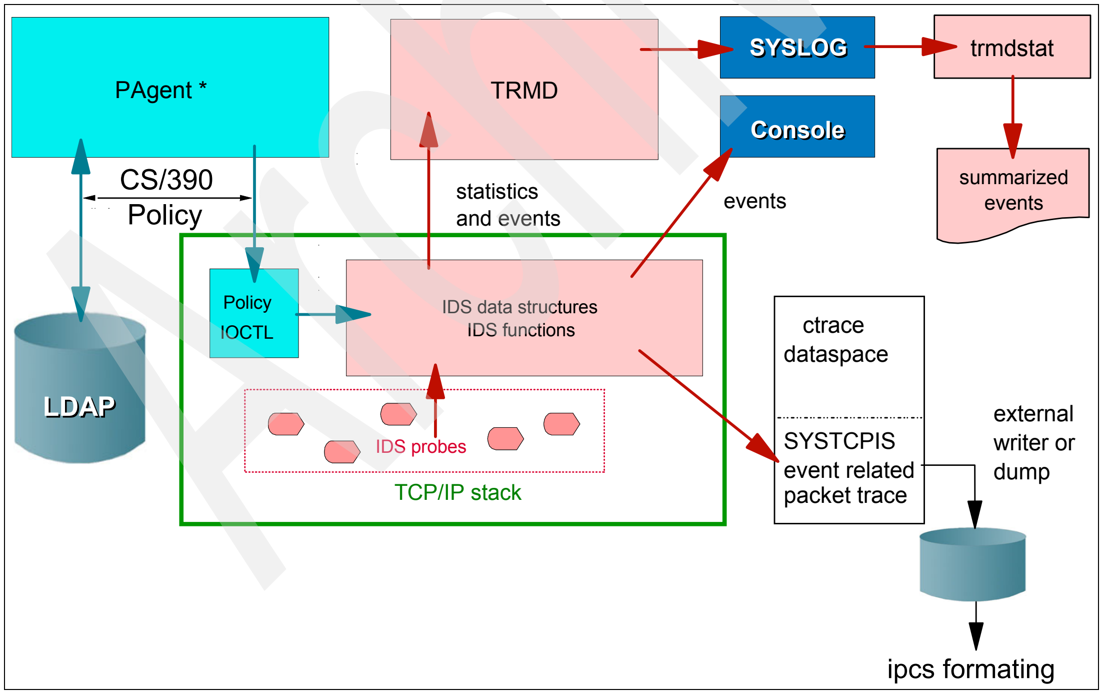


Figure 6-11 Intrusion Detection Services and IP stack overview

Clarifying the notion of an IDS event

An IDS *event* is a countable occurrence of a situation matching a set of conditions defined explicitly or implicitly in the policy. The IDS event can trigger an action or is part of the statistics gathered by TRMD, depending on the active IDS policies. The countable IDS event is dynamically given a *correlator number* by the TCP/IP stack code that is used to tag the finer detailed information, such as traces, that go along with the event.

Example 6-1 shows messages issued by TRMD and displayed at the MVS operator console.

Example 6-1 TRMD messages at the MVS Operator Console

```
EZZ8762I EVENT TYPE: TCP PORT CONSTRAINED
 EZZ8763I CORRELATOR 3 - PROBEID 01004400
 EZZ8764I SOURCE IP ADDRESS 9.139.240.34 - PORT 0
 EZZ8765I DESTINATION IP ADDRESS 0.0.0.0 - PORT 80
 EZZ8766I IDS RULE trtcpHttp-rule
 EZZ8767I IDS ACTION trtcpHttpLog-action
 EZZ8761I IDS EVENT DETECTED 544
```

Example 6-2 shows messages received and stored by SyslogD.

Example 6-2 TRMD SyslogD stored messages

```
EZZ9321I TRMD TCP constrained entry:02/01/2002
21:16:37.18,lhost=0.0.0.0,port=80,host=9.139.240.34,available=0,total=1,percent
=50,correlator=3,probeid=01004400,threshold=0
```

6.7.6 TRMD

The Traffic Regulation Management Daemon manages the IDS message collection. It can be run as a started task or started from the OE Shell. There must be one TRMD instance per TCP/IP stack in the z/OS image. The affinity between the TRMD instance and the stack is indicated in the RESOLVER_CONFIG environment variable.

Part 3

All Sysplex members with network connectivity

In this part we assume that all the sysplex members can be reached from the non-secure network.

Setting up proper security translates into expanding, to other sysplex members and intra-sysplex TCP/IP communications, the protections that we have seen in Part 2, “One Sysplex member with network connectivity” on page 129.

It also addresses the sysplex-specific features of SSL/TLS session ID sharing and VPNs Sysplex Wide Security Association (SWSA).

7

All Sysplex members with network connectivity

This chapter describes the security implications of giving an entire sysplex access to a non-secure TCP/IP network via a sysplex TCP/IP communications distribution scheme where any participating member could be eligible as an endpoint for IP communications. It assumes that the security considerations discussed in both Part 1, “Basic Parallel Sysplex security” on page 15, and Part 2, “One Sysplex member with network connectivity” on page 129, have already been taken into account, as they remain the mandatory ground on which to build additional protections.

7.1 VIPA and Sysplex Distributor

In this chapter we explain the concept of the z/OS Virtual IP Address (VIPA) and how it is used for the implementation of Sysplex network workload distribution.

7.1.1 Virtual IP Address (VIPA)

A VIPA is actually made of a virtual device and link pair. By virtual, we mean that the device and link do not refer to any physical adapter. Instead, a VIPA exists only in software. The advantage is that the VIPA, as it is not linked directly to a physical network adapter, is able to maintain an availability that might not otherwise be possible.

Note, however, that the IP address associated with this VIPA device and link statement is by no means virtual for all network entities. The VIPA IP address is advertised, as any other addresses, by the routing services.

The simplest availability scenario here is to create a single VIPA on a z/OS host that has two or more physical adapters connected to the network. IP applications in the network would be instructed to use the VIPA address for connecting to the host. If a physical adapter fails, traffic, via a dynamic routing protocol such as OSPF, could be rerouted to the VIPA using one of the remaining available physical adapters.

Dynamic VIPA

A dynamic VIPA (DVIPA) is a VIPA that can be moved from one TCP/IP stack to another stack, therefore still keeping the same IP address although changing host. A dynamic VIPA can be distributed in a Sysplex, via the Sysplex Distributor technology, so that several members of the Sysplex can be designated as hosting the distributed DVIPA.

Instead of defining a DVIPA using device and link statements, a DVIPA is defined in the VIPADYNAMIC statement in the PROFILE.TCPIP dataset, with the VIPADEFINE keyword. The VIPADEFINE statement activates the DVIPA in the same fashion as device and link statements do for VIPA. The TCP/IP stack with these definitions is referred to as the primary DVIPA.

Other TCP/IP stacks running in the Sysplex can use a VIPADYNAMIC BACKUP statement in the PROFILE.TCPIP to specify that they are candidates to host another instance (that is, a backup instance) of the DVIPA.

The backup hosts are ranked according to which member in the sysplex should be the next to take over the DVIPA in the event of a problem. In the event of a problem, a member failure, or even slow response times on the TCP/IP stack, the

DVIPA will be removed from the failing host and reactivated on the next backup stack.

Again, OSPF will do the work if necessary to maintain proper routing information leading to the stack currently hosting the DVIPA.

Sysplex Distributor

The Sysplex Distributor exploits the DVIPA concept to balance workload arriving via TCP/IP among members of the Sysplex. One member in the sysplex is designated as a distributing host. The TCP/IP stack on this image is configured to distribute specific TCP/IP traffic, on the basis of the destination port number, to target hosts in the Sysplex. When a new connection request arrives at the distributing host it makes a decision about which target host to send the request to. This decision, depending on the setup can be influenced by WLM weighting or by TCP/IP Quality of Service policy. Or more simply the distribution is made on a round-robin basis.

In summary, the Sysplex Distributor provides:

- A built-in network workload dispatching mechanism that can also cooperate with external network dispatcher devices.
- A single network-visible IP address of a sysplex cluster service. One IP address can be assigned to the entire sysplex cluster, although the service can be provided by different sysplex members.
- The Sysplex Distributor will query the Policy Agent to find whether there exists any policy defined for routing the incoming connection requests.
- WLM and the QoS policy can be specified for workload balancing in real-time on every new connection request.
- A backup capability so that in case of failure of the distributing IP stack, the connections distributed to other IP stacks in the sysplex will not be disrupted.
- Commands that are available to display both the connection routing table and destination port table information, showing information of configuration and current connection distribution.

Figure 7-1 show a Sysplex Distributor configuration where an inbound packet arrives at LPAR1, which is a distributing host. LPAR3 is the selected target. The inbound packet is redirected to the target host by the distributing host.

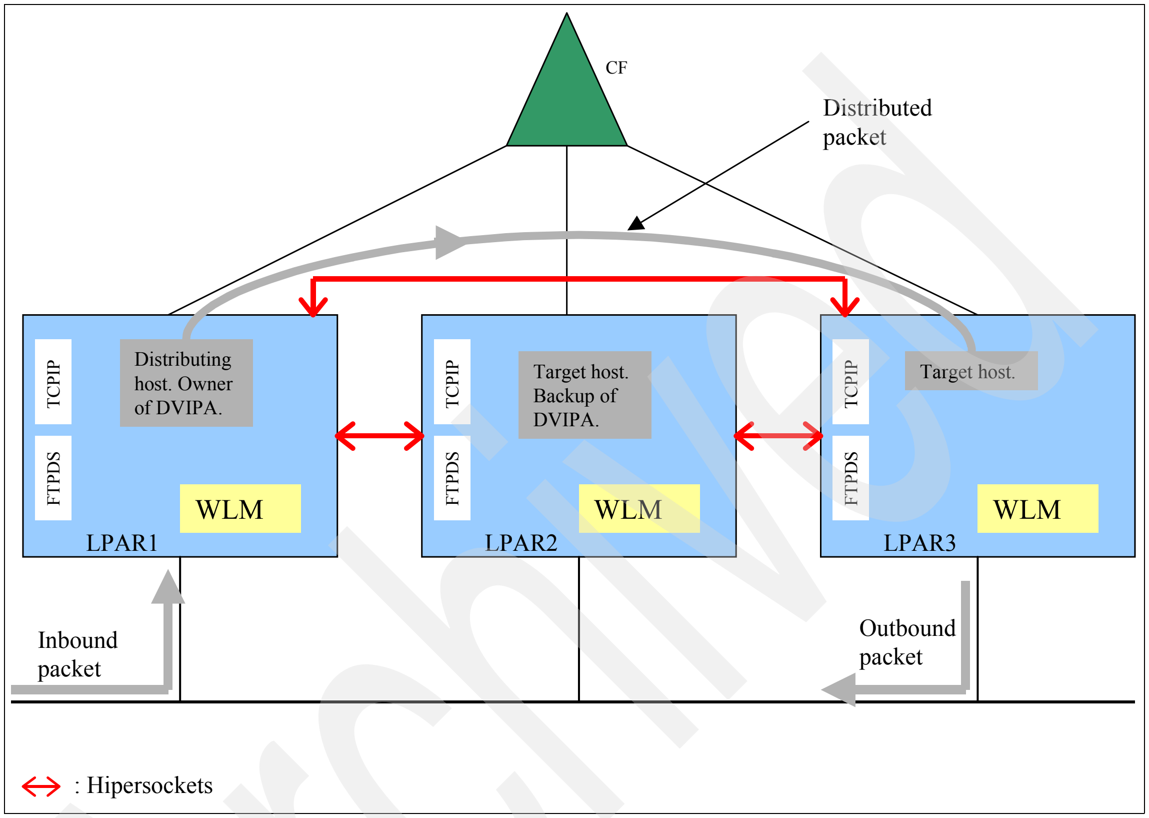


Figure 7-1 Sysplex Distributor

Notice the route taken by the distributed packet in Figure 7-1. It arrives at LPAR1 from the network, then makes its way to the target LPAR3, and from there the response goes directly back out onto the network.

In z/OS releases prior to z/OS V1R7, the target IP address specified in the distributing host had to be a dynamic XCF address (that is, implying an IP traffic carried only by the XCF links) defined via the TCP/IP profile DYNAMICXCF statement. With z/OS V1R7, the target IP address used as a distributed connection could be defined using a VIPADYNAMIC VIPAROUTE statement. In that case the target members can be reached via other means than IP over XCF, like an actual LAN connectivity between the distributor and the potential target members.

This puts the choice of how to reach the target host into the hands of the routing table (and hence OSPF), allowing it to optimize the routes to reach the target host, such as by using hipersockets within the same physical system or an OSA Express Gigabit Ethernet between the physical systems instead of the XCF links.

When a connection from the client needs to be processed by the Sysplex Distributor, it will determine whether a matching VIPAROUTE statement has been specified. If it has, the best available route will be determined using the normal IP routing tables. If no matching VIPAROUTE statement exists for that target, IP packets distributed by the Sysplex Distributor to that target will use the Dynamic XCF interfaces.

Note that some workloads are still routed via XCF only:

- Sysplex Wide Security Association (IPSec) packets (that is, SA negotiation packets intended for a target member)

  Note, however, that it is expected that SWSA is in effect, implying that only the distributing stack is involved in SA negotiation. SWSA is further explained in 7.3.1, “Sysplex-wide security associations” on page 185.
- Multi Level Security (MLS) tagged packets
- Policy Agent QoS performance data collection

7.1.2 TCP/IP stacks participation in a Sysplex configuration

Each z/OS V1R7 TCP/IP stack in the Sysplex when brought up:

- Joins the EZBTCPCS Sysplex group
- Exchanges IP address information with other active stacks
- Sets up dynamic connectivity with other stacks:
  - Using Dynamic IUTSAMEH, if the partner stack is in the same z/OS instance
  - Using HiperSockets™, if the partner stack is on the same physical system and uses the same CHPID to connect to the hipersocket
  - Using Dynamic XCF Connectivity (that is, IP over XCF links)
- Can coordinate DVIPA movement
- Can be a Sysplex Distributor target

  Processing of VIPADISTRIBUTE statements is controlled by the DYNAMICXCF parameter on the IPCONFIG.
- Can access SWSA Coupling Facility structure

  Sysplex Wide Security Association is explained at 7.3.1, “Sysplex-wide security associations” on page 185.

Important: The activation of a stack with the DYNAMICXCF statement in the PROFILE.TCPIP data set automatically creates a point-to-multipoint connection with other active TCP/IP stacks that also have a DYNAMICXCF statement. In some cases you might have to consider this as a potential security exposure, as it may implicitly open undesirable routes.

As of the writing of this book, the announcement letter for z/OS V1R8 mentions the capability of specifying *subplexes* within a sysplex.

A subplex is a subset of VTAM® nodes or TCP stacks in the Sysplex. All the members of the subset can establish dynamic connectivity through the Sysplex with other members of the subset. Members of the subset cannot establish dynamic connectivity through the Sysplex with non-members of the subset.

The Sysplex subplexing at z/OS V1R8 will solve the DYNAMICXCF security exposure by restricting dynamic connectivity to stacks within the same subplex.

7.2 IP filtering considerations

The Sysplex distributor implementation requires TCP/IP connectivity between the involved Sysplex members. Look at Figure 7-2 on page 183. It describes a scenario where we are using for IP communications in the Sysplex both the XCF interface, the hipersockets interfaces, and the OSA cards. They can be used to communicate with the Sysplex distributed applications or with other non-distributed applications.

In these Sysplex configurations all of the members are running in the same physical system and the OSA card is being shared by all partitions.

Figure 7-2 shows Sysplex TCP/IP IP filtering network interfaces.

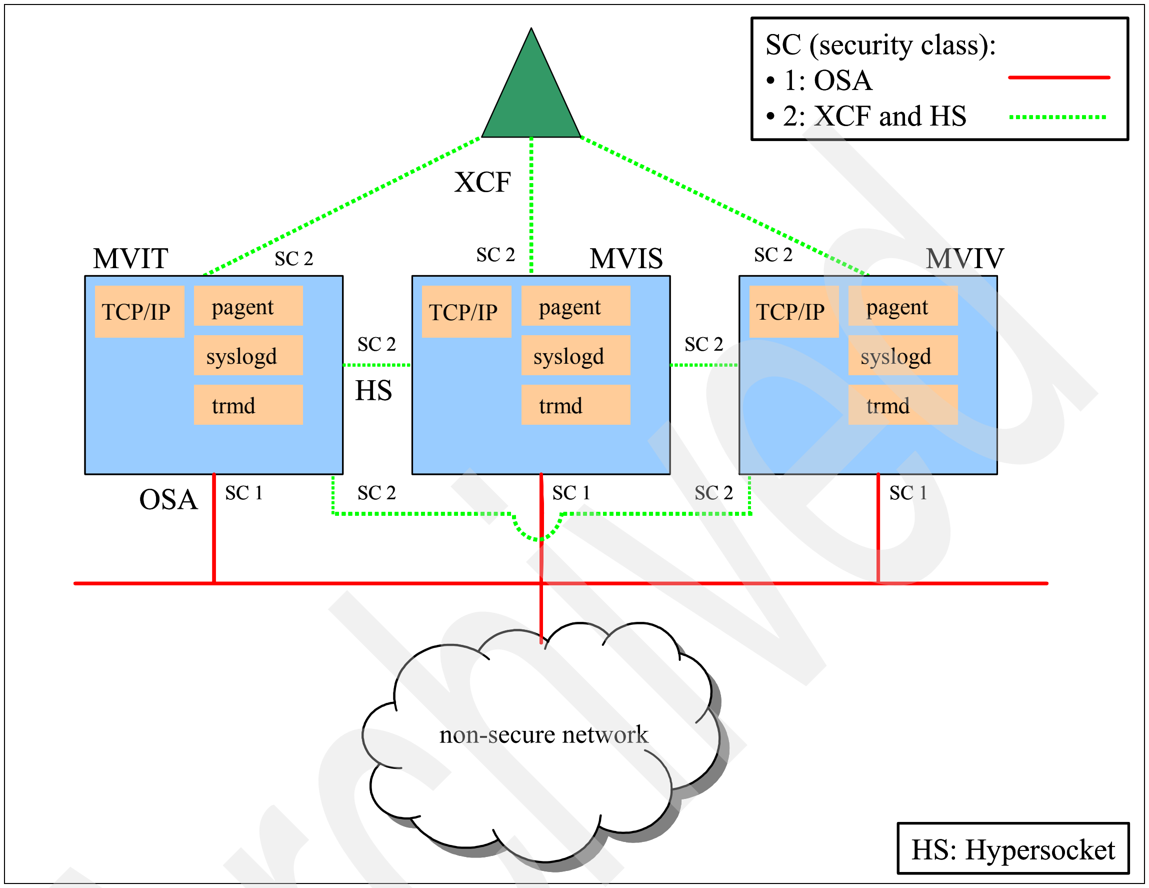


Figure 7-2 Sysplex TCP/IP IP filtering network interfaces

We have implemented two different security classes in our IP filtering scenario:

- The security class 1 is used for the interfaces connected to the non-secure network (that is, the OSA card).
- The security class 2 is used only for the interfaces connecting the members of the sysplex (that is, the XCF and hipersockets interfaces).

The security class is defined in the LINK statement for the device definition and in the DYNAMICXCF statement for the XCF links in the TCP/IP profile. Remember that the Security Class is an arbitrary number used to relate to each other LINK statements and filter rules. It is used to define groups of network interfaces that require the same set of IP filtering rules.

This is an approach that could be implemented to differentiate the security environment between the non-secure network and the intra-Sysplex network. Security classes are mapped against interfaces and not IP addresses. A good example of the use of Security classes is the setup of filter rules to detect packets with spoofed origin addresses. The rules are linked by the security class number to physical interfaces and are defined in such a way that any inbound packet that has a source address of an internal network node, but entered the system though an external interface, is probably spoofed and should be denied.

A more restrictive scenario here could also be to contemplate isolating from the filtering rules standpoint the hipersockets interface from the XCF interfaces by creating another security class.

The Sysplex distributor image will have to route the packets to the target members. Filtering rules have to be created at the targets to permit the packets with a destination address that is the distributed DVIPA to be accepted. The rules at the target must also allow outbound packets with a destination address that is the address of the initial client.

In z/OS V1R7, the VIPAROUTE statement in the PROFILE.TCPIP dataset of the distributing stack, and of its backup stack, is used to select a route from the distributing stack to the target stack that does not go through the dynamic XCF *adapters*. When a VIPAROUTE statement is in effect, packets are sent from the distributor stack to the target stack encapsulated as per the Generic Routing Encapsulation (GRE) RFC. The outer IP header will contain the VIPAROUTE target IP address as its destination IP address and the distributor's dynamic XCF address as the source IP address.

Generic Routing Encapsulation is a standard protocol described by RFC 1701. The receiving stack, which supports GRE, removes the GRE wrapper, allowing the retrieval of the original packet and proceeds with the transmission of the original packet.

In the target stack the encapsulated packet will be presented to the IP filtering layer as a normal packet after being decapsulated. The consequence of this implementation is that the filtering rules in the target remain the same whether the distributed packet is received over an XCF link or if it is transmitted via another path because of the VIPAROUTE statement.

Refer to Appendix C, “IP filtering implementation and management example” on page 221, for an IP filtering implementation example in a Sysplex configuration.

7.3 VPN considerations

In some environments it might be required to establish a VPN between the client (or a VPN gateway stack) and a sysplex distributed DVIPA. As explained in 6.5.2, “Security Associations (SA) and VPN” on page 161, the VPN is established after a Security Association has been created and installed by the IKE server at both ends of the tunnel. In the case of the sysplex distributor, however, if the intended tunnel end is the distributed DVIPA, the Security Association would normally have to be re-initiated on each target stack, as the distributor stack would dispatch the workload for this DVIPA to any one of these target stacks.

7.3.1 Sysplex-wide security associations

With the Sysplex Wide Security Association (SWSA) facility the negotiated Security Association for a tunnel ending at the distributed DVIPA is made sharable among the IKE servers residing on the target stacks. SWSA requires the use of a coupling facility structure, the EZBDVIPA list structure, that keeps information related to the Security Association that any target IKE can use.

All Security Association negotiations (that is, the Security Association (SA) creation and refreshes) are performed solely by the distributing stack. Once an SA is established by the distributing stack, shadow copies of the SA are sent to all target images. Traffic flowing through the tunnel passes through the distributing host un-encrypted. The shadow copy of the SA is used at the target stack to determine proper processing for the IPSec packet data to be decrypted or encrypted. Even if the traffic from the distributing host to a target host flows outside the sysplex, as a consequence of the VIPAROUTE specification, the IPSec traffic remains encrypted until it reaches the target.

The EZBDVIPA structure is used to keep track of the outbound sequence numbers. All systems using the same distributed SA obtain sequence numbers from the coupling facility. The information necessary to renegotiate the SA is also held in the EZBDVIPA structure. If a backup distributing stack takes over it can read this information and sends it to the IKE server to renegotiate the tunnels for this DVIPA.

The path taken by a VPN encrypted packet can be viewed graphically in Figure 7-3. Encryption and decryption occur only at the target image, LPAR3.

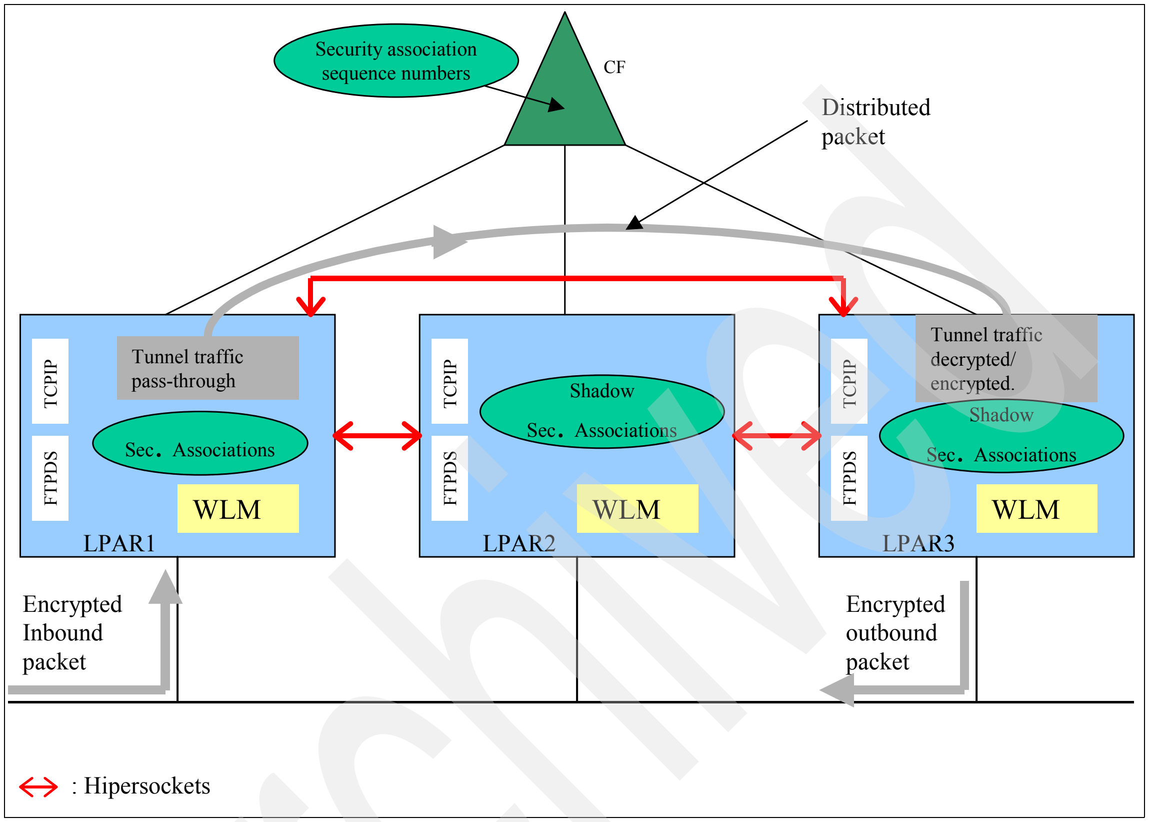


Figure 7-3 Sysplex-wide security associations

7.4 System SSL and AT-TLS considerations

If the Sysplex Distributor target members are hosting the same application that falls under an AT-TLS policy, then a common AT-TLS policy can be shared between the target stacks.

In a Sysplex scenario the sysplex session ID caching is an option to be exploited to save CPU cycles. Sysplex session ID caching has to be configured in the AT-TLS policy to be used.

7.5 Sysplex session ID caching

The Transaction Layer Security (TLS) client/server protocol has been designed to provide security for TCP/IP sockets communications (that is, confidentiality and integrity of the data flowing over the network and authentication of both the server and the client (the latter being optional)). TLS is used for TCP communications only.

The current TLS Version 1.0 protocol standard is delivered by IETF and is based on the Netscape Secure Socket Layer (SSL) Version 3.0 protocol (the latest version of SSL, which is now superseded by TLS).

The TLS protocol implements two distinct phases:

- The handshake protocol
- The record protocol

The record protocol provides integrity and privacy by using symmetric encryption and hashing algorithms for the transported data.

The handshake protocol is used by the client and server at the TLS communication initiation to:

- Authenticate the server, and, optionally, the client.
- Negotiate what symmetric encryption and hashing algorithms will be used by the record protocol, and securely exchange random secret keys that will be used in this phase.

The usage of encryption is CPU intensive by nature. But the TLS handshake protocol is known to be very demanding on computing resources because of its use of asymmetric, or *public key*, cryptography (the record protocol being less resource consuming because of its use of symmetric, or *shared secret key*, encryption).

New symmetric keys are needed, in theory, every time a TCP session is established between the client and the server (that is, the handshake has to be performed every time a new connection starts (which can be very often for protocols that, as http does, drop the TCP connection after a rather small bursts of data have been exchanged)).

To help reduce the overhead of the handshake TLS offers the option to shorten the handshake protocol by re-using the symmetric keys that were already negotiated during a previous TCP session between the same client and server. Every time that symmetric keys are created (that is, during a full handshake) the server produces a session ID number that can be used by the client, at the next connection, to recall the symmetric keys that have already been generated and exchanged in a previous connection. Both client and server keep in their storage the symmetric keys that correspond to previous session ID, for a duration that can be adjusted. The shorter the period while the session ID is remembered the more secure the communication is, as new random symmetric keys will be established more often, however, at the cost of frequent full handshakes. The longer the session ID is being remembered the less secure the communication is, as the same symmetric keys will be used for a longer period of time. However, there will be more partial handshakes, because of the re-use of the already existing symmetric keys material, and therefore less computing resources consumed.

In practice, when the client connects it sends the previous session ID, if any, to the server which, in turn, will recognize what previous session keys to re-use if it still remembers them.

The session ID function is natively implemented by the TLS protocol. There is no customization, except for setting up how long the server remembers a session ID (that is, the session-ID time-out). Setting the session-ID time-out to zero results in a full handshake to be performed at every connection of the client to the server.

The TLS protocol is described in RFC 2246. Figure 7-4 shows how the full TLS handshake looks.

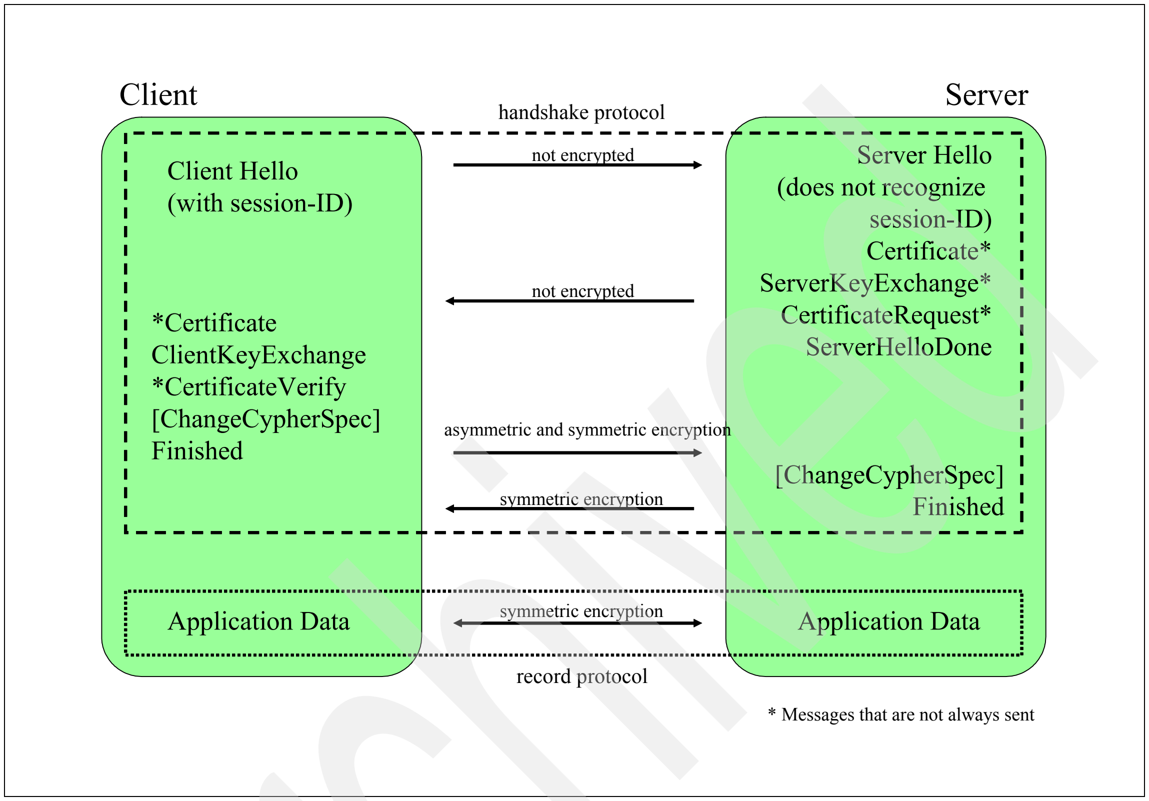


Figure 7-4 Full TLS handshake

In the abbreviated TLS handshake the asymmetric encryption is not necessary. The data will be sent using the symmetric keys negotiated previously in the last full handshake. See Figure 7-5. You can see that there is no more asymmetric encryption involved in the TLS handshake phase.

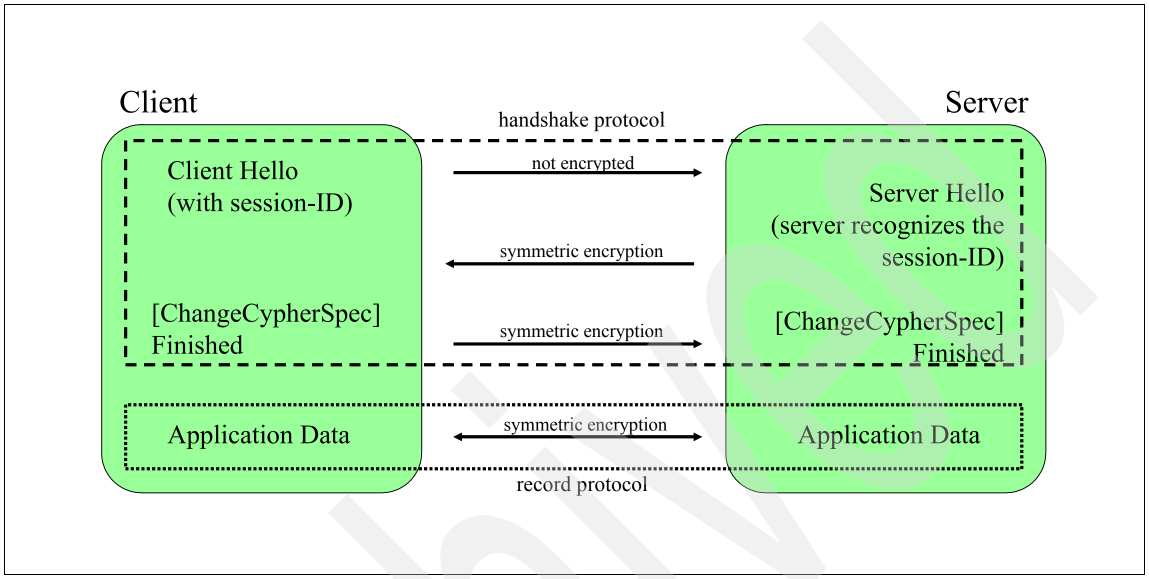


Figure 7-5 Abbreviated TLS handshake

Session-ID caching is a quite common practice to reduce the cryptographic workload overhead of generating and exchanging new symmetric keys again. However, it is based on the capability of the TLS server to keep in a cache the previous symmetric keys indexed with their corresponding session IDs. When TLS-protected requests are being distributed in a sysplex, the benefit of session-ID caching is lost, in that each TLS-enabled target member has its own local session ID cache that is not shared with other members. A full handshake must be performed again if a new connection is established with a target server that was not involved in the previous connections.

That is where the sysplex session-ID caching capability of z/OS System SSL comes in to play. The full description on how to set up the System SSL started task and configure the Sysplex session-ID caching is available in the z/OS System SSL document *z/OS Cryptographic Services System Security Layer Programming*. Here we provide a brief description and traces to exemplify how and when session-ID caching is being used across members in a Sysplex.

7.5.1 z/OS Sysplex session-ID caching support

Session ID caching is the main contributor in enhancing TLS performance by decreasing the frequency of full handshakes. With the z/OS Sysplex session-ID caching, the session-IDs can be shared among TLS-enabled members in the Sysplex under control of the GSKSRVR STC started in each sysplex member that is to share a session-ID with other members.

The local instance of GSKSRVR handles the handshake and keeps the session ID in a data space. When a GSKSRVR instance receives a non-locally known session ID from a client it enquires through XCF signalling whether this session ID is known from other GSKSRVRs in the sysplex. The inter GSKSRVR communications are performed via XCF. If the session ID is known from another GSKSRVR, this GSKSRVR instance sends the corresponding pre-master secret (the root key to derive the symmetric keys used by the session), still using XCF, to the requesting instance. This is graphically shown in Figure 7-6.

Note that the pre-master secret value is transmitted in clear as an intra-sysplex environment is considered as secure.

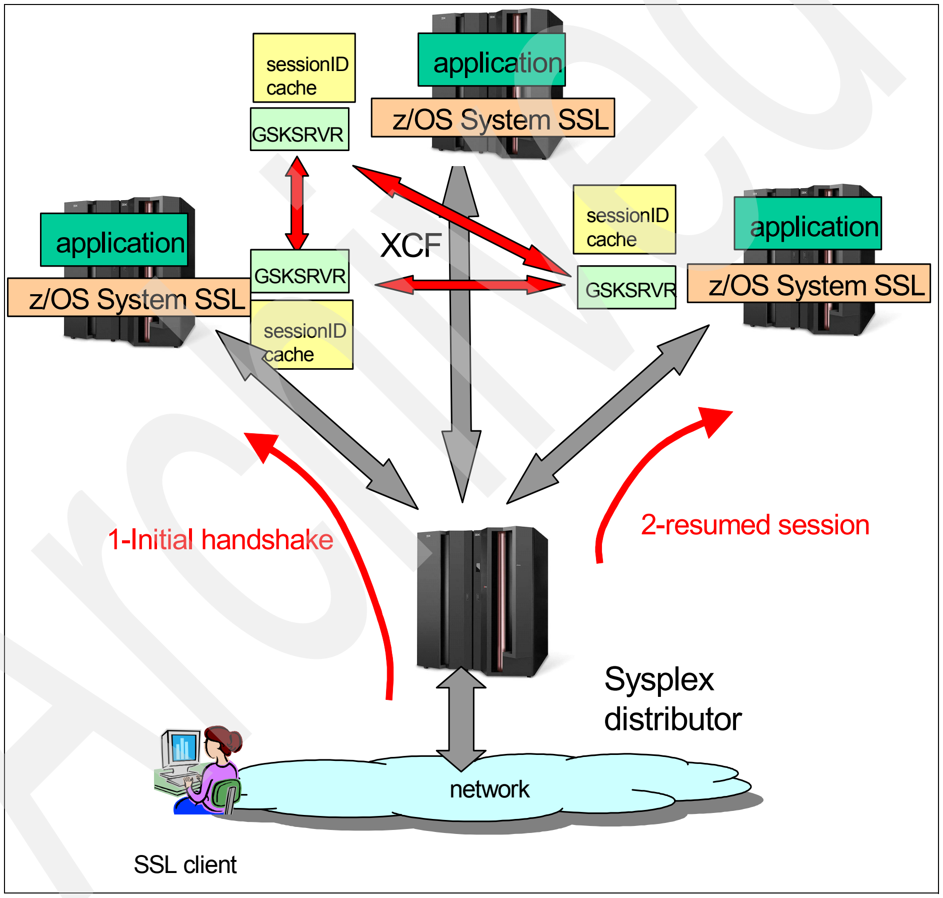


Figure 7-6 Sysplex Session ID caching

No additional structure has to be defined in the coupling facility to use this function. XCF messages are used for communication between System SSL servers using the group GSKSRVGP.

Note: Sysplex session-ID caching only supports SSL V3 and TLS V1 session information. SSL V2 session information is stored in the SSL server local cache and is therefore not sharable with other members of the Sysplex. Note, however, that SSL V2 should only be used for compatibility purposes. If not required, the usage of SSL V2 should be disabled when possible. SSL V3 and TLS V1 are more robust and secure protocols than SSL V2.

Unlike for the normal local session-ID cache implementation, the TLS-enabled applications must be customized for use of the Sysplex session-ID caching. This can be done using an environment variable called GSK_SYSPLEX_SIDCACHE. Alternatively, programs can be written to use the gsk_attribute_set_enum() call to set the environment variable internally.

Applications using session ID caching

As noted before, if an application has not been written to explicitly enable sysplex session-ID caching, then an environment variable may be used. This method, however, might not work with some applications. For example, the z/OS TN3270 server does not support the GSK_SYSPLEX_SIDCACHE environment. If Sysplex session-ID caching is really desired (although in this example it might have a small impact, as TN3270 connections last for a long time with very few handshakes), then AT-TLS can be used with the TN3270 server being configured as not using SSL/TLS.

Another thing to consider about TN3270 or any other application is that the client must support the session-ID caching by sending the session-ID with the client Hello message. The only IBM TN3270 client that supports this TLS function is Host On Demand.

Note: Refer to Appendix E, “AT-TLS implementation” on page 253, for an example of AT-TLS setup and session-ID exploitation.

7.6 External load balancing security considerations

External load balancer devices can also be used to dispatch work to Sysplex members. From the security perspective any sysplex member would then be treated as a single sysplex member connected to a non-secure network, as there will not be intra-sysplex routes involved in the workload dispatching.

In a mixed configuration where the Sysplex distributor is used with an external load balancer one must consider both sets of protections at each target member (that is, consider the security of the sysplex members as being individually connected to non secure networks and also consider that security has to also be implemented at the intra-sysplex TCP/IP routes).

Some load balancer devices interact with the Sysplex WLM to extract information to be used to make educated decisions about which member to route the request to. This brings two additional considerations: the WLM information flows to the load balancer as another TCP/IP connection, and some workload balancer implementations use the GRE protocol to encapsulate the traffic from the load balancer to a VIPA address. If IP filtering rules, for example, are being implemented, the filter policy should be defined to support the relevant specific connections.

Miscellaneous network-related considerations

After getting acquainted with the TCP/IP miscellaneous techniques and protections mechanisms available in z/OS, we now address more general considerations on network topologies that may influence sysplex configurations.

We also mention other security exposures that may indirectly result from TCP/IP threats.

8.1 The DeMilitarized zone (DMZ)

The demilitarized zone is a term often used when describing firewall configurations. Figure 8-1 shows a typical example. A DMZ is an isolated subnet between your secure network and the Internet. Anyone can enter it, but the only things present are those that you wish to allow access to anyway. The demilitarized zone is an area where you place Web servers and other servers that you want to make available for public access, but that you also wish to protect to some degree.

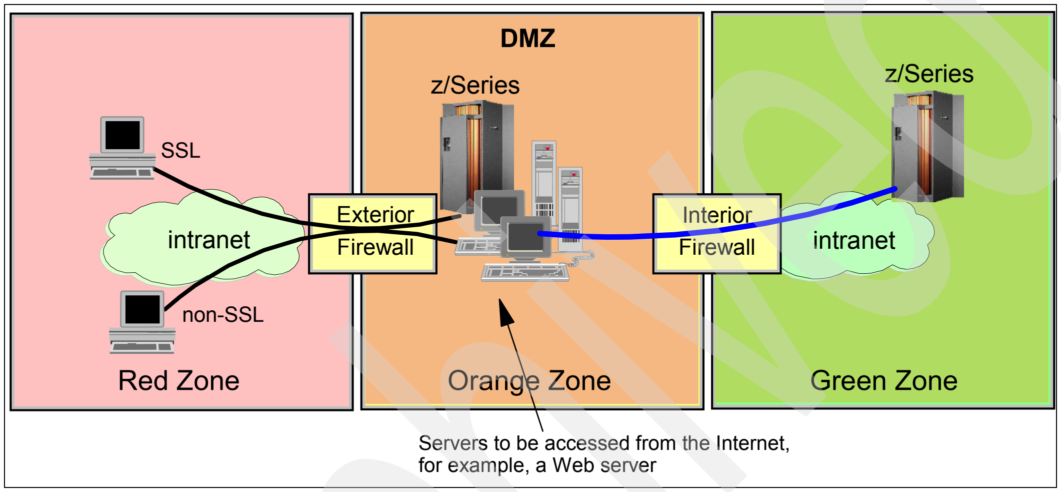


Figure 8-1 The DMZ principle

This is achieved by placing an outer firewall (often a packet filtering router) between the Internet and the servers in the DMZ, and another firewall (often an application-level gateway, such as a proxy server) between your secure network and the DMZ. The outer firewall is designed to allow into the DMZ only those requests that you wish to receive at your Web servers, but could also be configured to block denial-of-service attacks and to perform network address translation of the servers in your DMZ. The inner firewall is designed to prevent unauthorized access to your secure network from the DMZ and also, perhaps, to prevent unauthorized access from your secure network to the DMZ or the connected non-secure network. When you put a z/OS server into a DMZ, we strongly suggest that you use z/OS IP Security IP filtering. You should use z/OS IP filtering to block all traffic into and out of your z/OS server that does not belong to the services you are going to offer from this server. This control should be in place even if you already have a filtering router or firewall between the non-secure network and this server.

8.1.1 The z/OS TCP/IP stack in the DMZ

DMZ z/OS stack security can be enforced using several z/OS features, as shown in Figure 8-2, where:

- z/OS IP filtering is used as an additional protection on the outer side and as the protection mechanism on the inner side.
- The z/OS IDS is activated and monitors inbound packets.

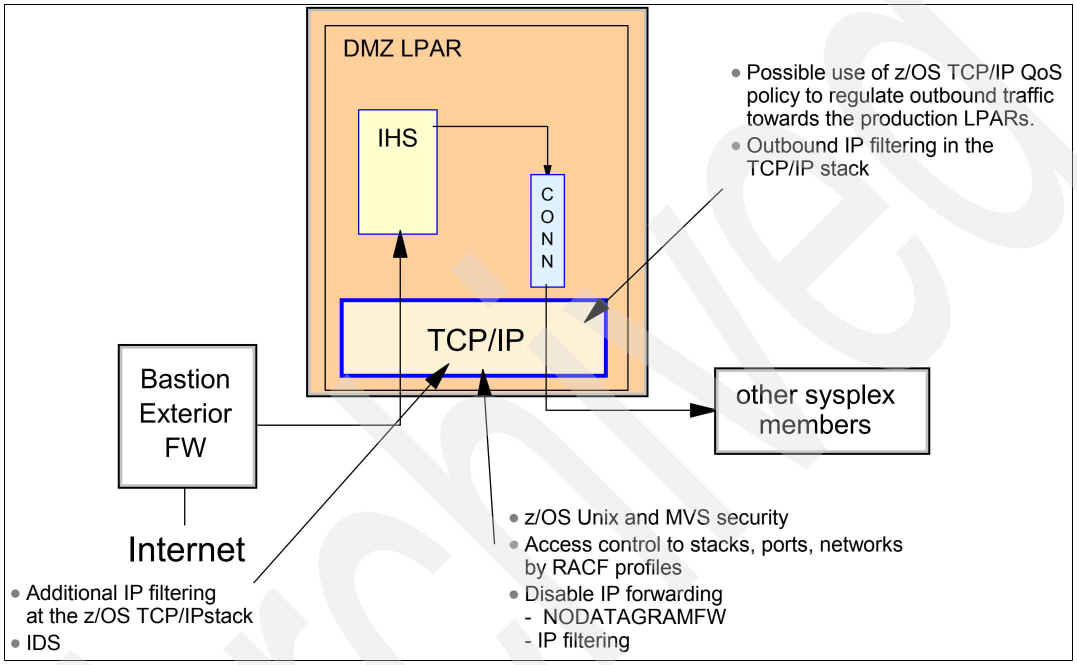

DMZ LPAR
IHS
CONN
TCP/IP
Possible use of z/OS TCP/IP QoS policy to regulate outbound traffic towards the production LPARs.
Outbound IP filtering in the TCP/IP stack
Bastion Exterior FW
other sysplex members
Internet
Additional IP filtering at the z/OS TCP/IPstack
IDS
z/OS Unix and MVS security
Access control to stacks, ports, networks by RACF profiles
Disable IP forwarding
- NODATAGRAMFW
- IP filtering

Figure 8-2 Enforcing z/OS DMZ stack security

Additional notes

We recommend that IP forwarding be disabled so that all communications with the rest of the sysplex have to go through, in this example, the IBM HTTP server and the connector, thus enforcing the logical insulation between the Internet and the secure area network. This can be achieved by either of the following:

- Coding IPCONFIG NODATAGRAMFWD in the PROFILE.TCPIP dataset
- Setting up z/OS IP filtering to reject packets that would need to be rerouted by the stack

Another possible exposure is the potential externalization by the DMZ stack of dynamic routing information pertaining to the secure area network. This can be minimized by using static routing to the internal network.

Attention: For a stack to be a distributing stack it takes having DATAGRAMFWD in the PROFILE.TCPIP dataset. This is discussed in 8.2.3, "Words of caution - Sysplex Distributor" on page 204.

8.1.2 The dual-stack configuration

This section describes a possible implementation that increases the security of a DMZ configuration by dedicating a stack to the outer connection and another one to the inner connection. It is not always possible to implement this configuration due to the requirement to establish stack affinity (explained in 8.1.3, "Establishing a stack affinity" on page 199), and also to the possible complexity of the IP naming plan that could result from this configuration. However, we think it could be of some interest for installations seeking maximum security since:

- A configuration mistake, or a code compromise, in the outer stack or the inner stack, cannot result in a bypass between the non-secure network and the secure network.
- The routing daemon activated in the outer stack cannot know, and therefore advertise, the secure network routes.
- The secure and non-secure traffic is separated and isolated from mutual interferences that might occur during peak workload periods or a denial-of-service attack.

Note: IHS to and from connector communications are cross-memory exchanges, and there is obviously a need to establish an affinity between IHS and the outer stack and between the connector and the inner stack.

Figure 8-3 shows a dual-stack configuration.

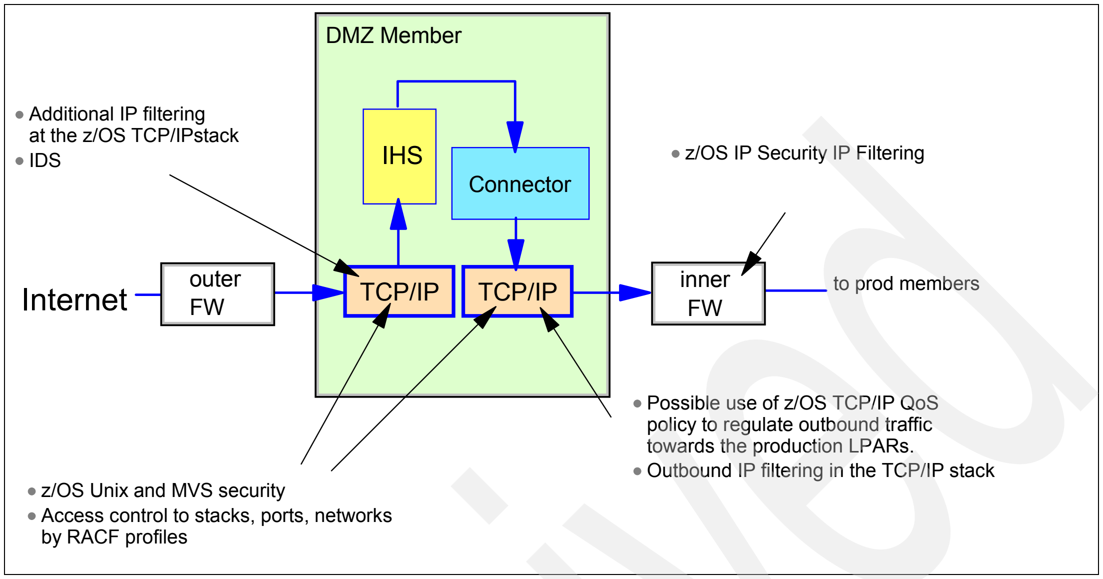

DMZ Member
• Additional IP filtering at the z/OS TCP/IPstack
• IDS
IHS
Connector
• z/OS IP Security IP Filtering
Internet
outer FW
TCP/IP
TCP/IP
inner FW
to prod members
• Possible use of z/OS TCP/IP QoS policy to regulate outbound traffic towards the production LPARs.
• Outbound IP filtering in the TCP/IP stack
• z/OS Unix and MVS security
• Access control to stacks, ports, networks by RACF profiles

Figure 8-3 Dual-stack configuration

8.1.3 Establishing a stack affinity

In order to support more than one TCP/IP stack, the Physical File System must be configured to support Common INET (cinet). The cinet environment, however, has some subtle implications for certain IP applications. Some servers function as GENERIC applications. This means that when the server sets up a LISTEN (listen() socket call), this listen is effective across all active TCP/IP stacks within the MVS image. These are the servers we need to force for affinity to a stack, while other servers can be termed *specific*, in that they are intended to operate with one stack only.

There are two ways of establishing stack affinity with a generic application:

- Transport affinity can be set via the ENVAR environment variable _BPXK_SETIBMOPT_TRANSPORT for C language and POSIX-based applications. This variable should be set to the job name of the TCP/IP stack to which you want this instance of the application to bind. Note that the stack job name is the name indicated in the SUBFILESSYSTEM definition and as the STC job name. Using _BPXK_SETIBMOPT_TRANSPORT causes the LE runtime environment to issue a setibmopt() call on behalf of the executing program environment variable.

Examples of applications enforcing this environment variable are the z/OS IHS, FTP server, and LDAP server.

- For other programs, meaning programs not exploiting an envvar file, the BPXTCAFF program (as a job step) can be executed to set transport affinity for an entire address space. For more details on BPXTCAFF (and other transport options), see *z/OS UNIX System Services Planning*, GA22-7800.

One test setup is illustrated in Figure 8-4. HTTP requests are coming from the non-secure network to the stack TCPIPOE, starting a servlet invoking JDBC™, the SQL requests being propagated by a DB2 DDF instance to a remote DB2 server over the secure network. The IHS was started with _BPXK_SETIBMOPT_TRANSPORT=TCPOE in its httpd.envvar file. The DB2 DDF instance was started with a step in its STC procedure invoking PGM=BPXTCAFF. Both TCPIPOE and TCPIPBE are declared in the BPXPRMxx member of PARMLIB, as shown in Example 8-1 on page 201, with the same name given to their STC procedure.

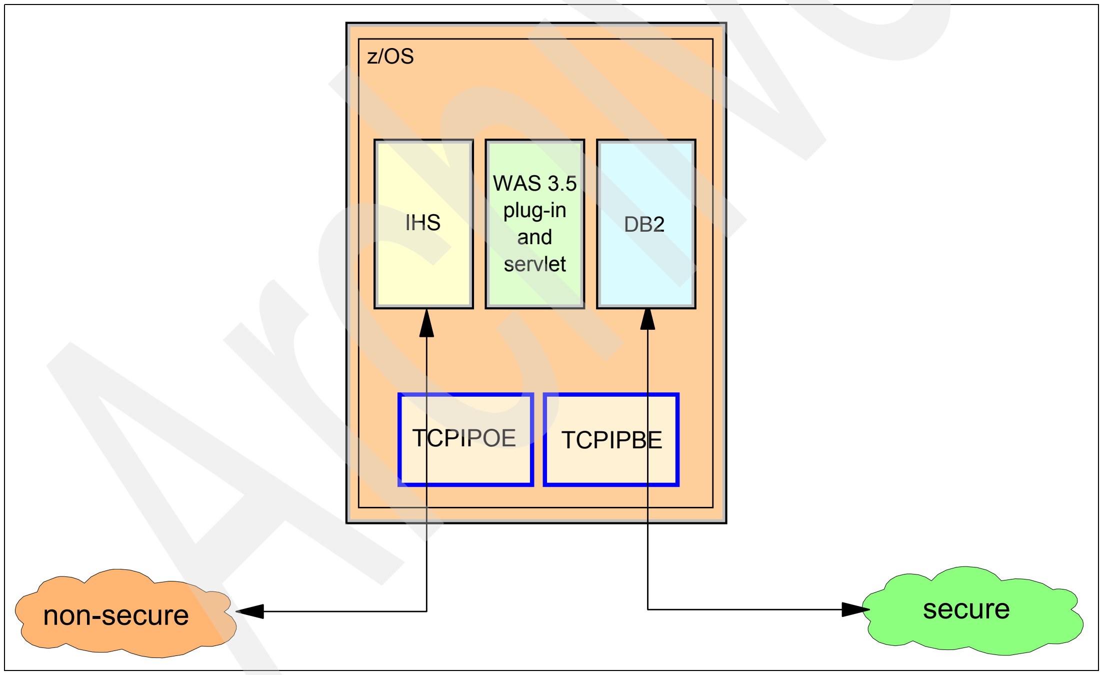


Figure 8-4 Our setup

Example 8-1 TCP/IP definitions in BPXPRMxx

```
/*******************************************************/
/*  Default TCPIP procedure                            */
/*******************************************************/
   SUBFILESYSTYPE NAME(TCPIPOE)
                  TYPE(CINET)
   ENTRYPOINT(EZBPFINI)
                  DEFAULT
  SUBFILESYSTYPE NAME(TCPIPBE)
TYPE(CINET)
```

8.2 Applicability of the DMZ principle to Parallel Sysplex

For the sake of simplicity, we assume here that there is only one member of the sysplex to be connected to the non-secure network. Application of the DMZ principle leads to the configuration shown in Figure 8-5, where the connected member constitutes the DMZ enclosed by a dedicated outer firewall box and, for instance, the z/OS IP filtering on the inner side.

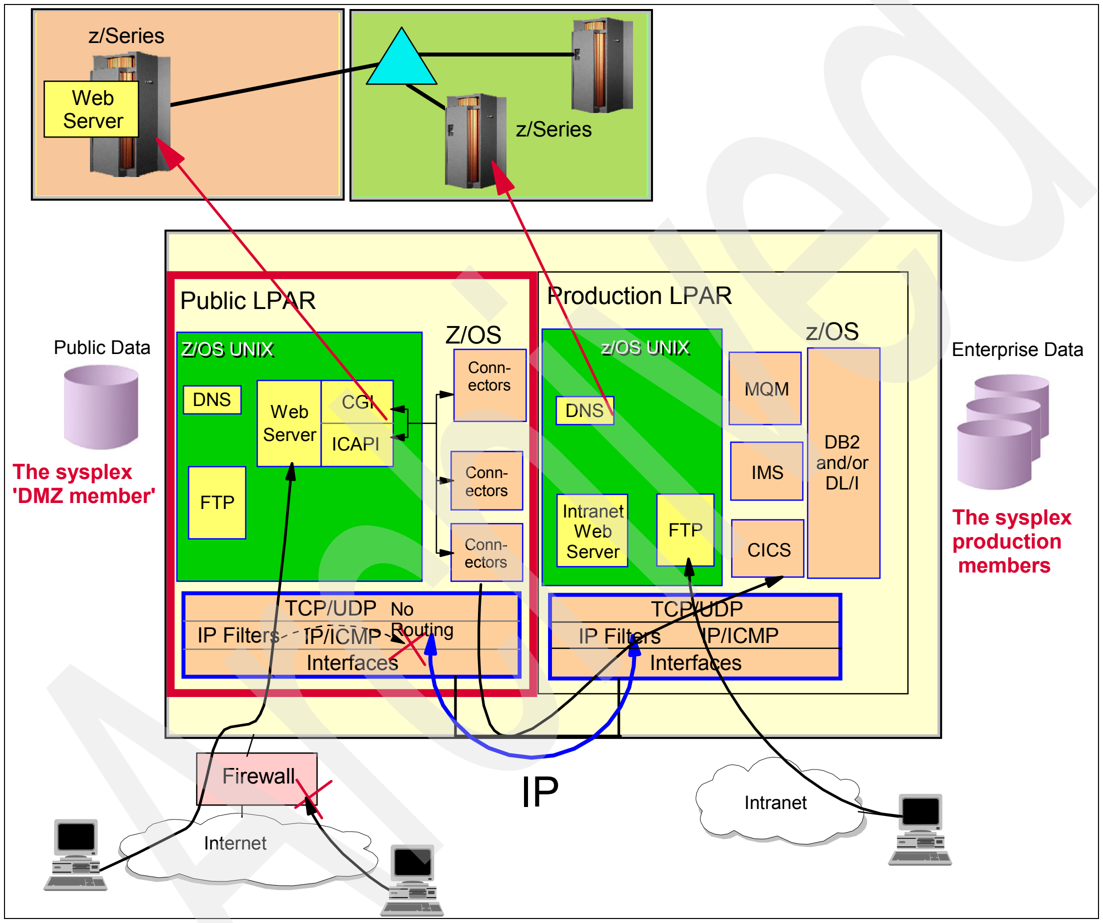


Figure 8-5 Applying the DMZ principle to the sysplex

8.2.1 Words of caution - XCF Messaging

As discussed previously, the intent of the DMZ is to screen all communications occurring between the non-secure network and the exposed servers, and between the secure zone and the same exposed servers. This principle cannot be fully applied in a sysplex configuration because there is no way to screen the XCF messaging itself. One can screen the IP communications occurring over XCF, but when it comes to other data being transported by XCF, such as console messages, there is no screening mechanism available. Therefore, strictly speaking (but we do not know how realistic the exposure is), one can imagine a denial of service condition being originated from the DMZ member that would be issuing, for some reason, a very high volume of XCF messages to the other members of the sysplex, eventually causing saturation of XCF traffic. We took as an example the console messages being broadcast to all sysplex members, whether or not they are actually to exploit or display the received message.

Again, not being able to assess the exposure, one may want to ensure maximum security by dedicating a separated sysplex for the DMZ, as shown in Figure 8-6. For this illustration we assume that the benefits of sysplex are required for the services provided from the DMZ.

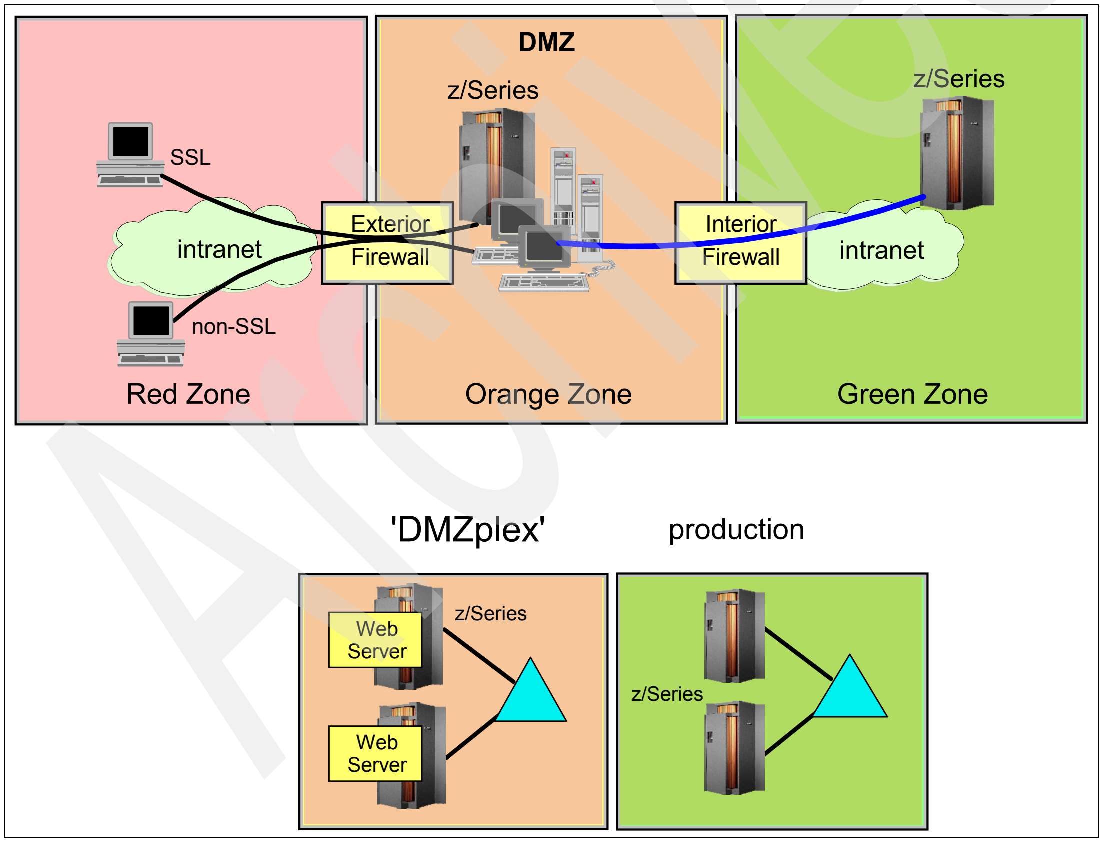


Figure 8-6 The DMZ-plex

8.2.2 Words of caution - RACF user revocation

Another issue when connecting to the Internet and when authenticating users by RACF user ID and password is the risk of intended user revocation by exceeding the authorized number of incorrect passwords. This is a form of denial of service attack that can be circumvented only by using SSL with user authentication by certificate.

One way to avoid exposing the production sysplex to Internet-initiated user revocation is to have one RACF database for the DMZ and a different RACF database for the production sysplex, possibly with an RRSF connection to manage user profile synchronization with:

- RRSF password synchronization with the prod-plex database
- One administrator SPECIAL and NOPASSWORD in the DMZ database, RRSF managed by one RACF administrator from the prod-plex

8.2.3 Words of caution - Sysplex Distributor

Important: As of the writing of this book, the announcement letter for z/OS V1R8 mentions the capability of specifying *subplexes* within a sysplex.

A subplex is a subset of VTAM nodes or TCP stacks in the Sysplex. All the members of the subset can establish dynamic connectivity through the Sysplex with other members of the subset. Members of the subset cannot establish dynamic connectivity through the Sysplex with non-members of the subset.

The Sysplex subplexing at z/OS V1R8 will solve the DYNAMICXCF security exposure that we are referring to here by restricting dynamic connectivity to stacks within the same subplex.

A few security-related items have to be considered when using the Sysplex Distributor:

- IP communications are carried by XCF, so there is no external firewall possible to filter these communications between the distributor and the target stacks. Filtering is only possible using the z/OS IP Security IP filtering.
- The Sysplex Distributor function requires DATAGRAMFWD at the distributor stack and DYNAMICXCF at all participating stacks to be active in the TCP/IP profile.

  DATAGRAMFWD permits the routing of packets at the stack level, which may become a security exposure in case of administration or filtering rule mistakes.

The consequence of having DYNAMICXCF on in a stack being brought up in the sysplex is that IP routes are automatically created with any stacks that are already up and that have DYNAMICXCF on. There is a risk here of ruining a DMZ setup if in the same sysplex you install has either of the following:

– A sysplex distributor with the distributor stack only in the DMZ (not a likely configuration, but possible). In that case the DMZ must be carefully closed on all possible IP routes that could be created with other stacks having the DYNAMICXCF on.

– More than one sysplex distributor, one in the DMZ and one for production (this is a more likely configuration). Again, as shown in Figure 8-7, all the unintended IP routes would have to be carefully closed by z/OS IP filtering. Although feasible, we deemed this configuration to be too error prone, especially when introducing new distributor's participating members. Instead, our recommendation is to install one single distributor per sysplex, leading again to having more than one sysplex, or alternatively to use external dynamic switches such as those shown in Figure 8-8 on page 207.

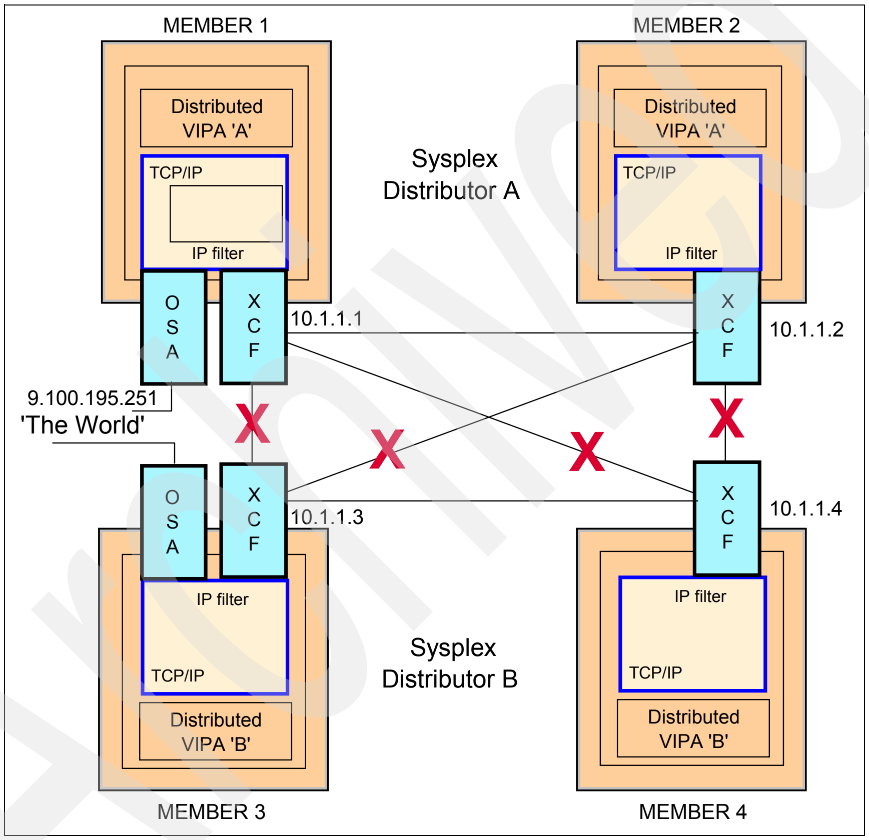


Figure 8-7 Closing IP routes created by DYNAMICXCF

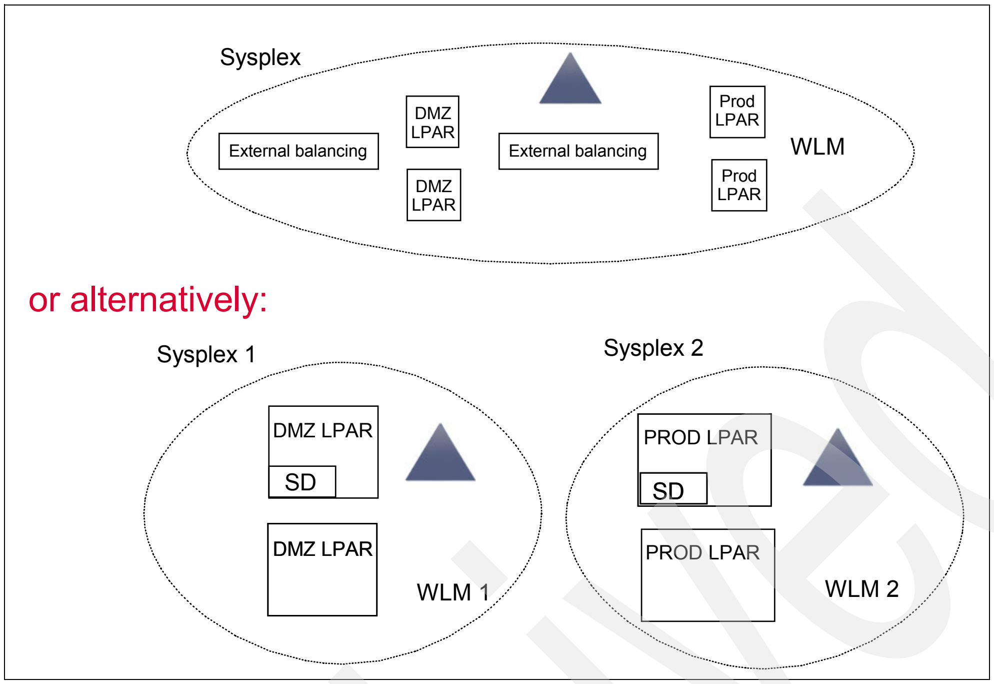


Figure 8-8 More than one Sysplex Distributor case

A

RACF protection of MVS commands

This appendix provides a lists of the MVS commands recommended to be protected by RACF.

Example: A-1 MVS commands, authorization levels, and associated RACF profiles

```
CANCEL device                       UPDATE      MVS.CANCEL.DEV.device
CANCEL jobname                      UPDATE      MVS.CANCEL.JOB.jobname
CANCEL jobname.id                   UPDATE      MVS.CANCEL.STC.mbrname.id
CANCEL id
CANCEL jobname                      UPDATE      MVS.CANCEL.STC.mbrname.jobname
CANCEL jobname                      UPDATE      MVS.CANCEL.ATX. jobname
CANCEL U=userid                     UPDATE      MVS.CANCEL.TSU.userid
CHNGDUMP                            UPDATE      MVS.CHNGDUMP
CMDS DISPLAY                        READ        MVS.CMDS.DISPLAY
CMDS SHOW                           READ        MVS.CMDS.SHOW
CMDS REMOVE                         CONTROL     MVS.CMDS.REMOVE
CMDS ABEND                          CONTROL     MVS.CMDS.ABEND
CONFIG                              CONTROL     MVS.CONFIG
CONTROL A                           READ        MVS.CONTROL.A
CONTROL C                           READ        MVS.CONTROL.C
CONTROL D                           READ        MVS.CONTROL.D
CONTROL E                           READ        MVS.CONTROL.E
CONTROL M                           CONTROL     MVS.CONTROL.M
CONTROL N                           READ        MVS.CONTROL.N
```

CONTROL Q	READ	MVS.CONTROL.Q
CONTROL S	READ	MVS.CONTROL.S
CONTROL T	READ	MVS.CONTROL.T
CONTROL V	READ	MVS.CONTROL.V
DEVSERV	READ	MVS.DEVSERV
DISPLAY A	READ	MVS.DISPLAY.JOB
DISPLAY APPC	READ	MVS.DISPLAY.APPC
DISPLAY ASM	READ	MVS.DISPLAY.ASM
DISPLAY ASCH	READ	MVS.DISPLAY.ASCH
DISPLAY ASM	READ	MVS.DISPLAY.ASM
DISPLAY CNGRP	READ	MVS.DISPLAY.CNGRP
DISPLAY CONSOLES	READ	MVS.DISPLAY.CONSOLES
DISPLAY DMN	READ	MVS.DISPLAY.DMN
DISPLAY DLF	READ	MVS.DISPLAY.DLF
DISPLAY DUMP	READ	MVS.DISPLAY.DUMP
DISPLAY EMCS	READ	MVS.DISPLAY.EMCS
DISPLAY ETR	READ	MVS.DISPLAY.ETR
DISPLAY GRS	READ	MVS.DISPLAY.GRS
DISPLAY IOS	READ	MVS.DISPLAY.IOS
DISPLAY IPLINFO	READ	MVS.DISPLAY.IPLINFO
DISPLAY JOBS	READ	MVS.DISPLAY.JOB
DISPLAY LOGREC	READ	MVS.DISPLAY.LOGREC
DISPLAY MMS	READ	MVS.DISPLAY.MMS
DISPLAY M	READ	MVS.DISPLAY.M
DISPLAY MPF	READ	MVS.DISPLAY.MPF
DISPLAY NET	READ	MVS.DISPLAY.NET
DISPLAY OPDATA	READ	MVS.DISPLAY.OPDATA
DISPLAY PARMLIB	READ	MVS.DISPLAY.PARMLIB
DISPLAY PFK	READ	MVS.DISPLAY.PFK
DISPLAY PROD	READ	MVS.DISPLAY.PROD
DISPLAY PROG	READ	MVS.DISPLAY.PROG
DISPLAY R	READ	MVS.DISPLAY.R
DISPLAY RTLS	READ	MVS.DISPLAY.RTLS
DISPLAY SLIP	READ	MVS.DISPLAY.SLIP
DISPLAY SMF	READ	MVS.DISPLAY.SMF
DISPLAY SMS	READ	MVS.DISPLAY.SMS
DISPLAY SSI	READ	MVS.DISPLAY.SSI
DISPLAY SYMBOLS	READ	MVS.DISPLAY.SYMBOLS
DISPLAY T	READ	MVS.DISPLAY.T
DISPLAY TP	READ	MVS.DISPLAY.TP
DISPLAY TRACE	READ	MVS.DISPLAY.TRACE
DISPLAY TS	READ	MVS.DISPLAY.TS
DISPLAY U	READ	MVS.DISPLAY.U
DISPLAY WLM	READ	MVS.DISPLAY.WLM
DISPLAY XCF	READ	MVS.DISPLAY.XCF
DUMP	CONTROL	MVS.DUMP
DUMPDS	UPDATE	MVS.DUMPDS
FORCE device	CONTROL	MVS.FORCE.DEV.device
FORCE jobname	CONTROL	MVS.FORCE.JOB.jobname

FORCE jobname.id	CONTROL	MVS.FORCE.STC.mbrname.id
FORCE id		
FORCE jobname	CONTROL	MVS.FORCE.STC.mbrname.jobname
FORCE U=userid	CONTROL	MVS.FORCE.TSU.userid
FORCE device,ARM	CONTROL	MVS.FORCEARM.DEV.device
FORCE jobname,ARM	CONTROL	MVS.FORCEARM.JOB.jobname
FORCE [jobname.]identifier,arm	CONTROL	MVS.FORCEARM.STC.mbrname.id
FORCE jobname,ARM	CONTROL	
MVS.FORCEARM.STC.mbrname.jobname		
FORCE U=userid,ARM	CONTROL	MVS.FORCEARM.TSU.userid
HALT EOD	UPDATE	MVS.HALT.EOD
HALT NET	UPDATE	MVS.HALT.NET
HALT TP	UPDATE	MVS.HALT.TCAM
HOLD	UPDATE	MVS.HOLD.TCAM
IOACTION	CONTROL	MVS.IOACTION
LIBRARY	UPDATE	MVS.LIBRARY
LOG	READ	MVS.LOG
MODE	UPDATE	MVS.MODE
MODIFY jobname	UPDATE	MVS.MODIFY.JOB.jobname
MODIFY userid	UPDATE	MVS.MODIFY.JOB.userid
MODIFY jobname	UPDATE	MVS.MODIFY.STC.mbrname.id
MODIFY jobname.id		
MODIFY id		
MODIFY jobname	UPDATE	MVS.MODIFY.STC.mbrname.jobname
MONITOR	READ	MVS.MONITOR
MOUNT	UPDATE	MVS.MOUNT
MSGRT	READ	MVS.MSGRT
PAGEADD	UPDATE	MVS.PAGEADD
PAGEDEL	UPDATE	MVS.PAGEDEL
QUIESCE	CONTROL	MVS.QUIESCE
RELEASE	UPDATE	MVS.RELEASE.TCAM
REPLY	READ	MVS.REPLY
RESET	UPDATE	MVS.RESET
RESET CN	CONTROL	MVS.RESET.CN
ROUTE system	READ	MVS.ROUTE.CMD.system
ROUTE *ALL	READ	MVS.ROUTE.CMD.ALLSYSTEMS
ROUTE *OTHER	READ	MVS.ROUTE.CMD.OTHERSYSTEMS
ROUTE sysgrpname	READ	MVS.ROUTE.CMD.sysgrpname
ROUTE (sys1,...,sysN)	READ	MVS.ROUTE.CMD.sys1
ROUTE (group1,...,groupN)	READ	MVS.ROUTE.CMD.group1
SEND	READ	MVS.SEND
SET APPC	UPDATE	MVS.SET.APPC
SET ASCH	UPDATE	MVS.SET.ASCH
SET CLOCK	UPDATE	MVS.SET.CLOCK
SET CNGRP	UPDATE	MVS.SET.CNGRP
SET DAE	UPDATE	MVS.SET.DAE
SET DATE	UPDATE	MVS.SET.DATE
SET GRSRNL	UPDATE	MVS.SET.GRSRNL
SET ICS	UPDATE	MVS.SET.ICS

SET IOS	UPDATE	MVS.SET.IOS
SET IPS	UPDATE	MVS.SET.IPS
SET MMS	UPDATE	MVS.SET.MMS
SET MPF	UPDATE	MVS.SET.MPF
SET OPT	UPDATE	MVS.SET.OPT
SET PKF	UPDATE	MVS.SET.PKF
SET PROG	UPDATE	MVS.SET.PROG
SET RESET	UPDATE	MVS.SET.TIMEDATE
SET RTLS	UPDATE	MVS.SET.RTLS
SET SCH	UPDATE	MVS.SET.SCH
SET SLIP	UPDATE	MVS.SET.SLIP
SET SMF	UPDATE	MVS.SET.SMF
SET SMS	UPDATE	MVS.SET.SMS
SETDMN	UPDATE	MVS.SETDMN.DMN
SETETR	UPDATE	MVS.SETETR.ETR
SETGRS MODE=STAR	UPDATE	MVS.SETGRS.MODE.STAR
SETIOS	UPDATE	MVS.SETIOS.IOS
SETLOAD	UPDATE	MVS.SETLOAD.LOAD
SETLOGIC	CONTROL	MVS.SETLOGIC.LOGRC
SETPROG	UPDATE	MVS.SETPROG
SETSMF	UPDATE	MVS.SETSMF.SMF
SETSMS	UPDATE	MVS.SETSMS.SMS
SETSSI ADD	CONTROL	MVS.SETSSI.ADD.subname
SETSSI ACTIVATE	CONTROL	MVS.SETSSI.ACTIVATE.subname
SETSSI DEACTIVATE	CONTROL	MVS.SETSSI.DEACTIVATE.subname
SETXCF	UPDATE	MVS.SETXCF.XCF
SLIP	UPDATE	MVS.SLIP
START mbrname[.identifier]	UPDATE	MVS.START.STC.mbrname[.id]
START mbrname,JOBNAME=jobname	UPDATE	MVS.START.STC.mbrname.jobname
STOP jobname	UPDATE	MVS.STOP.JOB.jobname
STOP userid	UPDATE	MVS.STOP.JOB.userid
STOP jobname	UPDATE	MVS.STOP.STC.mbrname.id
STOP jobname.id		
STOP id		
STOP jobname	UPDATE	MVS.STOP.STC.mbrname.jobname
STOPMN	READ	MVS.STOPMN
STOPTR	READ	MVS.STOPTR
SWAP	UPDATE	MVS.SWAP
SWITCH CN	CONTROL	MVS.SWITCH.CN.cnme1.cnme2
SWITCH SMF	UPDATE	MVS.SWITCH.SMF
TRACE CT	UPDATE	MVS.TRACE.CT
TRACE MT	CONTROL	MVS.TRACE.MT
TRACE ST	UPDATE	MVS.TRACE.ST
TRACE STATUS	UPDATE	MVS.TRACE.STATUS
TRACK	READ	MVS.TRACK
UNLOAD	UPDATE	MVS.UNLOAD
VARY CN	UPDATE	MVS.VARY.CN
VARY CN,ACTIVATE	READ	MVS.VARY.CN
VARY CN,AUTH	CONTROL	MVS.VARYAUTH.CN

VARY CN,DEACTIVATE	READ	MVS.VARY.CN
VARY CN,LOGON	CONTROL	MVS.VARYLOGON.CN
VARY CN,LU	CONTROL	MVS.VARYLU.CN
VARY CONSOLE	UPDATE	MVS.VARY.CONSOLE
VARY CONSOLE,AUTH	CONTROL	MVS.VARYAUTH.CONSOLE
VARY GRS	CONTROL	MVS.VARY.GRS
VARY HARDCPY	CONTROL	MVS.VARY.HARDCPY
VARY MSTCONS	CONTROL	MVS.VARY.MSTCONS
VARY NET	UPDATE	MVS.VARY.NET
VARY OFFLINE	UPDATE	MVS.VARY.DEV
VARY OFFLINE,FORCE	CONTROL	MVS.VARYFORCE.DEV
VARY ONLINE	UPDATE	MVS.VARY.DEV
VARY ONTP	UPDATE	MVS.VARY.TCAM
VARY OFFTP	UPDATE	MVS.VARY.TCAM
VARY PATH	UPDATE	MVS.VARY.PATH
VARY SMS	UPDATE	MVS.VARY.SMS
VARY WLM	CONTROL	MVS.VARY.WLM
VARY XCF	CONTROL	MVS.VARY.XCF
WRITELOG	READ	MVS.WRITELOG
Unrecognized CommandsREADMVS.UNKNOWN		

B

TCP/IP configuration information

In this appendix we provide details on the configuration of the following components of our TCP/IP configuration:

- The policy agent (PAGENT)
- The IKE daemon

Basic configuration of the policy agent

The policy agent can be started from the Unix System Services environment or it can also be started as a z/OS started task. In our case we chose to go with the started task option.

For readers from a non-z/OS background, the idea of using a started task to start a UNIX process may sound like an unnecessary complication. However, a started task has many advantages over a shell script, /etc/rc, or any other UNIX-based application management method. For example:

- Soft limits such a CPUTIME, ASSIZE, and MEMLIMIT can be imposed when executed from JCL.
- All non-UNIX tasks are managed from the z/OS console already.
- Most installations already have monitoring tools that use the z/OS system console log, as opposed to the z/OS UNIX shell.

A sample of the procedure used to start the policy agent can be found in Example B-1.

Example: B-1 Policy agent procedure

```
//PAGENT   PROC
//PAGENT   EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//*
//STDENV   DD DSN=SYSM.TCPCFG.PAGENT.ENV,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

The first thing to note in the policy agent started task is that STDENV is being used for controlling the PAGENT environment variables. The primary reason for this is that placing environment variables on an ENVAR parameter can quickly exceed the number of bytes allowed on a JCL PARM statement. Instead the STDENV DD statement points to a sequential z/OS data set.

Note: A file pointed to by _CEE_ENVFILE should be allocated as a VB data set. Although some parameters can still function in a non-VB data set, depending upon how the parser reads them, this can lead to unpredictable results.

The environment variables file that we used is shown in Example B-2.

Example: B-2 Standard environment file for PAGENT

```
PAGENT_LOG_FILE=SYSLOGD
PAGENT_CONFIG_FILE=//'SYSM.TCPCFG.TCPPARMS(PAGCON)'
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
```

More details

The system on which this policy agent is operating with CINET configured,

Note: CINET, which stand for Common INET, is a physical file system type that supports TCP/IP. An installation can choose between INET or CINET when configuring the BPXPRMxx parmlib member. If CINET is chosen, more than one instance of TCP/IP can be active at the same time in the MVS image.

At this stage, since there has been very little configured in the policy agent configuration file, the contents of SYSM.TCPCFG.TCPPARMS(PAGCON) consist of only one line:

```
TCPIMAGE TCPIP FLUSH PURGE 1800
```

Including a TCPIMAGE statement is to ensure that relevant policies are installed only in the specified stack (here the stack with a job name of TCPIP).

As we work our way through the setup of IP filtering, intrusion detection, VPN and AT-TLS configuration, the contents of the policy agent configuration file will grow significantly.

Sharing files

The sysplex used for this book used shared DASD. On each image, the TCP/IP stack had the same name: TCPIP. This made it easy to share all configuration files among the images. Only one PAGENT JCL file was needed, and one environment file and one resolver configuration file was used.

The only difference required was for the host name, as found in the resolver configuration file. For this the system symbolic for the &SYSCLONE value was used. See Example B-3 for the resolve configuration file.

Example: B-3 Shared resolver configuration file for all images

```
TCPIPJOBNAME  TCPIP
DOMAINORIGIN  PSSCMOP.IBM.COM
NSINTERADDR   127.0.0.0
HOSTNAME      DANU&SYSCLONE
```

A quick look around - pasearch and ipsec

Obviously, there is not much to see in terms of policy configuration with the aforementioned configuration. However, a command such as **`pasearch -c -p tcpip`** would give an idea of what defaults are there for the IDS, QoS, AT-TLS, and IPSec policies. The -c parameter tells **`pasearch`** to display policy object information, while the -p tcpip limits the output to the stack named TCPIP.

See Example B-4 for an output obtained with no policies defined yet.

Example: B-4 Default PAGENT policy

```
TCP/IP pasearch CS V1R7                 TCP/IP Image:      TCPIP
  Date:                 08/23/2005       Time:  07:32:00

Qos Policy Object:
  ApplyFlush:           True             PolicyFlush:       False
  ApplyPurge:           True             PurgePolicies:     False
  AtomicParse:          False            DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   False            ModifyOnIDChange:  True
  Configured:           False            UpdateInterval:    1800
  PerfColEnabled:       False

Ids Policy Object:
  ApplyFlush:           True             PolicyFlush:       False
  ApplyPurge:           True             PurgePolicies:     False
  AtomicParse:          False            DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   False            ModifyOnIDChange:  True
  Configured:           False            UpdateInterval:    1800

IPSec Policy Object:
  ApplyFlush:           False
  ApplyPurge:           False
  AtomicParse:          True             DeleteOnNoflush:   True
  DummyOnEmptyPolicy:   True             ModifyOnIDChange:  False
  IpSecEnabled:         True             UpdateInterval:    1800
  IpSec3DESEnabled:     True
  IpFilter Policy Object:
   Configured:          False
  KeyExchange Policy Object:
   Configured:          False
   AllowNat:            No                NatKeepAliveIntvl: 0
  LocalDynVpn Policy Object:
   Configured:          False

TTLS Policy Object:
  ApplyFlush:           True              PolicyFlush:       False
  ApplyPurge:           True              PurgePolicies:     False
  AtomicParse:          True              DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   True              ModifyOnIDChange:  False
  Configured:           False             UpdateInterval:    1800
```

It should be noted that policy information can also be displayed using the new **ipsec** command, which is available beginning with z/OS V1R7. The **ipsec** command can display policy information or it can also be used to change the behavior of IP filtering or VPN-related functions. More details on the use of the **ipsec** command can be found in Appendix G, “VPN setup” on page 269.

Important: Access to the **ipsec** command is controlled by the RACF profile EZB.IPSECCMD.sysname.tcpname.command_type in the SERVAUTH class of resources. If SERVAUTH is not active, or if the profile is not defined to SERVAUTH, the access to the command will be denied with the Communications Server message EZD0859I.

C

IP filtering implementation and management example

This appendix provides examples of:

- IP filtering implementation
- IP filtering management
- IP filtering implementation in a Sysplex Distributor context

z/OS IP filtering implementation

There are two ways of implementing the IP filtering policy:

- IP filtering *default policy* using the IPSEC statement in the TCP/IP profile dataset.
- IP filtering *filter policy* using the PAGENT policy configuration files. Those are flat files that can be created in a HFS file or in a sequential dataset under z/OS.

The TCP/IP profile has to be configured to activate the IP filtering implementation. There is a new option called IPSECURITY under the IPCONFIG statement that activates the IP filtering function on the TCP/IP stack.

If this option is not defined then the IP filtering function will not be available even if they are configured either in the PAGENT filter policy configuration file or in the IPSEC statement, that is the default policy, in the TCP/IP profile.

Figure C-1 shows the structure of the IP filtering implementation.

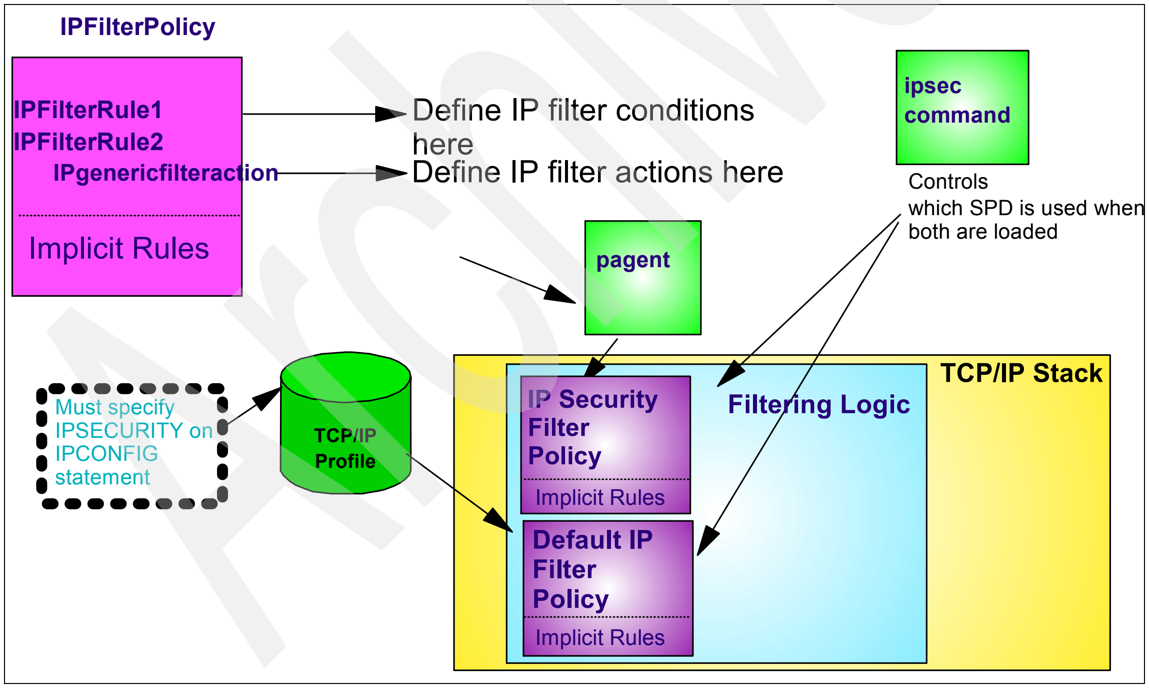


Figure C-1 IP security filtering structure

The structure is:

- The PAGENT loads the filter policy into the TCP/IP stack at PAGENT startup or when you make some changes to it and refresh the policy by using the **`modify`** console command.
- The default policy is always loaded into the stack at stack startup. Note that the implicit rules are always loaded even if there are no default rules specified. Any default rule specification can be changed by the **`vary tcpip obeyfile`** console command.
- The **`ipsec`** command is used to manage and monitor the IP security filtering.

Attention: Just specifying the IPSECURITY option in the IPCONFIG statement of the PROFILE.TCPIP dataset is enough to have the implicit rules to be created and to deny all the inbound and outbound TCP/IP traffic for this TCP/IP stack. Be careful when specifying this parameter.

The major differences between the default policy and the PAGENT filter policy are:

- In the default policy only permit rules can be defined. The implicit rules implement the deny all function. In the PAGENT filter policy you have the option to create deny rules.
- The default policy only applies to local packets. If packets are to be routed to other TCP/IP stacks, like in a sysplex distributor implementation, then the PAGENT filter policy has to be used.
- In the PAGENT filter policy there are options to group resources like IP addresses. This option is not available in the default policy.

The default policy will always be used in the absence of a PAGENT filter policy. If a PAGENT filter policy is defined, both will be loaded into the TCP/IP stack and the PAGENT filter policy will be used unless you specify by using the **`ipsec`** command that the default policy should be used.

By using the **`ipsec`** command it is possible to switch between the default and the PAGENT filter policy whenever it appears to be necessary. We discuss the usage of the **`ipsec`** command later in this chapter.

The IPSECURITY parameter is taken into account at the TCP/IP stack startup only. If you want to deactivate the IP filtering function you have to remove the IPSECURITY parameter in the IPCONFIG statement and recycle the stack.

Example C-1 shows the IP filtering definitions in the TCP/IP profile for the image MVIT.

Example: C-1 Integrated IP Security statements in the profile for image MVIT

```
IPCONFIG
   IPSECURITY                        1

; OSA 400
DEVICE ETH400 MPCIPA NONROUTER                                            2
LINK   ETH400 IPAQENET ETH400 READSTORAGE AVG INBPERF MINLATENCY SECCLASS 1

IPSEC        3
   LOGENABLE 4
   LOGIMPLICIT 5
ENDIPSEC
```

1 In the IPCONFIG statement there is an option to specify whether you want IPsecurity activated on the stack. This option automatically installs the default security policy that will contain only the implicit rules, meaning that all the IP traffic will be blocked. If the PAGENT is running and there is an IP filtering policy defined it will be installed and activated.

2 In the device definition statements there is a new option to classify the interfaces into security classes. When defining the IP filtering rules there is an option to specify on what security class the rule will be applied. The Security Class associated with a physical interface is expected to match the Security Class value of a filter rule. In this scenario we have only one security class for the non-secure network outside of the MVIT image, the security class 1. If we had more interfaces connected to that network for availability, which is recommended, and, if it is the case, scalability, they will belong to the same security class.

The VIPA devices, either static or dynamic, cannot be classified into security classes. Interfaces connected to the same network can be classified into different security classes. Look at Figure C-2 to see an example on how the interfaces can be classified into security classes.

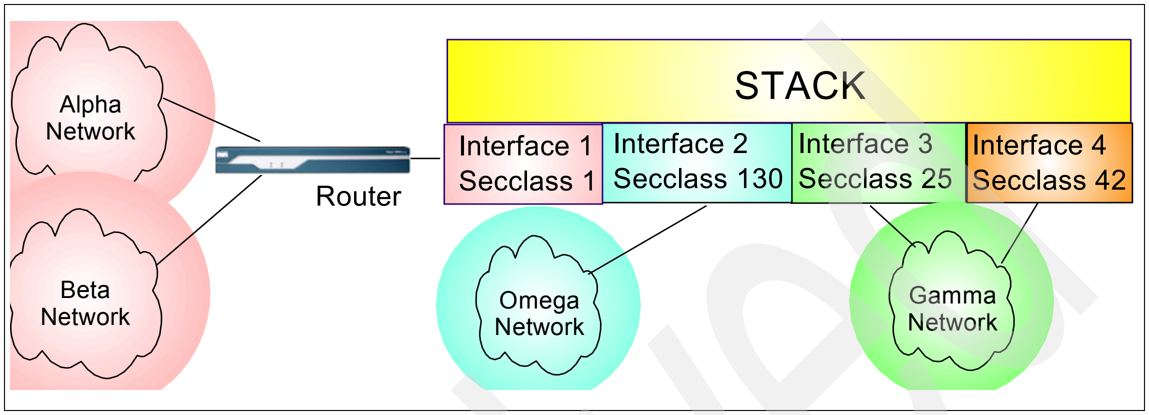


Figure C-2 Interface security class example

3 In this example the IPSEC statement does not specify any filter rules and therefore only the implicit rules are in use at stack startup. Additional rules have to be defined in order to get access to services running on this TCP/IP stack. In our scenario all the additional rules will be configured as a PAGENT filter policy. Filter rules can be defined in the IPSEC statement by using the IPSECRULE statement.

4 The LOGENABLE option activates the packet filter logging. All the log messages are sent to the syslogd by the TRMD daemon. You have the option to disable it by using the LOGDISABLE option. In each of the filter rules there is an option to generate a log record when the rule is applied to a packet.

5 The LOGIMPLICIT means that we want packets filter logging for packets denied by the implicit rules.

In our implementation we have all the configuration files for PAGENT, TRMD, SYSLOGD, and the profiles for the TCP/IP stacks installed in a PDS shared by all the members of the Sysplex, pointed at by the IpSecConfig statement in the PAGENT configuration file, as shown in Example C-2.

Example: C-2 PAGENT configuration file - IP security statements

```
IpsecConfig //'SYSM.TCPCFG.TCPPARMS(IPSECPO)'
```

This filter policy definition that we use in our scenario is shown in Example C-3.

Example: C-3 Filter policy definition for the MVIT in a single image scenario

```
IpFilterPolicy                      1
 {
   FilterLogging       On
   IpFilterLogImplicit Yes
   IpFilterGroupRef    MVI-All
 }
IpFilterGroup MVI-All               2
 {
   IpFilterRuleRef FTP235ControlLocal
   IpFilterRuleRef FTP235DataLocal
   IpFilterRuleRef InboundIcmp
   IpFilterRuleRef NologUdp
}
IpGenericFilterAction Permit-Log  3
 {
   IpFilterAction  Permit
   IpFilterLogging Yes
 }
IpGenericFilterAction Permit-NoLog
 {
   IpFilterAction  Permit
   IpFilterLogging No
 }
IpGenericFilterAction Deny-Log
 {
   IpFilterAction  Deny
   IpFilterLogging Yes
 }
IpGenericFilterAction Deny-NoLog
 {
   IpFilterAction  Deny
   IpFilterLogging No
 }
IpFilterRule FTP235ControlLocal                         4
 {
   IpSourceAddr       9.100.193.235
   IpDestAddr         9.100.199.0/24
   IpService
    {
      Protocol                Tcp
      SourcePortRange         21
      DestinationPortRange    1024 65535
      Direction               BiDirectional InboundConnect
      Routing                 Local
      SecurityClass           1
    }
   IpGenericFilterActionRef Permit-Log
 }
IpFilterRule FTP235DataLocal                              5
 {
   IpSourceAddr       9.100.193.235
   IpDestAddr         9.100.199.0/24
   IpService
    {
      Protocol                Tcp
      SourcePortRange         20
      DestinationPortRange    1024 65535
      Direction               BiDirectional OutboundConnect
      Routing                 Local
      SecurityClass           1
    }
   IpGenericFilterActionRef Permit-Log
 }
IpFilterRule NologUdp                               6
  {
    IpSourceAddr       All
    IpDestAddr         All
    IpService
     {
       Protocol                Udp
       SourcePortRange         0
       Direction               Inbound
       Routing                 Local
       SecurityClass           1
     }
    IpGenericFilterActionRef Deny-NoLog
  }
 IpFilterRule InboundIcmp                           7
  {
    IpSourceAddr       All
    IpDestAddr         All
    IpService
     {
       Protocol                Icmp
       Type                    Any
       Direction               Inbound
       Routing                 Local
       SecurityClass           1
     }
    IpGenericFilterActionRef Deny-NoLog
  }
```

1 The parameters of the IpFilterPolicy statement define the logging options and point to the rules that compose the policy that will be loaded into the TCP/IP stack. In our implementation we have enabled the logging and elected to log the packets denied by the implicit rules. We also point to a set of rules called MVI-All.

2 The IpFilterGroup statement creates a group of rules. It is possible to have multiples *group* statements for IP addresses to facilitate the management and organization of the filter policy configuration file by grouping the rules like FTP-specific rules, ICMP rules, and so on, into filter groups. In our implementation we have defined only one group with all rules listed on it.

3 The IpFilterAction defines the action that will be applied to a packet when matching to a specific filter rule. In our case we have defined four actions that will cover all the possibilities: deny and nolog, deny and log, permit and nolog, and permit and log. When defining the filter rules we will point to one of them as a result of matching the rule.

4 The IpFilterRule defines a filter rule. It contains all the data that will be looked in the IP packet TCP/IP header and other characteristics like the direction of the packet. This rule refers to the FTP control port 21. The FTP235ControlLocal has the following information:

- The IPSourceAddr and IPDestAddr are the source and destination of the TCP/IP packet onto which the rule applies.
- The IpService option describes the following packet:
  - The Protocol is TCP. The FTP control port uses the protocol TCP. The other options here are UDP, ICMP, and OSPF.
  - The SourcePortRange of the packet is 21, used by the FTP control. This statement could be used to define a range of ports, but, in our case, we only defined one port. The FTP clients will have to specify the port 21 to connect to the server, This is the default port for FTP.
  - The DestinationPortRange option defines a range of ports that are acceptable as a destination.
  - A packet Direction is either inbound, which means coming from the network to the z/OS image, or outbound, which means leaving the z/OS image towards the network. In this case we defined it as BiDirectional. This option will generate two rules: one outbound referring to the IP addresses and TCP ports as they are defined. Then an inbound rule will be generated by switching the IP address and TCP ports between source and destination. Without using this option two service options should be created, one for each direction, inbound and outbound. The InBoundConnect option means that the initial connection can only be established by inbound packets, which means by clients running outside

the z/OS image. This option can only be used when the TCP protocol is specified.

  – The Routing option specifies whether the packet is addressed to this z/OS stack (in which case it will be *local*) or if this stack is to be used to route the packet to another TCP/IP host (in which case we are dealing with a *routed* packet). It is possible to specify both local and routed by using the *either* option.
  – The SecurityClass option is intended to match the Security Class associated with one or several physical adapters. When a packet is entering or leaving the z/OS stack it will do it by using a physical network interface and the interfaces can be classified in a class when using IP filtering. In our example here, all the rules will be applied on the security class 1 interfaces. This is the Security Class we gave to the adapters for the external networks where the clients are located.

- Finally on the IpFilterRule we have the IpGenericFilterActionRef option, which defines what action is taken for the packets that match this rule. In this case we are permitting the packet and we are logging the action.

5 This is basically like the previous one with two differences:

- The SourcePortRange now is 20. In an active FTP session when some data transfer has to be made the FTP Server opens a connection with the client using a local port 20. In this case the client will be z/OS and the server will be the other host connected to it, in terms of TCP/IP protocol. The port 20 is the default.
- In the Direction option we specified OutboundConnect to allow only the FTP Server in the z/OS image to start the connection, as opposed to having a client trying to establish an inbound connection to port 20.

6 **7** This is a rule to prevent the logging of UDP and ICMP packets. In our network there are some servers that send a lot of UDP and ICMP packets, and we defined those rules to keep the size of the log file small by preventing the log of them. But notice that we elected to have both rules apply to the inbound packets only.

Important: By using IP filtering the stack and the applications will only see packets from acceptable clients and servers. All the other packets will be discarded in the IP layer of the stack.

The log will be the only way to see what packets are being discarded and what is the source of them. The number of logged messages can be another issue if you elect in your syslogd configuration to route the IP filtering messages to the MVS System Console.

IP filtering management

When putting restrictions on systems for the sake of Security, as IP filtering does, there is obviously a need to manage and monitor the system for problems, auditing, and operating conditions. We need to know if the filter rules are being applied, and if they are working as expected. It is possible to make some mistakes when defining the rules that can lead to some problems like either denying legitimate packets or, even worse, allowing packets that should not be permitted to get through the z/OS TCP/IP stack.

Some installations will require that all packet being denied or permitted should be logged and saved for a specific period of time for analysis. Proper procedures have to be put in place to save, retrieve, and search the logged data.

The filter policy will be automatically loaded into the TCP/IP stack at stack startup. To refresh a policy already in place there is a console MODIFY command to tell PAGENT to reload the policy from the file into the stack. Example C-4 shows the PAGENT REFRESH command.

Example: C-4 PAGENT refresh example console display

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

Example C-5 shows the message recorded by syslogd following the PAGENT MODIFY command.

Example: C-5 PAGENT refresh example syslogd messages

```
EZD0816I IPSec Policy updated: 08/29/2005 15:53:51.93 type= Pagent status= Active
```

Other messages are recorded when the PAGENT is in the process of updating the policy in the TCP/IP stack. An example of a refresh sequence is shown in Example C-6.

Example: C-6 PAGENT refresh example PAGENT syslogd messages

```
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  LOG
:pzos_command_handler[01]: EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:check_main_config_file[03]: Main configuration file updated
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:process_time_condition[04]: Entry='FTP235ControlLocal' active, next check in 486 minutes
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule: 'FTP235ControlLocal'
```

```
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule: 'FTP235DataLocal'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule: 'InboundIcmp'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule: 'NologUdp'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule:
'DenyAllRule_Generated___________Inbnd'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished processing policy rule:
'DenyAllRule_Generated___________Outbnd'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_A_PolicyRule[04]: Dummy Rule policy discipline (1), ioctl discipline (C0),
InstanceID '0', Image Name:  'TCPIP'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_A_PolicyRule[04]: Policy change state flag is set for image 'TCPIP'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_A_PolicyRule[04]: Finished installing policy rule: 'Dummy_Rule'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Dummy Rule policy discipline (20), InstanceID
'1125330831', Image Name:  'TCPIP'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_IPSec_FilterTable[04]: Finished installing IPSec filter table
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_A_PolicyRule[04]: Policy change state flag is set for image 'TCPIP'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  EVENT
:pzos_install_A_PolicyRule[04]: Finished installing policy rule: 'Dummy_Rule'
Aug 29 15:53:51 localhost/PAGENT   PAGENT   Pagent[67174683]:  LOG
:instantiate_policies[04]: EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

We modified the PAGENT definition file to simulate an error changing one of the objects and referencing a nonexistent object. The error messages that we are getting at the system console are shown in Example C-7.

Example: C-7   PAGENT refresh example console display with errors

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8438I PAGENT POLICY DEFINITIONS CONTAIN ERRORS FOR TCPIP : IPSEC
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

More specific information pertaining to the error message EZZ8438I is given in the syslogd message log for the PAGENT. Example C-8 shows one of these messages pertaining to the error simulated above.

Example: C-8   PAGENT refresh example PAGENT syslogd messages with errors

```
Pagent[67174683]:  OBJERR :process_IPSec_group_ref[04]: Referenced IP Filter group 'MVI-All1'
not found or error in referenced object
definition
```

Using the ipsec command

The **ipsec** command can be used to manage the IP filtering policies. For a complete description of the **ipsec** command syntax, security, and options see the manual *IP System Administrator Commands,* SC31-8781.

The **ipsec** command can be used to:

- Display current filter rules and change the filter rule set that the stack is using.
- Activate, deactivate, display, and refresh manual and dynamic IPSec tunnels.
- Deactivate, display, and refresh IKE-specific tunnels.
- Display stack interfaces, including their security class and DVIPA status.
- For a particular type of data traffic between two specific endpoints, display which filter rules apply.

Example C-9 shows the use of the **ipsec** command to display the filter rule being currently used by the stack.

Example: C-9   ipsec display command

```
ipsec -f display -r detail

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 08:06:34 2005
Primary:  Filter         Function: Display          Format:   Detail
Source:   Stack Policy 1 Scope:    Current          TotAvail: 8   2
Logging:  Yes            Predecap: Yes              DVIPSec:  No
NatKeepAlive:  0

FilterName:                 FTP235ControlLocal
FilterNameExtension:        1
GroupName:                  MVI-All
Type:                       Generic
State:                      Active
Action:                     Permit
Scope:                      Local
```

```
Direction:                    Outbound
SecurityClass:                1
Logging:                      All
Protocol:                     TCP(6)
TCPQualifier:                 Connect Inbound
ProtocolGranularity:          Rule
SourceAddress:                9.100.193.235
SourceAddressGranularity:     Packet
SourcePort:                   21
SourcePortGranularity:        Rule
DestAddress:                  9.100.199.0
DestAddressPrefix:            24
DestAddressGranularity:       Packet
DestPort:                     1024
DestPortRange:                65535
DestPortGranularity:          Rule
.
.
.
***********************************************************************
FilterName:                   DenyAllRule_Generated___________Inbnd  3
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
SecurityClass:                0
Logging:                      All
Protocol:                     All
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressPrefix:          0
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressPrefix:            0
DestAddressGranularity:       Packet
DestPort:                     All
DestPortGranularity:          Rule
***********************************************************************
FilterName:                   DenyAllRule_Generated___________Outbnd  4
TunnelID:                     0x00
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Outbound
```

```
SecurityClass:              0
Logging:                    All
Protocol:                   All
ProtocolGranularity:        Rule
SourceAddress:              0.0.0.0
SourceAddressPrefix:        0
SourceAddressGranularity:   Packet
SourcePort:                 All
SourcePortGranularity:      Rule
DestAddress:                0.0.0.0
DestAddressPrefix:          0
DestAddressGranularity:     Packet
DestPort:                   All
DestPortGranularity:        Rule
***********************************************************************

8 entries selected
```

1 *Stack Policy* means that the rules that are in effect are from the PAGENT configuration file.

2 There are eight rules listed here. If you look at the configuration file there will be only six defined there. The other two are the implicit rules listed at the end of the example.

3 4 These two rules are the implicit rules created automatically. It is not possible to remove or disable them.

Using the **ipsec** command it is possible to switch between the filter and the default policy, as shown in Example C-10.

Example: C-10 ipsec example of switching between default and filter policies

```
ipsec -f default           1

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 08:42:31 2005
Primary:  Filter          Function: Default

Stack Profile filters now in effect 2
```

1 The **ipsec -f default** switches to the default rules defined in the IPSEC statement in the TCP/IP stack profile.

2 This message says that the stack profile filter rules (the default rules) are in effect now.

In our implementation we did not define any default rule in the IPSEC statement. Therefore the only rules that will appear as a result of the **`ipsec -f display`** command will be the implicit rules. Example C-11 shows this.

Example: C-11 ipsec -f display without a default policy installed

```
(MVIT)-/u/gianca-(NONO)-(148):ipsec -f display

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 08:46:18 2005
Primary:  Filter         Function: Display          Format:   Detail
Source:   Stack Profile   Scope:    Current          TotAvail: 2
Logging:  Yes            Predecap: No               DVIPSec:  No
NatKeepAlive:  0

FilterName:                   SYSDEFAULTDENYRULE
FilterName:                   SYSDEFAULTDENYRULE
FilterNameExtension:          1
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Outbound
SecurityClass:                0
Logging:                      All
Protocol:                     All
ProtocolGranularity:          Packet
SourceAddress:                0.0.0.0
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortGranularity:        Packet
DestAddress:                  0.0.0.0
DestAddressGranularity:       Packet
DestPort:                     All
DestPortGranularity:          Packet
**********************************************************************
FilterName:                   SYSDEFAULTDENYRULE
FilterNameExtension:          2
FilterNameExtension:          2
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
SecurityClass:                0
Logging:                      All
Protocol:                     All
ProtocolGranularity:          Packet
SourceAddress:                0.0.0.0
SourceAddressGranularity:     Packet
```

```
SourcePort:                  All
SourcePortGranularity:       Packet
DestAddress:                 0.0.0.0
DestAddressGranularity:      Packet
DestPort:                    All
DestPortGranularity:         Packet
***********************************************************************

2 entries selected
```

Now we can see that the source of the policy is the stack profile and there are only two rules defined, which are the implicit rules.

To switch back to the filter policy we can use the command **ipsec -f reload**, as shown in Example C-12.

Example: C-12 ipsec -f reload command

```
ipsec -f reload

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 08:50:58 2005
Primary:  Filter          Function: Reload

Stack Policy filters now in effect
```

Tip: This capability of switching between the filter and the default policies can prove to be useful in special cases. You can use the filter policy as the enterprise policy, with all rules that are needed for normal operation, and have the default policy being loaded in some emergency or maintenance situations. In our implementation, the default policy, when installed, will deny all the traffic.

With the **ipsec** command it is also possible to display the characteristics of the TCP/IP stack interfaces to the network, their current status, the security class they are in, and so on. Example C-13 shows an output of the **ipsec** command.

Example: C-13 ipsec interface display

```
ipsec -i display -r detail

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 09:01:38 2005
Primary:  Interface       Function: Display          Format:   Detail
Source:   Stack           Scope:    Current          TotAvail: 7

InterfaceName                ETH400
SecurityClass                001
Active                       Yes
```

```
DVIPA                         No
Address                       9.100.193.229
InterfaceName                 IUTIQDEC
SecurityClass                 002
Active                        Yes
DVIPA                         No
Address                       10.1.1.1
InterfaceName                 EZASAMEMVS
SecurityClass                 002
Active                        No
DVIPA                         No
Address                       10.1.2.1
InterfaceName                 EZAXCFIS
SecurityClass                 002
Active                        Yes
DVIPA                         No
Address                       10.1.2.1
InterfaceName                 EZAXCFIV
SecurityClass                 002
Active                        Yes
DVIPA                         No
Address                       10.1.2.1
InterfaceName                 VIPL0964C1EA
SecurityClass                 000
Active                        Yes
DVIPA                         Yes
Address                       9.100.193.234
InterfaceName                 VIPL0964C1EB
SecurityClass                 000
Active                        Yes
DVIPA                         Yes
Address                       9.100.193.235
7 entries selected
```

We can see the security classes for all interfaces and whether they are activated. Note that the DVIPA interfaces have a security class of 0 (no security class), which cannot be changed.

Another useful option in the **ipsec** command is the test option. Using it, it is possible to test whether specific traffic will be permitted or denied and what specific rules will be applied on it. Example C-14 shows the use of the test option.

Example: C-14 ipsec test command

```
ipsec -t 9.100.199.233 9.100.193.235 tcp 1050 21 in 1 1

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 09:13:05 2005
```

```
Primary:  IP Traffic Test Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    n/a               TotAvail: 2
TestData: 9.100.199.233  9.100.193.235  tcp 1050 21 in 1

FilterName:                   FTP235ControlLocal
FilterNameExtension:          2
GroupName:                    MVI-All
Type:                         Generic
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
SecurityClass:                1
Logging:                      All
Protocol:                     TCP(6)
TCPQualifier:                 Connect Inbound
ProtocolGranularity:          Rule
SourceAddress:                9.100.199.0
SourceAddressPrefix:          24
SourceAddressGranularity:     Packet
SourcePort:                   1024
SourcePortRange:              65535
SourcePortGranularity:        Rule
DestAddress:                  9.100.193.235
DestAddressGranularity:       Packet
DestPort:                     21
DestPortGranularity:          Rule
***********************************************************************
FilterName:                   DenyAllRule_Generated__________Inbnd
Type:                         Generic
State:                        Active
Action:                       Deny
Scope:                        Both
Direction:                    Inbound
SecurityClass:                0
Logging:                      All
Protocol:                     All
ProtocolGranularity:          Rule
SourceAddress:                0.0.0.0
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortGranularity:        Rule
DestAddress:                  0.0.0.0
DestAddressGranularity:       Packet
DestPort:                     All
DestPortGranularity:          Rule
***********************************************************************

2 entries selected
```

```
ipsec -t 9.100.199.233 9.100.193.235 tcp 1050 21 out 2

CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 09:17:47 2005
Primary:  IP Traffic Test Function: Display           Format:   Detail
Source:   Stack Policy    Scope:    n/a               TotAvail: 1
TestData: 9.100.199.233  9.100.193.235  tcp 1050 21 out

FilterName:                  DenyAllRule_Generated__________Outbnd
Type:                        Generic
State:                       Active
Action:                      Deny
Scope:                       Both
Direction:                   Outbound
SecurityClass:               0
Logging:                     All
Protocol:                    All
ProtocolGranularity:         Rule
SourceAddress:               0.0.0.0
SourceAddressGranularity:    Packet
SourcePort:                  All
SourcePortGranularity:       Rule
DestAddress:                 0.0.0.0
DestAddressGranularity:      Packet
DestPort:                    All
DestPortGranularity:         Rule
***********************************************************************

1 entries selected
```

1 In this test we simulate a client from the address 9.100.199.233 using the local port 1050 trying to send a packet to the address 9.100.193.235 in the port 21 as a inbound packet using the security class 1. The display shows that two rules will be applied. The first one is the rule that permits the packet to the system and the second one is the implicit rule that would deny the packet in the absence of the first one or if the first one does not match the packet. All the rules will be listed here. In a real situation only the first rule will be used and the search will always stop in the first rule.

2 In the second example we just switch the direction of the packet, and the rule that would be applied in this case is the implicit deny all outbound. There is no rule defined to permit a packet to be issued from the z/OS TCP/IP stack.

Using the pasearch command

The **pasearch** command can be used to display the IP filtering policies. The format is a little different but basically contains the same information as the **ipsec** command in a different format. The **pasearch -v a** command will show the IP security policies, as shown in Example C-15.

Example: C-15 pasearch IP security policies display

```
pasearch -v a

TCP/IP pasearch CS V1R7                   TCP/IP Image:       TCPIP
  Date:                 08/30/2005        Time:  10:54:22
  IPSec Instance Id:    1125330831

policyRule:             FTP235ControlLocal
  Rule Type:            IpFilter
  Version:              3                 Status:             Active
  GroupName:            MVI-All
  Weight:               106               ForLoadDist:        False
  Priority:             6                 Sequence Actions:   Don't Care
  No. Policy Action:    1                 ConditionListType:  CNF
  IpSecType:            policyIpFilter
  policyAction:         Permit-Log
   ActionType:          IpFilter GenericFilter
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       1111111111111111111111111111111
   Last to First:       1111111111111111111111111111111
   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:    00:00             To TimeOfDay UTC:   00:00
   TimeZone:            Local
  IpSec Condition Work:                   NegativeIndicator: Off
   IpFilter Condition:
    Source Address:
     FromAddr:          9.100.193.235
     ToAddr:            9.100.193.235
    Destination Address:
     FromAddr:          All
     ToAddr:            All
 IpSec Condition Work:                   NegativeIndicator: Off
  IpFilter Condition:
   Source Address:
    FromAddr:          All
    ToAddr:            All
   Destination Address:
    FromAddr:          9.100.199.0
    Prefix:            24
 IpSec Condition Work:                  NegativeIndicator: Off
  IpFilter Condition:
   Source Address:
    FromAddr:          All
    ToAddr:            All
   Destination Address:
    FromAddr:          All
    ToAddr:            All
   Service Condition:
    Protocol:          TCP  (6)
     SrcPortFrom:      21                 SrcPortTo:         21
     DestPortFrom:     1024               DestPortTo:        65535
    Direction:         Bidirectional   InboundConnect
    RouteType:         Local              SecurityClass:     1

 IpFilter Action:      Permit-Log
   Version:            3                  Status:            Active
   Scope:              GenericFilter
   ipFilterAction:     Permit             IpFilterLogging:   Yes
```

For more information about the **pasearch** command look in the IBM manual *IP System Administrator Commands,* SC31-8781.

IP filtering logging considerations

The information logged is provided by two z/OS components:

- The PAGENT, which issues the IP filtering policies loading and management messages
- The TRMD, which issues the IP filter actions messages

All the messages are written by the syslogd address space. In our configuration we split the PAGENT and TRMD messages into different files to facilitate the viewing of the messages depending on what they pertain to.

Depending on the contents of the filtering rules, the number of log messages being generated can be very important. As we said previously, in our configuration we defined some rules preventing messages from being written, and by doing so controlling, in some way, the size of the messages logs.

In Example C-16 we show the syslogd configuration file that specifies the log files that we have been using for the PAGENT and TRMD messages.

Example: C-16 syslogd configuration for IP filtering

```
pagent.*.*.* /tmp/pagent%Y%m%d.log
trmd.*.*.* /tmp/trmd%Y%m%d.log
```

Example C-17 shows how the TRMD messages for IP filtering look.

Example: C-17 trmd message logging

```
Aug 25 15:57:25 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZZ8495I TRMD STARTED 1
Aug 25 15:57:25 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZA8545I TRMD IPSEC LOGGING
ACTIVATED 2
Aug 25 15:57:25 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZZ8500I TRMD INITIALIZATION
COMPLETE
Aug 26 00:49:37 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZD0815I Packet denied by
policy: 08/25/2005 23:49:29.09 filter rule= DenyAllRule_Generated__________Inbnd ext=
sipaddr= 9.65.165.178 dipaddr= 9.100.193.235 proto= tcp(6) sport= 2397 dport= 445 -= Interface=
9.100.193.229 (I) secclass= 1 dest= local len= 64 vpnaction= N/A tunnelID= N/A
Aug 26 16:08:28 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZD0816I IPSec Policy
updated: 08/26/2005 15:08:19.09 type= Pagent status= Active 3
Aug 26 16:08:38 localhost/TRMD     TRMD1    TRMD.TCPIP[33620265]: EZD0814I Packet permitted:
08/26/2005 15:08:35.47 filter rule= FTP235ControlLocal ext= 1 sipaddr= 9.100.193.235 dipaddr=
9.100.199.233 proto= tcp(6) sport= 21 dport= 1991 -= Interface= 9.100.193.229 (O) secclass= 1
dest= local len= 44 vpnaction= N/A tunnelID= N/A
```

1 2 When TRMD starts there are some informational messages on the initialization process and the activation of the IPSEC logging function.

3 Another example of the policy update messages.

There are also messages pertaining to permitted and denied packets. These messages contain the information about the rule that was used and how they matched the contents of the packets.

IP filtering and Sysplex Distributor

This appendix carries on with the IP filtering setups in a Sysplex, in the particular context of a Sysplex Distributor configuration that we describe in Figure D-1 on page 244.

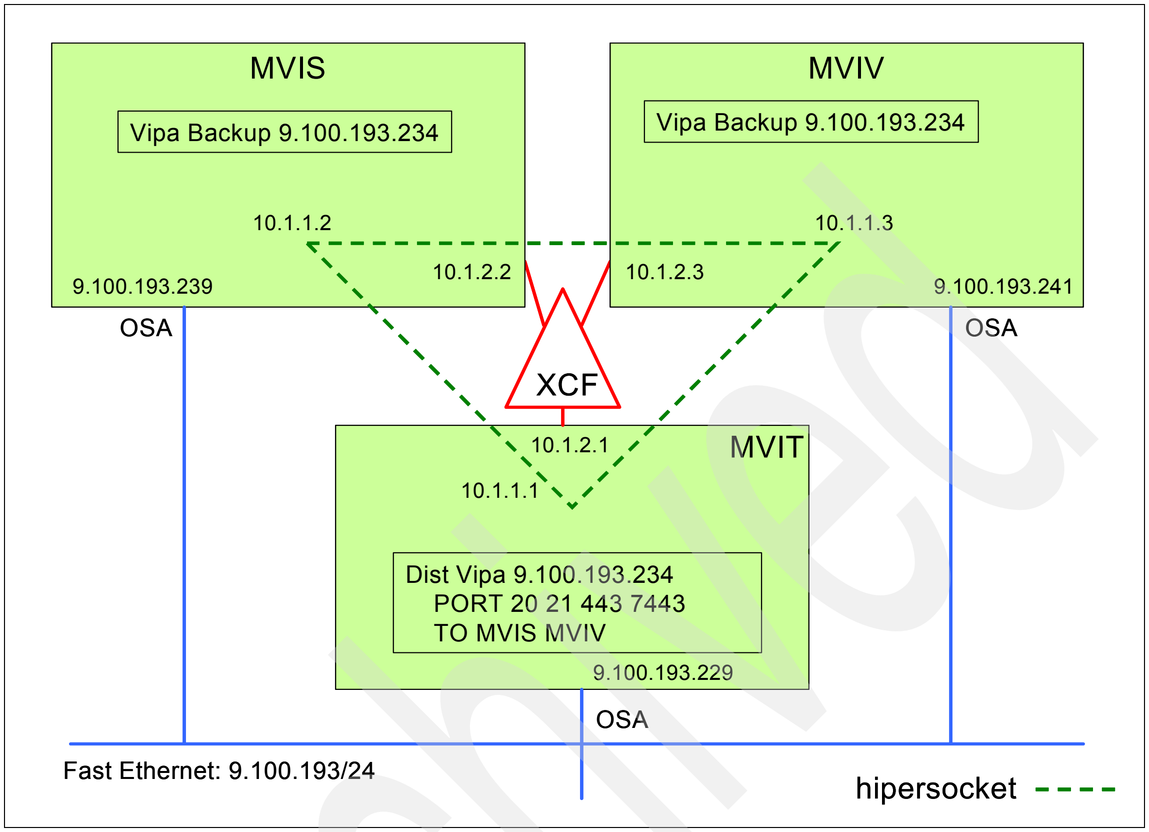


Figure D-1 Sysplex distributor scenario

The distributed DVIPA 9.100.193.234 is being distributed to the MVIS and MVIV members for ports 20 and 21 (FTP server), 443 (http server with SSL), and 7443 for the TN3270 server.

System MVIS is the primary backup for the DVIPA and MVIV is the secondary backup.

Example D-1 and Example D-2 on page 245 show the PROFILE.TCPIP datasets contents for the three members of the Sysplex Distributor configuration.

Example: D-1 MVIT Sysplex distributor statements

```
IPCONFIG
   NODATAGRAMFWD
   SYSPLEXROUTING
   IPSECURITY
   IGNOREREDIRECT
   DYNAMICXCF 10.1.2.1/24 0 SECCLASS 2

; OSA 400
DEVICE ETH400 MPCIPA NONROUTER
LINK   ETH400 IPAQENET ETH400 READSTORAGE AVG INBPERF MINLATENCY SECCLASS 1
; HiperSockets Chpid ED
DEVICE IUTIQDED MPCIPA
LINK   IUTIQDED IPAQIDIO IUTIQDED  READSTORAGE AVG SECCLASS 2

HOME
   9.100.193.229 ETH400
   10.1.1.1      IUTIQDED

BEGINROUTES
   ROUTE 9.100.193.0/22 =             ETH400   MTU 1500
   ROUTE 10.1.1.0/24    =             IUTIQDED MTU 8192
   ROUTE DEFAULT        9.100.193.69 ETH400   MTU 1500
ENDROUTES

VIPADYNAMIC
   VIPADEFINE MOVEABLE IMMEDIATE 255.255.255.0 9.100.193.234
   VIPADEFINE MOVEABLE IMMEDIATE 255.255.255.0 9.100.193.235
   VIPADISTRIBUTE DISTM ROUNDROBIN 9.100.193.234 PORT 20 21 443 7443
      DESTIP 10.1.2.2 10.1.2.3
ENDVIPADYNAMIC
```

Example: D-2 MVIS Sysplex distributor statements

```
IPCONFIG
   NODATAGRAMFWD
   SYSPLEXROUTING
   IPSECURITY
   IGNOREREDIRECT
   DYNAMICXCF 10.1.2.2/24 0 SECCLASS 2

; OSA 400
DEVICE ETH400 MPCIPA NONROUTER
LINK   ETH400 IPAQENET ETH400 READSTORAGE AVG INBPERF MINLATENCY SECCLASS 1
; HiperSockets Chpid ED
DEVICE IUTIQDED MPCIPA
LINK   IUTIQDED IPAQIDIO IUTIQDED  READSTORAGE AVG SECCLASS 2

HOME
   9.100.193.239 ETH400
   10.1.1.2      IUTIQDED
;
BEGINROUTES
   ROUTE 9.100.193.0/22 =             ETH400   MTU 1500
   ROUTE 10.1.1.0/24    =             IUTIQDED MTU 8192
   ROUTE DEFAULT        9.100.193.69 ETH400   MTU 1500
ENDROUTES

VIPADYNAMIC
   VIPABACKUP 10 9.100.193.234
ENDVIPADYNAMIC
```

Example: D-3   MVIV Sysplex distributor statements

```
IPCONFIG
   NODATAGRAMFWD
   SYSPLEXROUTING
   IPSECURITY
   IGNOREREDIRECT
   PATHMTUDISCOVERY
   DYNAMICXCF 10.1.2.3/24 0 SECCLASS 2

; OSA 400
DEVICE ETH400 MPCIPA NONROUTER
LINK   ETH400 IPAQENET ETH400 READSTORAGE AVG INBPERF MINLATENCY SECCLASS 1
; HiperSockets Chpid ED
DEVICE IUTIQDED MPCIPA
LINK   IUTIQDED IPAQIDIO IUTIQDED  READSTORAGE AVG SECCLASS 2

HOME
   9.100.193.241 ETH400
   10.1.1.3      IUTIQDED
;
BEGINROUTES
   ROUTE 9.100.193.0/22 =            ETH400   MTU 1500
   ROUTE 10.1.1.0/24    =            IUTIQDED MTU 8192
   ROUTE DEFAULT        9.100.193.69 ETH400   MTU 1500
ENDROUTES

VIPADYNAMIC
   VIPABACKUP 05 9.100.193.234
ENDVIPADYNAMIC
```

Example D-3 shows the filtering rules that we have built for the Sysplex Distributor scenario, taking into consideration the communication between the distributed DVIPA and clients in the network 9.100.199.0.

Example: D-4   IP filtering rules for the Sysplex distributor scenario

```
IpFilterPolicy
 {
   FilterLogging       On
   IpFilterLogImplicit Yes
```

```
    IpFilterGroupRef    MVI-All
  }
 IpFilterGroup MVI-All
  {
    IpFilterRuleRef TN3270-7443-Sysplex
    IpFilterRuleRef Https-Sysplex
    IpFilterRuleRef FTP234-Control-Dist
    IpFilterRuleRef FTP234-Data-Dist
    IpFilterRuleRef Tn-Sysplex-Dist-Xcf
    IpFilterRuleRef Tn-Sysplex-Target-Xcf
    IpFilterRuleRef Https-Sysplex-Dist-Xcf
    IpFilterRuleRef Https-Sysplex-Target-Xcf
    IpFilterRuleRef Ftp-Sysplex-Dist-Xcf
    IpFilterRuleRef Ftp-Sysplex-Target-Xcf
    IpFilterRuleRef VipaRoute-GRE
  }
 IpGenericFilterAction Permit-Log
  {
    IpFilterAction  Permit
    IpFilterLogging Yes
  }
 IpGenericFilterAction Permit-NoLog
  {
    IpFilterAction  Permit
    IpFilterLogging No
  }
 IpGenericFilterAction Deny-Log
  {
    IpFilterAction  Deny
    IpFilterLogging Yes
  }
 IpGenericFilterAction Deny-NoLog
  {
    IpFilterAction  Deny
    IpFilterLogging No
  }
 IpFilterRule TN3270-7443-Sysplex
  {
    IpSourceAddr       9.100.193.234
    IpDestAddr         9.100.199.0/24
    IpService
     {
       Protocol                Tcp
       SourcePortRange         7443
       DestinationPortRange    1024 65535
       Direction               BiDirectional InboundConnect
       Routing                 Either
       SecurityClass           1
     }
```

```
      IpGenericFilterActionRef   Permit-Log
    }
   IpFilterRule Https-Sysplex
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         443
         DestinationPortRange    1024 65535
         Direction               BiDirectional InboundConnect
         Routing                 Either
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
    }
   IpFilterRule FTP234-Control-Dist
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         21
         DestinationPortRange    1024 65535
         Direction               BiDirectional InboundConnect
         Routing                 Either
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
    }
   IpFilterRule FTP234-Data-Dist
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         20
         DestinationPortRange    1024 65535
         Direction               BiDirectional OutboundConnect
         Routing                 Either
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
    }
   IpFilterRule Https-Sysplex-Dist-Xcf
    {
```

```
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     443
       Direction                Outbound
       Routing                  Routed
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
IpFilterRule Https-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     443
       Direction                Inbound
       Routing                  Local
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
IpFilterRule Tn-Sysplex-Dist-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     7443
       Direction                Outbound
       Routing                  Routed
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
IpFilterRule Tn-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                Tcp
       SourcePortRange         1024 65535
       DestinationPortRange    7443
       Direction               Inbound
       Routing                 Local
       SecurityClass           2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Ftp-Sysplex-Dist-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                Tcp
       SourcePortRange         1024 65535
       DestinationPortRange    20 21
       Direction               Outbound
       Routing                 Routed
       SecurityClass           2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Ftp-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                Tcp
       SourcePortRange         1024 65535
       DestinationPortRange    20 21
       Direction               Inbound
       Routing                 Local
       SecurityClass           2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule VipaRoute-GRE
  {
    IpSourceAddr       10.1.0.0/16
    IpDestAddr         10.1.0.0/16
   IpService
    {
      Protocol                47
      Direction               Outbound
      Routing                 Local
      SecurityClass           2
    }
   IpGenericFilterActionRef   Permit-Log
 }
```

We have for each one of the services one rule for:

- Allowing an inbound packet with a routed option. The sysplex distributor will send the packet to the target system using an XCF interface.
- Allowing an outbound packet to be sent via the XCF interface.
- Allowing the inbound packet on the target image
- Allowing the outbound packet on the target image to the client.

This rules are shared by all the members of the Sysplex. In the case of one of the targets becoming a distributor stack the rules will apply to it as well.

The rule VipaRoute-GRE is an example of a rule that has to be implemented in the case of VIPAROUTE being used. The GRE protocol number is 47. The packet will be encapsulated by GRE and decapsulated on the target before being presented to the IP filtering policy. There is no need for an inbound rule at the target.

Tip: The rules will be checked from top to bottom. They will be defined in the order in which they appear in the IpFIlterPolciy/IpFilterGroup statements. If you know what are the most-used IP services you can sort the rules and keep the most-used services at the top of the definition to reduce searching at the TCP/IP stack filtering logic level.

E

AT-TLS implementation

This appendix provides implementation details for the z/OS V1R7vApplication Transparent TLS (AT-TLS) protecting the TN3270 communications.

Creating the configuration file

The z/OS Network Security Configuration Assistant was used to quickly create an AT-TLS configuration file. The file, however, required some modifications after its creation. The modified file is shown in Example E-1. An explanation of these statements, along with the changes, follows.

Example: E-1 AT-TLS configuration file

```
TTLSRule                          AT-TLS1~1 1
{
  LocalAddrRef                    addr1
  RemoteAddrSetRef                addr2
  LocalPortRangeRef               portR1
  RemotePortRangeRef              portR2
  Direction                       Inbound
  Priority                        255
  TTLSGroupActionRef              enableGrpAct~MVSIT
  TTLSEnvironmentActionRef        eAct1~TN3270-Server
  TTLSConnectionActionRef         cAct1~TN3270-Server
}
TTLSGroupAction                   enableGrpAct~MVSIT
{
  TTLSEnabled                     On 2
  TTLSGroupAdvancedParmsRef       Syslogparm
}
TTLSGroupAdvancedParms           Syslogparm 3
{
  SyslogFacility                  auth
}
TTLSEnvironmentAction             eAct1~TN3270-Server
{
  HandshakeRole                   Server 4
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR~MVSIT
  TTLSEnvironmentAdvancedParmsRef TN3270AdParm
  TTLSGskAdvancedParmsRef         TN3270GSK
}
TTLSGskAdvancedParms TN3270GSK 5
{
   GSK_SYSPLEX_SIDCACHE ON
   GSK_V3_SESSION_TIMEOUT 1200
   GSK_V3_SIDCACHE_SIZE  20
}
TTLSEnvironmentAdvancedParms TN3270AdParm
{
ClientAuthType PassThru 6
}
TTLSConnectionAction              cAct1~TN3270-Server
```

```
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~AT-TLS__Platinum
  Trace                           255
}
TTLSKeyringParms                  keyR~MVSIT 7
{
  Keyring                         tlsKeyring
}
TTLSCipherParms                   cipher1~AT-TLS__Platinum 8
{
  V3CipherSuites                  TLS_RSA_WITH_AES_256_CBC_SHA
}
IpAddr                            addr1
{
  Addr                            9.100.193.234 9
}
IpAddrSet                         addr2
{
  Prefix                          9.100.0.0/16
}
PortRange                         portR1
{
  Port                            8443 10
}
PortRange                         portR2
{
  Port                            1024-65535
}
```

1 This is the statement that specifies an AT-TLS rule. Many of these statements point to other statements, which will be explained later. Note here that this rule is defined for inbound connections only and has been given the highest possible priority of 255, which is what will be looked at first for any match with the connection.

2 TTLSEnabled is the statement that turns on the AT-TLS function.

3 The statement was added, pointed at by a reference in the preceding statement, in order to tag messages with the specific syslog *facility* name of *auth*, to clearly separate in message logs the AT-TLS messages from others.

4 This statement controls who initiates the handshake. If in the Server role, the stack AT-TLS will wait for an inbound hello from the client.

5 These policy statements were all added manually. The GSK_SYSPLEX_SIDCACHE is there to turn on the sysplex-wide session ID caching, as described in 7.5, "Sysplex session ID caching" on page 187.

The GSK_V3_SESSION_TIMEOUT parameter controls the duration of a TLS session ID. The appropriate value for this parameter should be similar to the value that is used by the client. In other words, the duration of a cached session should be the same at both the client and the server end of a TLS session.

Each cached session ID requires storage. The GSK_V3_SIDCACHE_SIZE controls the number of active session IDs that can be cached concurrently. Note that whether a session ID is reused is ultimately controlled by the client. If the client does not send in a cached session ID, then session ID reuse cannot occur.

6 At the time of this writing, the z/OS Secure Network Configuration Assistant appeared to default to requiring client authentication. And this is the function of the policies themselves. If ClientAuthType is not coded, it will default to Required. This policy statement was added manually to turn off client authentication.

7 The key ring used was permitted to the user ID under which the TN3270 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.

8 The strongest ciphersuite was selected, including the new AES encryption algorithm.

9 The sysplex distributor address is used. However, AT-TLS only needs to be active on the target hosts. The distributing host itself is not involved with AT-TLS.

10 The destination port used for testing was 8443.

Because AT-TLS functions at a different level from filtering and VPN policies, there is no need to integrate the AT-TLS policies into the rules used for VPN and filtering. In fact, a separate PAGENT configuration directive is used altogether. The complete policy agent configuration file can be seen in Example E-2.

Example: E-2 Policy agent configuration file with AT-TLS

```
TcpImage TCPIP FLUSH PURGE 1800
LogLevel 31
IpsecConfig //'SYSM.TCPCFG.TCPPARMS(IPSECP0)'
TTLSConfig  //'SYSM.TCPCFG.TCPPARMS(TTLS0)'
```

Member TTLS0 contains the policy statements identified in Example E-1 on page 254.

F

Sysplex session-ID caching setup example

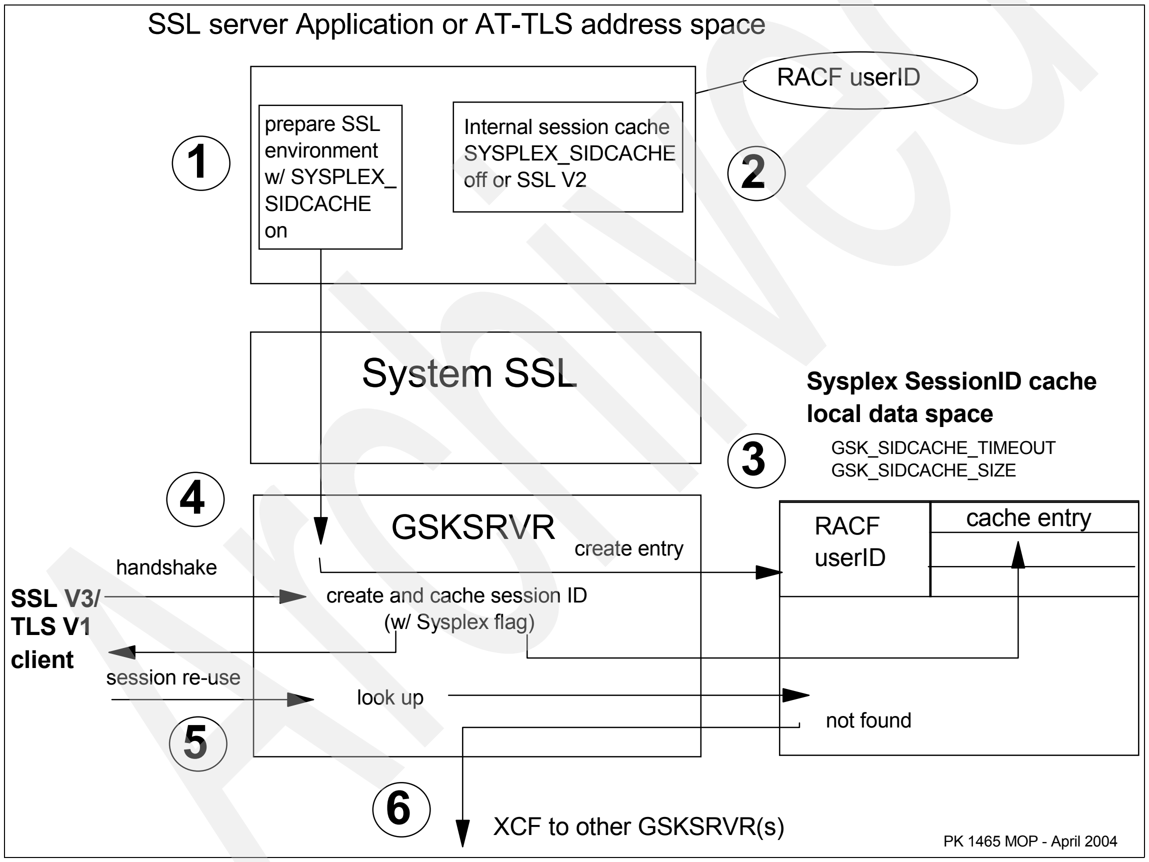
This appendix provides:

- Explanations and a setup example for the System SSL Sysplex session-ID caching function
- Excerpts of System SSL traces to demonstrate the expected sequence of events when AT-TLS exploits the Sysplex session-ID caching

Sysplex session-ID caching configuration example

The Sysplex session-ID cache requires the usage of the System SSL GSKSRVR server task. The server is responsible for managing the local cache and retrieve session-ID from others System SSL servers on the sysplex. For a full explanation of how to configure the System SSL server please see the manual *z/OS Cryptographic Services System Secure Sockets Layer Programming,* SC24-5901.

Figure F-1 shows the implementation and interactions of System SSL session-ID caching.



SSL server Application or AT-TLS address space
RACF userID
1
prepare SSL environment w/ SYSPLEX_ SIDCACHE on
Internal session cache SYSPLEX_SIDCACHE off or SSL V2
2
System SSL
Sysplex SessionID cache local data space
3
GSK_SIDCACHE_TIMEOUT
GSK_SIDCACHE_SIZE
4
GSKSRVR
create entry
RACF userID
cache entry
handshake
SSL V3/ TLS V1 client
create and cache session ID (w/ Sysplex flag)
session re-use
look up
not found
5
6
XCF to other GSKSRVR(s)
PK 1465 MOP - April 2004

Figure F-1 System SSL session-ID caching flow

SSL session information, including symmetric keys values, is kept in a session cache. The session-ID is actually an index to the session cache entries.

System SSL supports an internal local session cache (that is, a cache in the requesting application address space) or optionally provides communicating instances of caches in sysplex members (that is, the *sysplex cache*). In order to use a sysplex cache instead of the local internal cache, the GSKSRVR started task must be running in the sysplex member.

All instances of the same SSL server in the sysplex have the same RACF user ID or are permitted to the GSK.SIDCACHE.<owner> profile in the FACILITY class to get access to <owner>'s cache.

Sysplex cache support implies that the System SSL server task GSKSRVR is started.

1. An SSL server application calls System SSL to initialize an environment. GSK_SYSPLEX_SIDCACHE can be turned on as an exported environment variable or via the System SSL API.
2. If for some reason sysplex caching cannot be turned on, the application reverts to using the internal cache it provides, via the System SSL API, in its address space. SSL V2 session IDs are stored in the internal cache only.
3. A sysplex session cache entry is eventually created for the application user ID in the data space managed by the local instance of GSKSRVR. Cache size and session time out values are fixed by environment variables.
4. When started, GSKSRVR handles the handshakes. Session information is created for an handshake and a session ID is returned to the client and used as an index to the session information stored in the sysplex cache (in the entry for the application user ID). The session ID has a flag set to indicate that it is a sysplex cache session ID, as opposed to an internal cache session ID, and contains the issuing system's name.
5. When a client gets back for a new connection with a previously provided sysplex session ID, GSKSRVR looks for the session ID in its local sysplex cache data space.

   Note: There is no automatic propagation of the session information to other instances of GSKSRVR. The information tagged with the session ID is to remain in the local sysplex cache and will be transmitted on request only by another GSKSRVR instance.

6. If not found in its local sysplex cache, GSKSRVR sends an XCF request to the other instances of GSKSRVR in the sysplex in order to get the session information.

In our configuration the JCL for the started task is shown in Example F-1.

Example: F-1 System SSL server started task

```
//*******************************************************************
//*                                                                 *
//* Procedure for starting the System SSL Server GSKSRVR *
//*                                                                 *
//*******************************************************************
//GSKSRVR  PROC  REGSIZE=0M,OUTCLASS='A'
//*------------------------------------------------------------------
//GO       EXEC  PGM=GSKSRVR,REGION=&REGSIZE,TIME=1440,
//  PARM=('ENVAR("HOME=/etc/gskssl/server"),TERM(DUMP)               X
//             / 1>DD:STDOUT 2>DD:STDERR')
//STDOUT   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//STDERR   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//SYSOUT   DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
//CEEDUMP  DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
```

A file named envar is located in the HOME directory and contains the environment variables to control the usage of the GSKSRVR started task.

Example F-2 shows the *envar* file that we use in our scenario.

Example: F-2 GSKSRVR envar file

```
GSK_LOCAL_THREADS=5            1
GSK_SIDCACHE_SIZE=20           2
GSK_SIDCACHE_TIMEOUT=60        3
#  System configuration options
TZ=EST5EDT
LANG=En_US.IBM-1047
NLSPATH=/usr/lib/nls/msg/%L/%N
#  Debug options
GSK_TRACE_FILE=/tmp/gsktrace.trc 4
GSK_TRACE=0x3F 5
```

1 This variable specifies the number of threads that will be used to handle program call requests from SSL applications running on the same system as the GSKSRVR started task.

2 This variable specifies the size of the sysplex session cache in megabytes.

3 This variable specifies the sysplex session cache entry time out in minutes.

4 This variable sets the path for the trace file to be used.

5 This variable specifies the trace options.

Every instance of the System SSL server on all Sysplex images has to be configured.

In our scenario we are using an HTTP Server to test the use of the System SSL Sysplex session-ID caching. The HTTP server is the application that calls System SSL and as such has also to specify the proper environment variables that will be used by System SSL. These variables are shown in Example F-3.

Note the GSK_SYSPLEX_SIDCACHE variable with a value of ON.

Example: F-3 HTTP server system SSL environment variables

```
GSKV3CACHESIZE=1024             1
GSKV2CACHESIZE=512              2
GSK_TRACE=0x3F                  3
GSK_TRACE_FILE=/tmp/webtrace.trc 4
GSK_SYSPLEX_SIDCACHE=ON         5
```

1 **2** These entries set the size of the application cache in terms of number of entries for the SSL V2 and V3 protocols.

3 This variable sets the trace option for System SSL.

4 This is the path for the trace file.

5 This option enables the usage of the Sysplex session-ID caching. Only TLS V1 and SSL V3 session information are supported by the Sysplex session-ID. SSL V2 session information is always stored in the application local cache.

We started the HTTP server only with the SSL 443 port configured in the sysplex members MVIS and MVIV. The MVIT member was distributing the traffic between them.

Using a browser, we started many https sessions against the Sysplex Distributor VIPA to port 443. Then we issued the command **`f gsksrvr,display xcf`** that displays all the GSKSRVR server instances in the XCF group GSKSRVGP. The resulting display is shown in Example F-4.

Example: F-4 System SSL server display xcf command

```
RESPONSE=MVIT
 GSK01008I Sysplex status
 Server                      System    Status
 MVIT-BD952D0B-E73077AE      MVIT      ACTIVE
 MVIS-BD9511A0-E7304B05      MVIS      ACTIVE
 MVIV-BD95118D-E7304B28      MVIV      ACTIVE
```

The command to check whether the Sysplex session-ID cache is being used by applications is **`f gsksrvr,display sidcache`**. The command output is shown in Example F-5.

Example: F-5 System SSL server display sidcache command

```
RESPONSE=MVIV
 GSK01032I Session cache status
 User              Count
 *DSPACE*           1/20
 WEBSRV                1
RESPONSE=MVIS
 GSK01032I Session cache status
 User              Count
 *DSPACE*           1/20
 WEBSRV                1
```

The messages say that there are data spaces of one megabyte allocated as session-ID caches (out of a maximum size of 20 MB). The user WEBSRV, actually our HTTP server user ID, has one entry in the caches, each entry being for a different session ID, as there is no duplication of entries among the caches.

Sysplex session-ID caching trace analysis example

We traced the System SSL activity as invoked by our HTTP server (that is, job IMWEBSRV) with a user ID of WEBSRV.

Note: Some messages were suppressed in the trace examples for the sake of clarity.

Example F-6, Example F-7, and Example F-8 on page 265 show the relevant traces for System SSL instances in the MVIS and MVIV systems.

Example: F-6 MVIV system SSL server trace messages

```
09/06/2005-09:32:59 Thd-3 INFO srv_local_service_request(): Session cache function 4 entered:
Job IMWEBSRV, User WEBSRV
09/06/2005-09:32:59 Thd-3 INFO srv_assign_sysplex_session(): Session ID: D4E5C9E5 40404040
BD91A253 E734F8F6 00000000 00000001 431D462B 00000006
09/06/2005-09:32:59 Thd-3 INFO srv_local_service_request(): Session cache function 4 exited
09/06/2005-09:32:59 Thd-3 INFO srv_local_service_request(): Session cache function 1 entered:
Job IMWEBSRV, User WEBSRV
09/06/2005-09:32:59 Thd-3 INFO srv_create_sysplex_session(): Session ID: D4E5C9E5 40404040
BD91A253 E734F8F6 00000000 00000001 431D462B 00000006
09/06/2005-09:32:59 Thd-3 INFO srv_create_sysplex_session(): 207 bytes allocated at 00002400 in
session cache data space
09/06/2005-09:32:59 Thd-3 INFO srv_local_service_request(): Session cache function 1 exited
```

The example above shows that System SSL in system MVIV being requested to store a session ID in its cache. There is a message that one entry was created in it the cache for the specific session ID.

Example F-7 shows a full handshake performed by the HTTP server on the MVIV system.

Example: F-7 MVIV http server full handshake trace messages

```
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_hello(): Received CLIENT-HELLO message
09/06/2005-09:32:59 Thd-11 ASCII read_v3_client_hello(): CLIENT-HELLO message
        00000000:  0100005d 0301431d 48d6d6fd b3ad47fd  *...]..C.H.....G.*
        00000010:  fcb840f4 b628a41b b3ebf6d4 5e8c924b  *..@..(......^..K*
        00000020:  f86babd6 d88d20d4 e5c9e240 404040bd  *.k.... ....@@@@.*
        00000030:  91a218e7 34f8bd00 00000000 00000143  *....4.........C*
        00000040:  1d44a700 00001800 16000400 05000a00  *.D.............*
        00000050:  09006400 62000300 06001300 12006301  *..d.b.........c.*
        00000060:  00                                   *.              *
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_hello(): Creating new session for connection
with 9.100.199.233[1607]
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_hello(): Using TLSV1 protocol
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_hello(): Using V3 cipher specification 09
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_hello(): Using key exchange type 1
09/06/2005-09:32:59 Thd-11 ENTRY gsk_generate_random_bytes(): ---> Length 32
09/06/2005-09:32:59 Thd-11 EXIT gsk_generate_random_bytes(): <--- Exit status 0x00000000 (0)
09/06/2005-09:32:59 Thd-11 INFO send_v3_server_messages(): Session identifier D4E5C9E5 40404040
BD91A253 E734F8F6 00000000 00000001 431D462B 00000006
09/06/2005-09:32:59 Thd-11 INFO send_v3_server_messages(): Sent SERVER-HELLO message
09/06/2005-09:32:59 Thd-11 ASCII send_v3_server_messages(): SERVER-HELLO message
        00000000:  02000046 0301431d 462bb942 be3fdccb  *...F..C.F+.B.?..*
        00000010:  2393da0a 9a7f1313 3382b6fe 4ad7e3c4  *#.......3...J...*
        00000020:  41dbf335 3ed520d4 e5c9e540 404040bd  *A..5>. ....@@@@.*
        00000030:  91a253e7 34f8f600 00000000 00000143  *..S.4.........C*
        00000040:  1d462b00 00000600 0900               *.F+.......      *
09/06/2005-09:32:59 Thd-11 ENTRY gsk_get_record_by_label(): ---> Handle 199E1450, Label 'itso
webserver'
09/06/2005-09:32:59 Thd-11 INFO crypto_des_decrypt(): Clear key DES decryption performed for
640 bytes
09/06/2005-09:32:59 Thd-11 EXIT gsk_get_record_by_label(): <--- Exit status 0x00000000 (0)
09/06/2005-09:32:59 Thd-11 INFO gsk_get_local_certificates(): Using subject record 'itso
webserver'
09/06/2005-09:32:59 Thd-11 ENTRY gsk_get_record_by_id(): ---> Handle 199E1450, ID 1
09/06/2005-09:32:59 Thd-11 EXIT gsk_get_record_by_id(): <--- Exit status 0x00000000 (0) Label
'itso racf ca'
09/06/2005-09:32:59 Thd-11 ENTRY gsk_encode_signature(): --->
09/06/2005-09:32:59 Thd-11 ASCII gsk_encode_signature(): Encoded signature stream
        00000000:  30820283 308201ec a0030201 02020101  *0...0...........*
09/06/2005-09:32:59 Thd-11 EXIT gsk_encode_signature(): <--- Exit status 0x00000000 (0)
09/06/2005-09:32:59 Thd-11 ENTRY gsk_encode_signature(): --->
09/06/2005-09:32:59 Thd-11 ASCII gsk_encode_signature(): Encoded signature stream
        00000000:  30820283 308201ec a0030201 02020100  *0...0...........*
09/06/2005-09:32:59 Thd-11 EXIT gsk_encode_signature(): <--- Exit status 0x00000000 (0)
09/06/2005-09:32:59 Thd-11 INFO send_v3_server_messages(): Sent CERTIFICATE message
09/06/2005-09:32:59 Thd-11 ASCII send_v3_server_messages(): CERTIFICATE message
        00000000:  0b000517 00051400 02873082 02833082  *..........0...0.*
09/06/2005-09:32:59 Thd-11 INFO send_v3_server_messages(): Sent SERVER-HELLO-DONE message
09/06/2005-09:32:59 Thd-11 INFO gsk_write_v3_record(): Calling write routine for 1390 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_write_v3_record(): 1390 bytes written
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 5 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 5 bytes received
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 134 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 134 bytes received
09/06/2005-09:32:59 Thd-11 INFO read_v3_client_key_exchange(): Received CLIENT-KEY-EXCHANGE
message
09/06/2005-09:32:59 Thd-11 ASCII read_v3_client_key_exchange(): CLIENT-KEY-EXCHANGE message
        00000000:  10000082 00806345 3863429b a3dab350  *......cE8cB....P*
09/06/2005-09:32:59 Thd-11 INFO crypto_rsa_private_decrypt(): Using PKCS private key
09/06/2005-09:32:59 Thd-11 INFO crypto_rsa_private_decrypt(): RSA modulus is 1024 bits
09/06/2005-09:32:59 Thd-11 INFO crypto_rsa_private_decrypt(): Software RSA private key
decryption performed
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 5 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 5 bytes received
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 1 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 1 bytes received
09/06/2005-09:32:59 Thd-11 INFO read_v3_change_cipher_specs(): Received CHANGE-CIPHER-SPECS
message
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 5 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 5 bytes received
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): Calling read routine for 40 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_read_v3_record(): 40 bytes received
```

```
09/06/2005-09:32:59 Thd-11 INFO crypto_des_decrypt_ctx(): Clear key DES decryption performed
for 40 bytes
09/06/2005-09:32:59 Thd-11 INFO read_v3_finished(): Received FINISHED message
09/06/2005-09:32:59 Thd-11 ASCII read_v3_finished(): FINISHED message
        00000000:  1400000c 6f75f361 f73828ba 702603e7  *....ou.a.8(.p&..*
09/06/2005-09:32:59 Thd-11 INFO send_v3_change_cipher_specs(): Sent CHANGE-CIPHER-SPECS message
09/06/2005-09:32:59 Thd-11 INFO gsk_write_v3_record(): Calling write routine for 6 bytes
09/06/2005-09:32:59 Thd-11 INFO gsk_write_v3_record(): 6 bytes written
09/06/2005-09:32:59 Thd-11 INFO send_v3_finished(): Sent FINISHED message
09/06/2005-09:32:59 Thd-11 ASCII send_v3_finished(): FINISHED message
        00000000:  1400000c f176b629 0c2c2e2c 3a3f176a  *.....v.).,.,:?.j*
```

In the example above is possible to match the session ID stored by System SSL with the session ID generated in the handshake. The sequence is:

1. A client-hello message is received without a session ID.
2. The server generates a new session ID and stores it in the Sysplex session ID cache.
3. The server sends a server-hello message with the new session ID.
4. The server sends the certificate message and the server-hello-done message.
5. The the server receives a client-key-exchange message and performs a decryption with his private key.
6. The server receives a change-cipher-specs message. After this message all the following messages sent will be encrypted with the generated symmetric keys.
7. The server receives a finished message, encrypted, and sends a finished message, encrypted, to the client.

Example: F-8 MVIS system ssl server trace

```
09/06/2005-09:33:00 Thd-3 INFO srv_local_service_request(): Session cache function 2 entered:
Job IMWEBSRV, User WEBSRV
09/06/2005-09:33:00 Thd-3 INFO srv_get_sysplex_session(): Session ID: D4E5C9E5 40404040
BD91A253 E734F8F6 00000000 00000001 431D462B 00000006
09/06/2005-09:33:00 Thd-3 INFO srv_get_sysplex_session(): 207 bytes returned for remote session
entry
09/06/2005-09:33:00 Thd-3 INFO srv_local_service_request(): Session cache function 2 exited
```

In Example F-8 the GSKSRVR instance in system MVIS receives a request to send the information for a designated session ID.

In Example F-9 the HTTP server performs an abbreviated handshake.

Example: F-9   MVIV http server abbreviated handshake

```
09/06/2005-09:33:00 Thd-B INFO read_v3_client_hello(): Received CLIENT-HELLO message
09/06/2005-09:33:00 Thd-B ASCII read_v3_client_hello(): CLIENT-HELLO message
        00000000:  0100005d 0301431d 48d79008 d717612c  *...]..C.H.....a,*
        00000010:  335c1ad3 82a19062 f8e048ce c9b78401  *3\.....b..H.....*
        00000020:  b528964c 2f6520d4 e5c9e540 404040bd  *.(.L/e ....@@@@.*
        00000030:  91a253e7 34f8f600 00000000 00000143  *..S.4..........C*
        00000040:  1d462b00 00000600 16000400 05000a00  *.F+.............*
        00000050:  09006400 62000300 06001300 12006301  *..d.b.........c.*
        00000060:  00                                   *.               *
09/06/2005-09:33:00 Thd-B INFO read_v3_client_hello(): Using cached session for connection with
9.100.199.233[1608]
09/06/2005-09:33:00 Thd-B INFO read_v3_client_hello(): Using TLSV1 protocol
09/06/2005-09:33:00 Thd-B INFO read_v3_client_hello(): Using V3 cipher specification 09
09/06/2005-09:33:00 Thd-B INFO read_v3_client_hello(): Using key exchange type 0
09/06/2005-09:33:00 Thd-B ENTRY gsk_generate_random_bytes(): ---> Length 32
09/06/2005-09:33:00 Thd-B EXIT gsk_generate_random_bytes(): <--- Exit status 0x00000000 (0)
09/06/2005-09:33:00 Thd-B INFO send_v3_server_messages(): Session identifier D4E5C9E5 40404040
BD91A253 E734F8F6 00000000 00000001 431D462B 00000006
09/06/2005-09:33:00 Thd-B INFO send_v3_server_messages(): Sent SERVER-HELLO message
09/06/2005-09:33:00 Thd-B ASCII send_v3_server_messages(): SERVER-HELLO message
        00000000:  02000046 0301431d 462cc68b 0f4c1c9c  *...F..C.F,...L..*
        00000010:  eb9287be 8abcacb6 a78e3966 b674d823  *..........9f.t.#*
        00000020:  f447f255 d62620d4 e5c9e540 404040bd  *.G.U.& ....@@@@.*
        00000030:  91a253e7 34f8f600 00000000 00000143  *..S.4..........C*
        00000040:  1d462b00 00000600 0900               *.F+.......      *
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): Calling write routine for 79 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): 79 bytes written
09/06/2005-09:33:00 Thd-B INFO send_v3_change_cipher_specs(): Sent CHANGE-CIPHER-SPECS message
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): Calling write routine for 6 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): 6 bytes written
09/06/2005-09:33:00 Thd-B INFO send_v3_finished(): Sent FINISHED message
09/06/2005-09:33:00 Thd-B ASCII send_v3_finished(): FINISHED message
        00000000:  1400000c 51abf9b7 5ce1767c 5e590b69  *....Q...\.v|^Y.i*
09/06/2005-09:33:00 Thd-B INFO crypto_des_encrypt_ctx(): Clear key DES encryption performed for
40 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): Calling write routine for 45 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_write_v3_record(): 45 bytes written
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): Calling read routine for 5 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): 5 bytes received
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): Calling read routine for 1 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): 1 bytes received
09/06/2005-09:33:00 Thd-B INFO read_v3_change_cipher_specs(): Received CHANGE-CIPHER-SPECS
message
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): Calling read routine for 5 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): 5 bytes received
```

```
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): Calling read routine for 40 bytes
09/06/2005-09:33:00 Thd-B INFO gsk_read_v3_record(): 40 bytes received
09/06/2005-09:33:00 Thd-B INFO crypto_des_decrypt_ctx(): Clear key DES decryption performed for
40 bytes
09/06/2005-09:33:00 Thd-B INFO read_v3_finished(): Received FINISHED message
09/06/2005-09:33:00 Thd-B ASCII read_v3_finished(): FINISHED message
        00000000:  1400000c 34f7829d 93c5ac88 2f23d136  *....4......./#.6*
```

In Example F-9 on page 266 the number of messages exchanged by the server and client in the system MVIS are fewer than in the system MVIV. The sequence now is:

1. The server receives a client-hello message containing a session ID and retrieves the information from the Sysplex session-ID cache.
2. Then the server sends a server-hello with the same session ID.
3. Now the server sends a change-cipher-spec message indicating that the following messages will be encrypted by symmetric keys.
4. Then the server sends a finished message encrypted by the same keys used previously in another system, the system MVIV.
5. Then the server receives a change-cipher-spec message.
6. And finally the server receives a finished message from the client, encrypted.

Note: The application RACF user ID is associated with the information stored in the Sysplex session-ID cache. To retrieve cache information for another user ID, one has to be permitted to the RACF profile GSK.SIDCACHE.<user> in the facility class where <user> is the owner of the session-ID cache entry.

G

VPN setup

In this appendix we give details on the implementation of the IPsec VPNs that we have been using in our z/OS Sysplex implementation. The first part deals with setting up the VPN at a single z/OS host. We also discuss the proper setup of z/OS IPSec VPNs in the several hosts of a sysplex distributor configuration.

VPN scenario

This section describes the VPN test configuration we used between one member of the sysplex and a Windows® XP workstation. We also explain the contents of the configuration files we used.

The network configuration is shown in Figure G-1. Note that, as already mentioned, only one member of the Sysplex has IP connectivity with external systems.

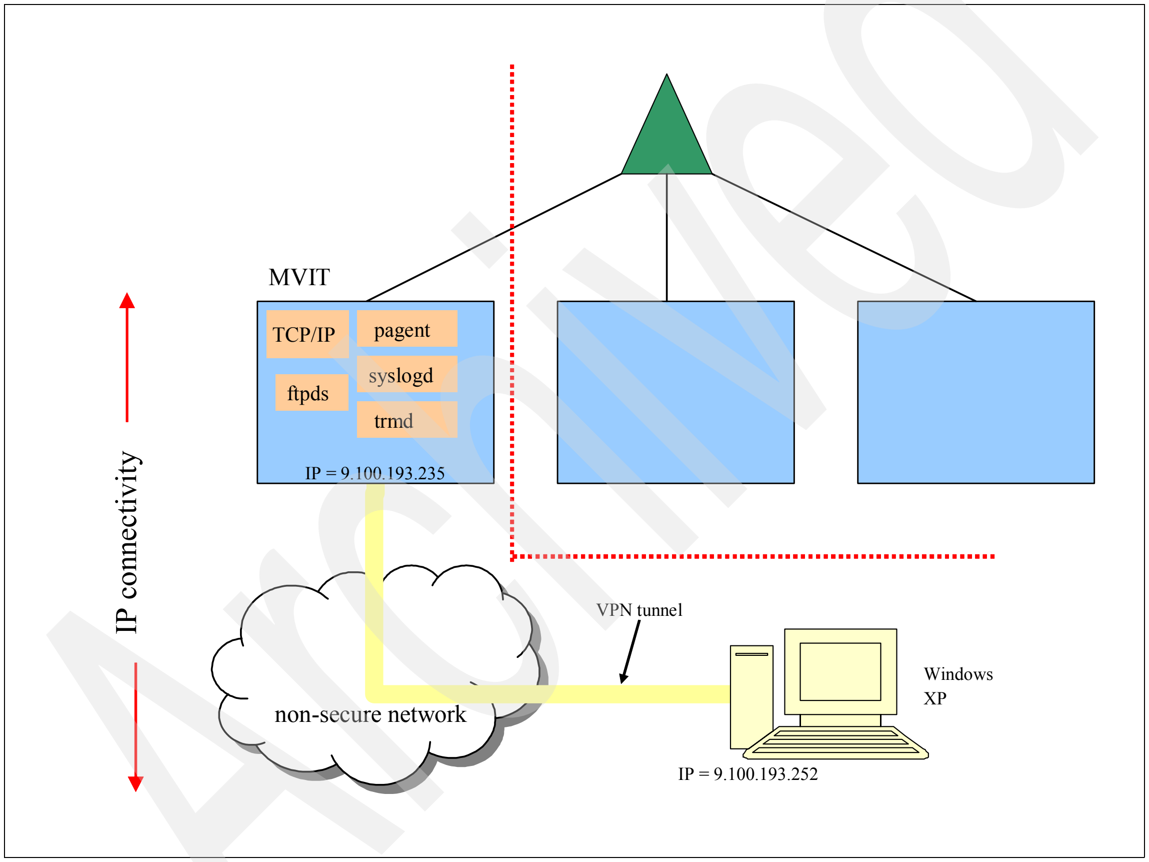


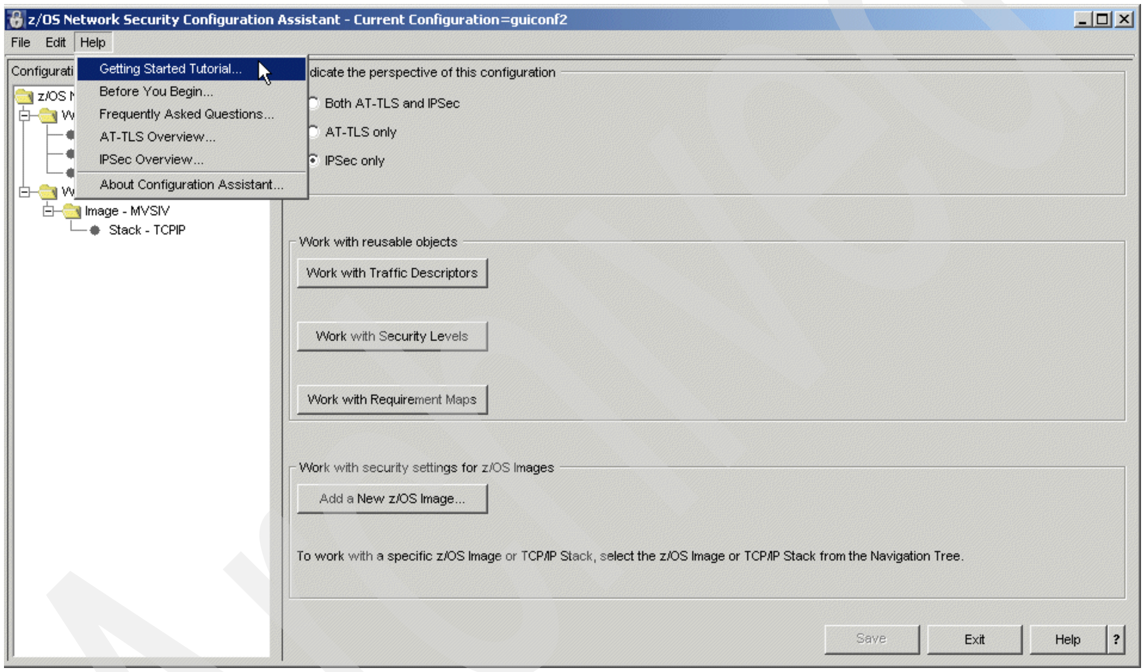
Figure G-1 Network configuration for VPN

z/OS Network Configuration Assistant

The z/OS Network Configuration Assistant is a graphical user interface (GUI) that can be used to define AT-TLS and IPsec, VPN, and IKE configuration policies. Once defined at the Configuration Assistant, the policy files and other relevant parameters or JCL files are sent to the z/OS TCP/IP host using FTP. The GUI can be downloaded from the following URL:

http://www.ibm.com/support/search.wss?tc=SSSN3L&rs=852&rank=&dc=D400&dtm

The Configuration Assistant also provides the necessary definition changes that must go into the PROFILE.TCPIP dataset, such as the IPCONFIG and IPSECURITY statements. It can also provide the JCL to be used for starting the IKE daemon. A sample of the Configuration Assistant main configuration page is shown in Figure G-2.


z/OS Network Security Configuration Assistant - Current Configuration=guiconf2
File
Edit
Help
Getting Started Tutorial...
Before You Begin...
Frequently Asked Questions...
AT-TLS Overview...
IPSec Overview...
About Configuration Assistant...
Configurati
z/OS N
Image - MVSIV
Stack - TCPIP
dicate the perspective of this configuration
Both AT-TLS and IPSec
AT-TLS only
IPSec only
Work with reusable objects
Work with Traffic Descriptors
Work with Security Levels
Work with Requirement Maps
Work with security settings for z/OS Images
Add a New z/OS Image...
Save
Exit
Help
?

Both AT-TLS and IPSec [ ] AT-TLS only [ ] [✓ IPSec only]

To work with a specific z/OS Image or TCP/IP Stack, select the z/OS Image or TCP/IP Stack from the Navigation Tree.

Figure G-2   *z/OS Network Configuration Assistant*

For inexperienced users to use the z/OS Network Security Configuration Assistant it only takes selecting the **Help** pull-down menu and then **Getting Started Tutorial**. Following this tutorial through readily allows you to configure the z/OS portion of the VPN.

Although it does seem useful to provide a detailed description of the policies generated, the usage of the Configuration Assistant is not described in this book. It may also be helpful to review these policies prior to using the GUI, since understanding the structure of policies will assist in understanding the GUI.

TCP/IP profile statements generated by the Assistant

The PROFILE.TCPIP statements generated by the z/OS Network Security Configuration Assistant can be seen in Example G-1.

Example: G-1 Profile statements generated by Configuration Assistant

```
IPCONFIG IPSECURITY
;;
IPSEC  LOGENABLE
IPSECRULE *  *  NOLOG  PROTOCOL OSPF 1
IPSECRULE *  *  NOLOG  PROTOCOL 2 2
IPSECRULE *  *  NOLOG  PROTOCOL UDP      SRCPORT *  DSTPORT 53  SECCLASS 100 3
IPSECRULE *  9.1.1.1  LOG  PROTOCOL * 4
ENDIPSEC
```

The IPSECURITY keyword is discussed in Appendix C, "IP filtering implementation and management example" on page 221.

The IPSEC keyword begins the definition of default IP filter rules, which provides the opportunity to permit IP traffic prior to the policy agent policies being read into the TCP/IP stack. Without the default permit rules shown here, all traffic is denied by the stack until the policy agent has activated its policies. The IPSEC keyword is discussed in Appendix C, "IP filtering implementation and management example" on page 221.

Default rules 1 and 2 are in place to permit OSPF traffic. If OSPF is active prior to the policy agent having read the policy files, this will allow connectivity to be established at an earlier stage.

Rule 3 is there to allow DNS traffic to flow.

The last rule, 4, was added by the Configuration Assistant unexpectedly. A comment in the profile generated stated that the rule was for administrative access. This is intended to allow IP connectivity (for example, a TN3270 session) coming from a local workstation (address 9.1.1.1). This address was automatically generated by the Configuration Assistant. There appeared to be no place within the Assistant to change this. It should be manually changed to the appropriate value for your installation.

Policies generated by the Configuration Assistant

The policies created by the z/OS Network Security Configuration Assistant can be seen in Example G-2 on page 273. These policies are to reside in a file, pointed at in the policy agent configuration file with the following statement:

```
IpsecConfig //'SYSM.TCPCFG.POLICY.VPN'
```

The data set SYSM.TCPCFG.POLICY.VPN was created by FTPing the file created by the Configuration Assistant to the z/OS host. This was done using the Configuration Assistant's Configuration Files Installation option.

Example: G-2 Polices generated

```
IpGenericFilterAction          Permit~LogYes 1
{
  IpFilterAction               Permit
  IpFilterLogging              Yes
}

IpGenericFilterAction          IpSec~LogYes 2
{
  IpFilterAction               IpSec
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~1 3
{
  HowToEncrypt                 DES
  HowToAuthMsgs                MD5
  HowToAuthPeers               RsaSignature
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      240 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R 4
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      14 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~2 5
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      14 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted       None
}

IpService                       IKE~Gen 6
{
  Protocol                      UDP
  SourcePortRange               500
  DestinationPortRange          500
  Direction                     BiDirectional
  Routing                       Local
}

IpService                       All_other_traffic 7
{
  Protocol                      All
  Direction                     BiDirectional
  Routing                       Local
}

IpDynVpnAction                  IPSec__Silver 8
{
  Initiation                    Either
  VpnLife                       1440
  Pfs                           None
  IpDataOfferRef                IPSec__Silver~R
  IpDataOfferRef                IPSec__Silver~R~2
}

IpAddr                          O~ADR~1 9
{
  Addr                          9.100.193.235
}

IpAddr                          O~ADR~2
{
  Addr                          9.100.199.252
}

LocalSecurityEndpoint           O~LSE~4 10
{
  Identity                      X500dn CN=ITSO,OU=Montpellier,O=IBM,C=France
  LocationRef                   O~ADR~1
}

RemoteSecurityEndpoint          O~RSE~3 11
{
  Identity                      X500dn CN=ITSO,OU=Montpellier,O=IBM,C=France
  LocationRef                   O~ADR~2
}
```

```
KeyExchangeRule              0~5 12
{
  LocalSecurityEndpointRef   0~LSE~4
  RemoteSecurityEndpointRef  0~RSE~3
  KeyExchangeActionRef       0
}

KeyExchangeAction            0 13
{
  HowToInitiate              Main
  HowToRespond               Either
  KeyExchangeOfferRef        KEO~1
  AllowNat                   No
}

IpFilterRule                 0~6 14
{
  IpSourceAddrRef            0~ADR~1
  IpDestAddrRef              0~ADR~2
  IpServiceRef               IKE~Gen
  IpGenericFilterActionRef   Permit~LogYes
}

IpFilterRule                 0~7 15
{
  IpSourceAddrRef            0~ADR~1
  IpDestAddrRef              0~ADR~2
  IpServiceRef               All_other_traffic
  IpGenericFilterActionRef   IpSec~LogYes
  IpDynVpnActionRef          IPSec__Silver
}

KeyExchangePolicy 16
{
  AllowNat                   No
  KeyExchangeRuleRef         0~5
}

IpFilterPolicy 17
{
  PreDecap                   OFF
  FilterLogging              ON
  IpFilterLogImplicit        No
  AllowOnDemand              Yes
  IpFilterRuleRef            0~6
  IpFilterRuleRef            0~7
}
```

When reading policy files, it helps if someone is familiar with a programming language. This is because the filter policies are built as structures that can be referenced by other structures, exactly analogous to the way data areas would point to other data areas in a program. Or, to put it into more modern terminology, policy objects can reference other policy objects.

Inherent in the rules discussed below is the implicit rule that is always installed by the policy agent, which is to deny any traffic and which is located at the end of the user filter policy. If no matching rule is found in the user-designed filter, then the packet is denied by this default rule.

1 This statement is used to define an action of permitting a packet and logging the event. This statement exists so that it can be re-used later within a rule.

2 This statement is the same as **1** above except that it requires that the packet be protected within an IPsec VPN tunnel.

3 A VPN requires a symmetric key for encrypting data that flows inside the tunnel. This statement defines the security parameters associated with the exchange of symmetric keys. DES encryption is used along with MD5 for message authentication. RSASignature is specified since this is a dynamic tunnel using certificates to authenticate both ends key servers. These parameters are used by the IKE daemon.

4 Similar to the statement in 3 above, this policy controls security parameters that relate to the data travelling over the VPN itself. The Configuration Assistant has several predefined customer security proposals already built in. This proposal is referred to in the Configuration Assistant as IPSec_Silver.

Note that the RefreshLifetimeAccepted has been set to between 14 and 480 minutes. The Windows XP client was found to have a default of only 900 seconds (15 minutes) for its Session Key settings. The Configuration Assistant chose a range of 120 to 480 minutes. This policy was manually edited after exporting it from the Configuration assistant. This is the only alteration made to the originally created file.

The encapsulation mode chosen is transport, with ESP as the desired security method.

5 This is the second choice offer, behind **4** being the first choice, for security parameters for a data. The encryption for this offer is Triple-DES (3DES).

6 Because the implicit rule is to deny everything, there needs to be a service definition that describes IKE traffic necessary for dynamic tunnel negotiation that will be eventually permitted. ISAKMP/IKE uses UDP datagrams via port 500 for negotiating tunnel characteristics.

7 This service describes the overall traffic that will be, in our case, directed through the VPN. For finer control the Configuration Assistant contains built-in definitions of most well-know traffic types based upon well-known port usage.

8 This statement defines the parameters to be taken into account when establishing a dynamic VPN. Either endpoint can initiate the VPN, with the first security characteristics being offered are those described in the IPSec_Silver~1 statement, the second choice being IPSec__Silver~R~2.

9 These two IpAddr statements were automatically generated as *variables* containing the z/OS host IP address (9.100.193.235) and the XP workstation address (9.100.199.252).

10 The VPN requires a method for establishing the identity of an endpoint. There are four different identity specifications available from the Configuration Assistant: IP address, Fully Qualified Domain Name, e-mail address, or X.500 distinguished name. Here we state that the endpoint distinguished name, as specified in the certificate generated in RACF in "Setting up the digital certificates" on page 279, is to be used to identify the endpoint.

The LocationRef points to the IpAddr defined as 0~ADR~1, the local host's IP address.

11 Because we used, for the sake of simplicity, the same certificate at the XP workstation, the identity of the RemoteSecurityEndpoint is the same. Of course, 0~ADR~2 is the workstation's IP address.

12 Things are starting to be put together now. This rule encompasses the two endpoints from **10** and **11** and specifies that the key exchange action identified in **13** be used between these two endpoints. The forward reference was created as the Configuration Assistant built the policy file.

13 This is the action pointed to by **12** above. Specifying *Main* for the HowToInitiate parameter means that the IKE protocol sequence for identity exchange occurs using the so-called *Main Mode*. The other possibility would be to specify the Aggressive Mode. However, Main is providing better protection for the exchanged identities.

Because no Network Address Translation (NAT) firewalls exist between the endpoints, NAT traversal (AllowNat) was set to No.

14 and **15** specify the rules that are to control the exchanges between the two endpoints we defined. These statements refer to previously defined statements that contain the other parameters to take into account. The statement **14** specifies the rule to allow the ISAKMP/IKE exchange, while **15** indicates the rule for the data traffic itself.

16 This is the statement that ultimately controls the behavior of the IKE daemon. It references the KeyExchangeRule from **12**.

17 This is the global policy that identifies the operation environment of the IP filtering mechanism. Turning off PreDecap in most cases will improve throughput by avoiding filtering before the packets are decapsulated. Filter logging is set on, but implicit filter logging (that is, logging of the implicit deny rules) is turned off.

AllowOnDemand means that if a packet matches the rule for the VPN filter and the tunnel does not already exist, z/OS IP Security Services will attempt to establish the tunnel immediately.

Setup of the IKE daemon

The _CEE_ENVFILE environment variable was used to set up a STDENV DD card for controlling the environment variables. And MVS data sets were used for all files. A relevant sample of the environment file can be seen in Example G-3.

Example: G-3 IKED environment file

```
IKED_FILE=//'SYSM.TCPCFG.TCPPARMS(IKECON)'
IKED_CTRACE_MEMBER=CTIIKE01
```

There is no need to specify a resolver configuration file here because, as it is the case with the policy agent, only one server should be used for all stacks within the image.

The data set member pointed to by IKED_FILE was created by the z/OS network configuration assistant. All of these configuration statements are explained using the field sensitive help within the GUI.

However, it is appropriate to provide a few comments on some of the statements generated in the IKE daemon configuration file shown in Example G-4 on page 279.

- IkeSyslogLevel and PagentSyslogLevel

  These levels were set to 127 during testing to obtain helpful messages. After successful configuration, a much lower value should be used to avoid over-filling the syslogd file.

- key ring

  This is the RACF key ring name used to hold the certificate authority certificate required for ISAKMP/IKE authentication. This key ring was created as owned by the user ID of the IKED started task, IKED. The key ring could have been owned by another user ID, then this user ID would have then needed to be pre-pended in the statement to the key ring name using the syntax <owning_userID>/<key_ring_name>. Note, however, that in order to access a key ring that is owned by another user ID requires the accessor to have UPDATE access to IRR.DIGTCERT.LISTRING.

Example: G-4 IKE daemon configuration file

```
IkeConfig
{
IkeSyslogLevel 127
PagentSyslogLevel 127
key ring VPNRing
KeyRetries 10
KeyWait 30
DataRetries 10
DataWait 15
Echo No
PagentWait 0
}
```

The IKED_CTRACE_MEMBER shown in Example G-4 points to the parameter file for IKE daemon CTRACE parameters, and should only be used under the instructions of IBM support.

Setting up the digital certificates

The next step is to establish a certificate environment for the key exchange. Both the z/OS host and the Windows XP host must have valid certificates configured in order for the ISAKMP/IKE exchange to eventually establish a security association.

A certificate authority (CERTAUTH) certificate was created in RACF, and then this certificate was used to sign a personal certificate. Then a key ring was created and both certificates were connected. The personal certificate was connected as the default. The commands, in the sequence they were entered, were as follows:

1. `racdcert certauth gencert SUBJECTSDN(OU('Montpellier') O('IBM') C('France')) keyusage(certsign) withlabel('VPNCertAuth')`
2. `racdcert id(iked) gencert subjectsdn(CN('ITSO') OU('Montpellier') O('IBM') C('France')) withlabel('VPNCert') signwith(CERTAUTH LABEL('VPNCertAuth'))`
3. `racdcert id(iked) addring(VPNRing)`

4. `racdcert id(iked) connect(certauth Label('VPNCertAuth') ring(VPNRing) usage(certauth))`
5. `racdcert id(iked) connect(Label('VPNCert') ring(VPNRing) usage(personal))`

In order to keep the configuration scenario simple, the same certificate authority and personal certificate were used on the XP workstation (something you will not be doing in a real-world configuration). To do this, the certificates need to be exported from the RACF database into MVS data sets using the following commands:

- `racdcert certauth export(label('VPNCertAuth')) dsn('matt.vpn.certauth') format(pkcs7der)`
- `racdcert id(iked) export(label('VPNCert')) dsn('MATT.VPNCERT.PKCS12D') format(pkcs12der) password('matt')`

Note that the certificate authority certificate does not require coming with a private key. However, the personal certificate that we are going to use in our Windows XP client should include the private key, and hence a PKCS12DER format file, along with a protection password, was used.

Windows XP setup

To summarize the current situation, the policy agent and IKE daemons are both 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 on demand IPsec VPN. A set of certificates have been created and IKE is to access the key ring that contains those certificates in the RACF database. What we are describing now is the equivalent setup for our test Windows XP workstation.

Certificate and VPN management

The first step required for proper certificate setup on Windows XP is to physically place the two certificates exported in “Setting up the digital certificates” on page 279 onto a disk of the XP workstation. FTP was used in our scenario. Because the certificates were both created using a binary format, a binary mode (image mode) FTP transfer was used.

To import a certificate into the XP certificate database, you will need to run a command called MMC, which stands for Microsoft® Management Console. From within MMC, three snap-ins should be added:

- Certificates (local computer)
- IP Security Policies on local computer

- IP Security Monitor (not required, but of some interest)

As with the z/OS Configuration Assistant, the snap-ins are GUI interfaces and are reasonably self-explanatory. Detailed, step-by-step instructions can be found in appendix B of the IBM Redbook *Implementing VPNs in a z/OS Environment*, SG24-6530. These instructions, written for Windows 2000, still apply to the Windows XP environment. Brief guidelines are provided below.

For the certificates:

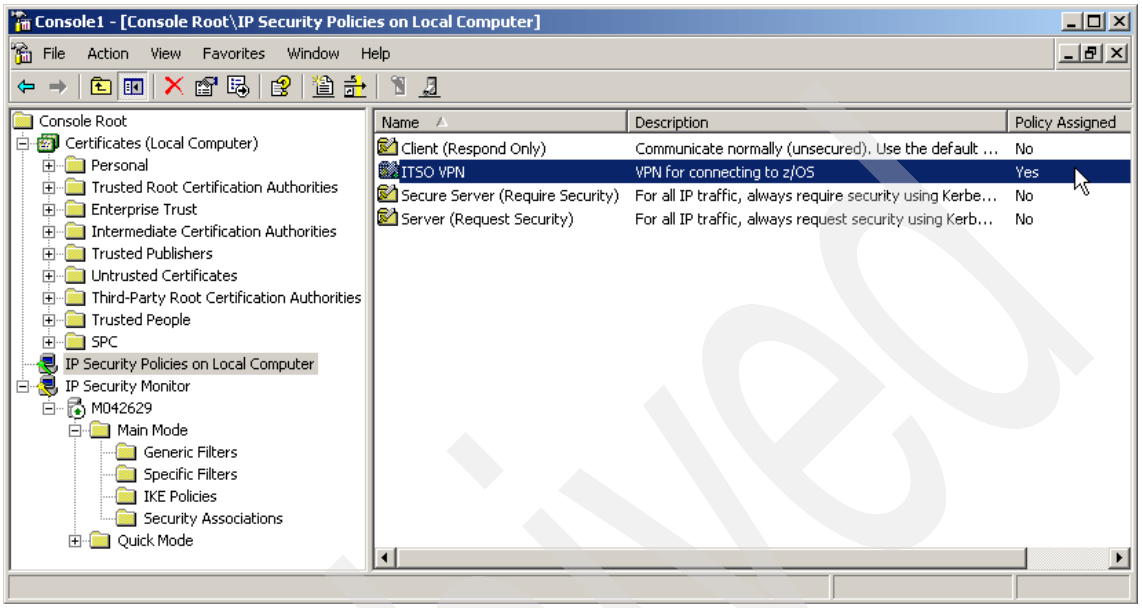
1. Expand the Certificates (Local Computer) snap-in. Right-click Trusted Root Certification Authorities to get to **All Tasks** → **Import** the signing (PKCS7DER format) certificate.
2. Right-click **Personal** to import the personal (PKCS12DER formatl) certificate.

For the VPN policy configuration:

1. Right-click **IP Security Policies on Local Computer** and select **Create IP Security Policy**. The configuration details are all exactly analogous to those provided in the description for configuration the z/OS policy files.
2. Configure the security method preferences in the order desired. For our testing, AH of <none> was used with DES being the top two encryption algorithms.
3. Once the policy is in place it needs to be activated (that is, *Assigned*) in the MMC terminology, right-clicking while viewing the policies in the right-hand window pane of the MMC console.

Note: Within the Rule Properties, there is a tab called Tunnel Setting. Contrary to what you might think, select **This rule does not specify an IPSec tunnel** when creating your policies on the workstation.

At this point, the MMC console should look something like what is shown in Figure G-3.


Console1 - [Console Root\IP Security Policies on Local Computer]
File
Action
View
Favorites
Window
Help

- Console Root
  - Certificates (Local Computer)
    - Personal
    - Trusted Root Certification Authorities
    - Enterprise Trust
    - Intermediate Certification Authorities
    - Trusted Publishers
    - Untrusted Certificates
    - Third-Party Root Certification Authorities
    - Trusted People
    - SPC
  - IP Security Policies on Local Computer
  - IP Security Monitor
    - M042629
      - Main Mode
        - Generic Filters
        - Specific Filters
        - IKE Policies
        - Security Associations
      - Quick Mode

Name	Description	Policy Assigned
Client (Respond Only)	Communicate normally (unsecured). Use the default ...	No
ITSO VPN	VPN for connecting to z/OS	Yes
Secure Server (Require Security)	For all IP traffic, always require security using Kerbe...	No
Server (Request Security)	For all IP traffic, always request security using Kerb...	No

Figure G-3 Microsoft Management Console

In the left pane of the window, the three snap-ins are visible.

The VPN policy used for testing was called ITSO VPN, which is visible in the right-hand pane. It is active as Policy Assigned is Yes.

Starting the VPN at the z/OS end

The first thing to accomplish is to ensure that the policy agent has read in the current policy. The following command was used, 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 policy was read, then a message such as follows should be seen in response:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : NONE
```

If policy changes have been made, then instead a message such as what follows should be seen:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : IPSEC
```

Next, try opening a command prompt on the XP workstation and attempt to FTP to the z/OS host. That should automatically trigger the creation of a VPN.

Determining what went wrong

The syslogd daemon is being used in z/OS to collect the messages issued by the policy agent and the IKE daemons.

If the connection failed have a look at the syslogd file for informational messages. Specifically, the message number of interest should be EZD1093I.

Our first attempt at activation produced the following message:

```
EZD1093I Policy mismatch : IpDataOffer ( 1 ) requires parameter (
RefreshLifetimeAccepted ) with value ( 120-480 minutes ) but proposal ( 1 )
value is ( 15 minutes )
```

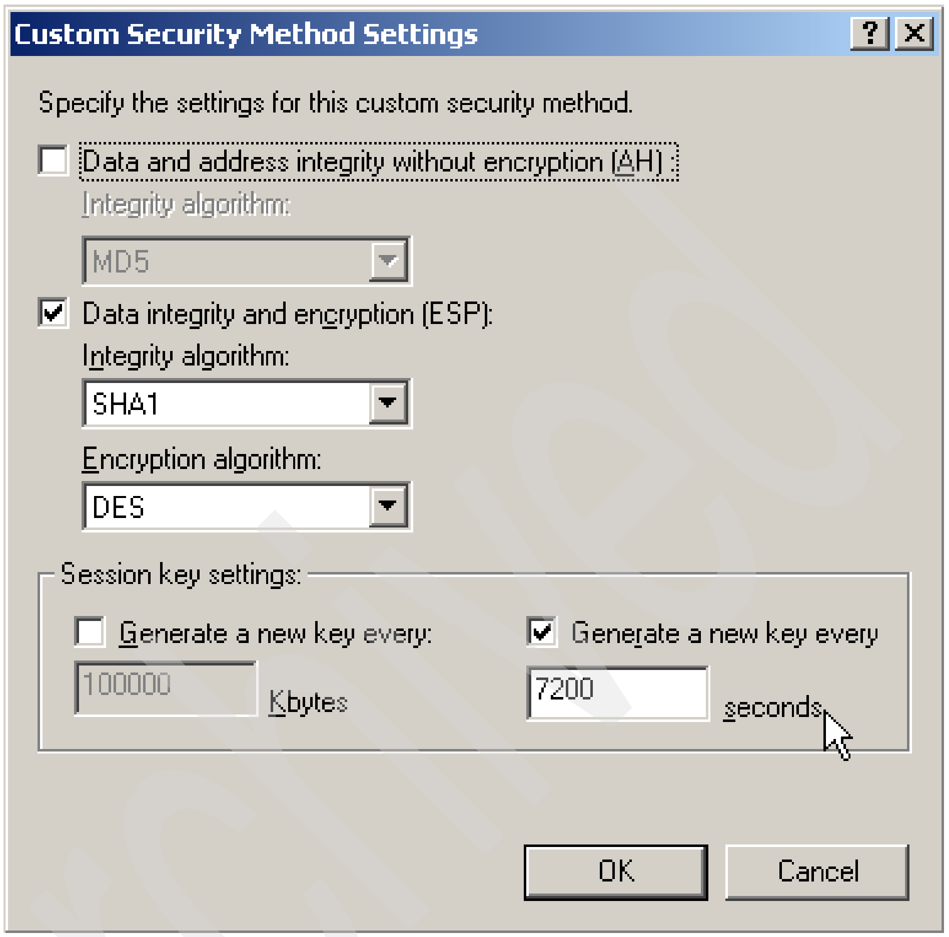
This error was alluded to earlier in "Policies generated by the Configuration Assistant" on page 272. It can be fixed by setting a low enough value on the RefreshLifetimeProposed statement for the IPDataOffer policy. You will then need to refresh the modified policy at the TCP/IP stack with the command:

```
MODIFY PAGENT,REFRESH
```

Alternatively, edit the Rule Properties at the XP workstation, and then in turn edit the Require Security filter action. Finally, by choosing to edit the Security Methods, you are presented with a Custom Security Method Settings panel. Within this window, change the default of 900 seconds (15 minutes) to something larger. A sample of this page can be seen in Figure G-3 on page 282.

If no obvious errors are showing up in syslogd, it might be worthwhile to try temporarily changing IpFilterLogImplicit from No to Yes. This statement controls whether packets denied by the default deny all rule are logged. By changing this value to Yes, you can quickly discover if a packet is not matching your filtering rules and is being denied by the default rule.

It is probably a good idea to turn this back to No once things are working correctly. The deny all rule is unlikely to be something that needs to be logged under normal conditions, and the volume of messages could impact syslogd.


Custom Security Method Settings

Specify the settings for this custom security method.

[ ] Data and address integrity without encryption (AH) :

Integrity algorithm: MD5

[✓] Data integrity and encryption (ESP):

Integrity algorithm: SHA1

Encryption algorithm: DES

Session key settings:

[ ] Generate a new key every: 100000 Kbytes

[✓] Generate a new key every 7200 seconds

Figure G-4 Microsoft Management Console, increasing refresh lifetime

To verify the active dynamic VPN policies, try the following UNIX system services command in the z/OS UNIX shell:

```
ipsec -p tcpip -f display -a Y0
```

The first parameter, -p, allows specific stacks policies to be queried. The second parameter indicates that filters are to be displayed, while the third parameter indicates a list of all dynamic anchor filters. The display output should look something like what is shown in Example G-5.

Example: G-5 ipsec command dynamic anchor display

```
CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 18:09:32 2005
Primary:  Filter         Function: Display          Format:   Detail
Source:   Stack Policy   Scope:    Current          TotAvail: 8
Logging:  Yes            Predecap: No               DVIPSec:  No
NatKeepAlive:  20

FilterName:                    0~7
FilterNameExtension:           1
GroupName:                     n/a
LocalStartActionName:          n/a
VpnActionName:                 IPSec__Silver
TunnelID:                      Y0
Type:                          Dynamic Anchor
State:                         Active
Action:                        Permit
Scope:                         Local
Direction:                     Outbound
OnDemand:                      Yes
SecurityClass:                 0
Logging:                       All
Protocol:                      All
ICMPType:                      n/a
ICMPCode:                      n/a
OSPFType:                      n/a
TCPQualifier:                  n/a
ProtocolGranularity:           Rule
SourceAddress:                 9.100.193.235
SourceAddressPrefix:           n/a
SourceAddressRange:            n/a
SourceAddressGranularity:      Packet
SourcePort:                    All
SourcePortRange:               n/a
SourcePortGranularity:         Rule
DestAddress:                   9.100.199.162
DestAddressPrefix:             n/a
DestAddressRange:              n/a
DestAddressGranularity:        Packet
DestPort:                      All
DestPortRange:                 n/a
DestPortGranularity:           Rule
OrigRmtConnPort:               n/a
RmtIDPayload:                  n/a
```

```
***********************************************************************
FilterName:                   0~7
FilterNameExtension:          2
GroupName:                    n/a
LocalStartActionName:         n/a
VpnActionName:                IPSec__Silver
TunnelID:                     Y0
Type:                         Dynamic Anchor
State:                        Active
Action:                       Permit
Scope:                        Local
Direction:                    Inbound
OnDemand:                     Yes
SecurityClass:                0
Logging:                      All
Protocol:                     All
ICMPType:                     n/a
ICMPCode:                     n/a
OSPFType:                     n/a
TCPQualifier:                 n/a
ProtocolGranularity:          Rule
SourceAddress:                9.100.199.162
SourceAddressPrefix:          n/a
SourceAddressRange:           n/a
SourceAddressGranularity:     Packet
SourcePort:                   All
SourcePortRange:              n/a
SourcePortGranularity:        Rule
DestAddress:                  9.100.193.235
DestAddressPrefix:            n/a
DestAddressRange:             n/a
DestAddressGranularity:       Packet
DestPort:                     All
DestPortRange:                n/a
DestPortGranularity:          Rule
OrigRmtConnPort:              n/a
RmtIDPayload:                 n/a
***********************************************************************

2 entries selected
```

The first thing to notice in this output is that filter 0~7 is listed twice. The rule has been automatically 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 on the VPN's policies. More complete information can be displayed using the following command.

```
pasearch -p tcpip -s DynamicVpn
```

The second parameter restricts display output to policies relating to dynamic VPN.

Proving things are working

If the FTP command at the workstation completed successfully, scan syslogd for a message similar to the following:

```
EZD1016I Phase 2 security association 22 created - LocalIp : 9.100.193.235
RemoteIp: 9.100.199.162 LocalDataPort : ALL RemoteDataPort : ALL Protocol : any
(0) HowToEncap : TRANSPORT HowToEncrypt : DES_CBC_8 HowToAuth : HMAC_SHA ( ESP
) RefreshLifetime :7200 RefreshLifesize : NONE VpnLife : 86400 PFS : NONE
```

Alternatively, the following command can be used to display the active tunnel:

```
ipsec -p tcpip -y display
```

Output from this command can be seen in Example G-6.

Example: G-6 ipsec active tunnel display

```
CS V1R7 ipsec  TCPIP Name: TCPIP  Tue Aug 30 18:20:24 2005
Primary:  Dynamic tunnel  Function: Display         Format:   Detail
Source:   Stack           Scope:    Current         TotAvail: 1

TunnelID                     Y24
VpnActionName:               IPSec__Silver
State:                       Active
LocalEndPoint:               9.100.193.235
RemoteEndPoint:              9.100.199.162
HowtoEncap:                  Transport
HowToAuth:                   ESP
 AuthAlgorithm:              Hmac_Sha
 AuthInboundSpi:             564369184
 AuthOutboundSpi:            3882794343
HowToEncrypt:                DES
 EncryptInboundSpi:          564369184
 EncryptOutboundSpi:         3882794343
OutboundPackets:             4
OutboundBytes:               223
InboundPackets:              5
InboundBytes:                115
Lifesize:                    0K
LifesizeRefresh:             0K
CurrentByteCount:            0b
LifetimeRefresh:             2005/08/30 20:01:05
```

```
LifetimeExpires:              2005/08/30 20:20:14
CurrentTime:                  2005/08/30 18:20:24
VPNLifeExpires:               2005/08/31 18:20:14
ParentIKETunnelID:            K21
LocalDynVpnRule:              n/a
NAT Traversal Topology:
  UdpEncapMode:               No
  LclNATDetected:             No
  RmtNATDetected:             No
  RmtIsGw:                    No
  RmtIsZOS:                   No
  zOSCanInitP2SA:             Yes
  SrcNATOARcvd:               n/a
  DstNATOARcvd:               n/a
***********************************************************************

1 entries selected
```

After we know that the VPN is operating

The trace levels used at this stage produce too much output for long-term or production usage. The first step is to change the IKE configuration file to lower log levels. For IkeSyslogLevel, level 1 is appropriate. For PagentSyslogLevel, a value of 7 was chosen. However, individual organizations should choose values appropriate for their needs.

After making this change, the following command will cause IKE to reread its configuration file:

```
MODIFY IKED,REFRESH,FILE=//'SYSM.TCPCFG.TCPPARMS(IKECON)'
```

The same goes for policy agent logging, but this can be accomplished in one step using the following command:

```
MODIFY PAGENT,LOGLEVEL,LEVEL=31
```

A level of 31, the default, was chosen.

Finally, it is not unreasonable to consider turning off logging for any permitted packets within the VPN. This is done by changing the IPFilterLogging from yes to no.

VPN and filters in a Sysplex Distributor configuration

Now that the VPN has been configured and tested using a single image, the next step is to test it in a sysplex distributor environment. This will provide an opportunity to see Sysplex Wide Security Association (SWSA) in action. The scenario is illustrated in Figure G-5.

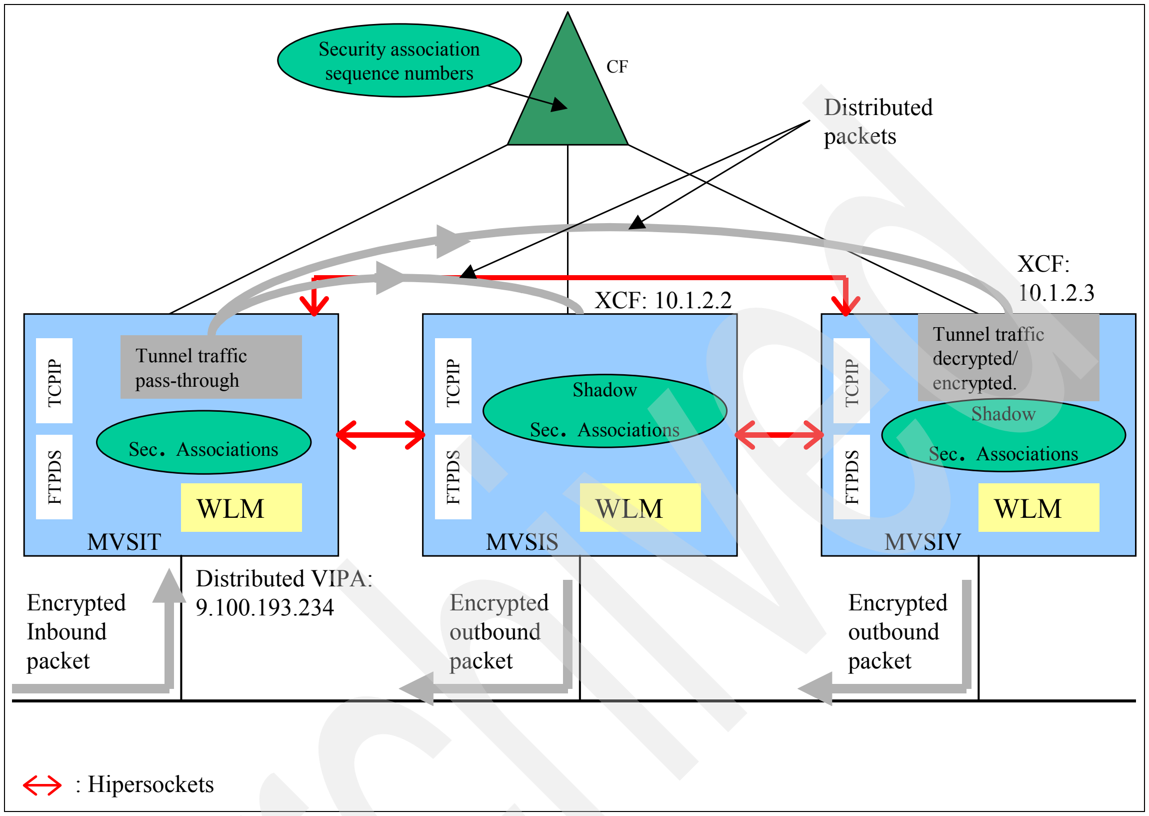

Security association sequence numbers
CF
Distributed packets
XCF: 10.1.2.2
XCF: 10.1.2.3
TCPIP
FTPDS
Tunnel traffic pass-through
Sec. Associations
WLM
MVSIT
Shadow Sec. Associations
MVSIS
Tunnel traffic decrypted/ encrypted.
MVSIV
Distributed VIPA: 9.100.193.234
Encrypted Inbound packet
Encrypted outbound packet
Encrypted outbound packet
: Hipersockets

Figure G-5 VPN Sysplex-wide security associations

z/OS setup

In addition, it was decided that this would be a good opportunity to test the filtering rules established in Appendix C, "IP filtering implementation and management example" on page 221. The combined rules can be seen in Example G-7.

This policy was activated on all three members of the sysplex.

Example: G-7 Sysplex distributor with VPN and filtering together

```
IpFilterPolicy
 {
  PreDecap                     OFF 1
  AllowOnDemand                Yes 2
```

```
   FilterLogging       On
   IpFilterLogImplicit Yes
   IpFilterGroupRef    MVI-All
 }
IpFilterGroup MVI-All
 {
   IpFilterRuleRef              0~6 3
   IpFilterRuleRef              0~7 4
   IpFilterRuleRef TN3270-7443-Local
   IpFilterRuleRef TN3270-7443-Sysplex
   IpFilterRuleRef Https-Local
   IpFilterRuleRef Https-Sysplex
   IpFilterRuleRef FTP235-Control-Local
   IpFilterRuleRef FTP235-Data-Local
   IpFilterRuleRef FTP234-Control-Dist
   IpFilterRuleRef FTP234-Data-Dist
   IpFilterRuleRef VipaRoute-GRE
   IpFilterRuleRef Tn-Sysplex-Dist-Xcf
   IpFilterRuleRef Tn-Sysplex-Target-Xcf
   IpFilterRuleRef Https-Sysplex-Dist-Xcf
   IpFilterRuleRef Https-Sysplex-Target-Xcf
   IpFilterRuleRef Ftp-Sysplex-Dist-Xcf
   IpFilterRuleRef Ftp-Sysplex-Target-Xcf
   IpFilterRuleRef Inbound-Icmp
   IpFilterRuleRef Nolog-Udp
 }
IpGenericFilterAction Permit-Log
 {
   IpFilterAction  Permit
   IpFilterLogging Yes
 }
IpGenericFilterAction Permit-NoLog
 {
   IpFilterAction  Permit
   IpFilterLogging No
 }
IpGenericFilterAction Deny-Log
 {
   IpFilterAction  Deny
   IpFilterLogging Yes
 }
IpGenericFilterAction Deny-NoLog
 {
   IpFilterAction  Deny
   IpFilterLogging No
 }
IpFilterRule TN3270-7443-Local
 {
   IpSourceAddr       9.100.193.235
```

```
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         7443
         DestinationPortRange    1024 65535
         Direction               BiDirectional InboundConnect
         Routing                 Local
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
   }
   IpFilterRule TN3270-7443-Sysplex
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         7443
         DestinationPortRange    1024 65535
         Direction               BiDirectional InboundConnect
         Routing                 Either
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
   }
   IpFilterRule Https-Local
    {
      IpSourceAddr       9.100.193.235
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
         SourcePortRange         443
         DestinationPortRange    1024 65535
         Direction               BiDirectional InboundConnect
         Routing                 Local
         SecurityClass           1
       }
      IpGenericFilterActionRef   Permit-Log
   }
   IpFilterRule Https-Sysplex
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
       {
         Protocol                Tcp
```

```
          SourcePortRange         443
          DestinationPortRange    1024 65535
          Direction               BiDirectional InboundConnect
          Routing                 Either
          SecurityClass           1
        }
      IpGenericFilterActionRef   Permit-Log
    }
  IpFilterRule FTP235-Control-Local
    {
      IpSourceAddr       9.100.193.235
      IpDestAddr         9.100.199.0/24
      IpService
        {
          Protocol                Tcp
          SourcePortRange         21
          DestinationPortRange    1024 65535
          Direction               BiDirectional InboundConnect
          Routing                 Local
          SecurityClass           1
        }
      IpGenericFilterActionRef   Permit-Log
    }
  IpFilterRule FTP235-Data-Local
    {
      IpSourceAddr       9.100.193.235
      IpDestAddr         9.100.199.0/24
      IpService
        {
          Protocol                Tcp
          SourcePortRange         20
          DestinationPortRange    1024 65535
          Direction               BiDirectional OutboundConnect
          Routing                 Local
          SecurityClass           1
        }
      IpGenericFilterActionRef   Permit-Log
    }
  IpFilterRule FTP234-Control-Dist
    {
      IpSourceAddr       9.100.193.234
      IpDestAddr         9.100.199.0/24
      IpService
        {
          Protocol                Tcp
          SourcePortRange         21
          DestinationPortRange    1024 65535
          Direction               BiDirectional InboundConnect
          Routing                 Either
      SecurityClass           1
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule FTP234-Data-Dist
  {
    IpSourceAddr       9.100.193.234
    IpDestAddr         9.100.199.0/24
    IpService
     {
      Protocol                Tcp
      SourcePortRange         20
      DestinationPortRange    1024 65535
      Direction               BiDirectional OutboundConnect
      Routing                 Either
      SecurityClass           1
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule VipaRoute-GRE
  {
    IpSourceAddr       10.1.0.0/16
    IpDestAddr         10.1.0.0/16
    IpService
     {
      Protocol                47
      Direction               BiDirectional
      Routing                 Either
      SecurityClass           2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Https-Sysplex-Dist-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
      Protocol                Tcp
      SourcePortRange         1024 65535
      DestinationPortRange    443
      Direction               Outbound
      Routing                 Routed
      SecurityClass           2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Https-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     443
       Direction                Inbound
       Routing                  Local
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Tn-Sysplex-Dist-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     7443
       Direction                Outbound
       Routing                  Routed
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Tn-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
       Protocol                 Tcp
       SourcePortRange          1024 65535
       DestinationPortRange     7443
       Direction                Inbound
       Routing                  Local
       SecurityClass            2
     }
    IpGenericFilterActionRef   Permit-Log
  }
 IpFilterRule Ftp-Sysplex-Dist-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
     {
        Protocol                Tcp
        SourcePortRange         1024 65535
        DestinationPortRange    20 21
        Direction               Outbound
        Routing                 Routed
        SecurityClass           2
      }
    IpGenericFilterActionRef   Permit-Log
  }
IpFilterRule Ftp-Sysplex-Target-Xcf
  {
    IpSourceAddr       9.100.199.0/24
    IpDestAddr         9.100.193.234
    IpService
      {
        Protocol                Tcp
        SourcePortRange         1024 65535
        DestinationPortRange    20 21
        Direction               Inbound
        Routing                 Local
        SecurityClass           2
      }
    IpGenericFilterActionRef   Permit-Log
  }
IpFilterRule Nolog-Udp
  {
    IpSourceAddr       All
    IpDestAddr         All
    IpService
      {
        Protocol                Udp
        SourcePortRange         0
        Direction               Inbound
        Routing                 Local
        SecurityClass           1
      }
    IpGenericFilterActionRef   Deny-NoLog
  }
IpFilterRule Inbound-Icmp
  {
    IpSourceAddr       All
    IpDestAddr         All
    IpService
      {
        Protocol                Icmp
        Type                    Any
        Direction               Inbound
        Routing                 Local
        SecurityClass           1
    }
   IpGenericFilterActionRef   Deny-NoLog
 }
IpFilterRule Outbound-Icmp
 {
   IpSourceAddr       9.100.193.0/24
   IpDestAddr         9.100.0.0/16
   IpService
    {
      Protocol                Icmp
      Type                    Any
      Direction               Outbound
      Routing                 Local
      SecurityClass           1
    }
   IpGenericFilterActionRef   Permit-Log
 }
## VPN policy information 5
IpGenericFilterAction          Permit~LogYes
{
  IpFilterAction               Permit
  IpFilterLogging              Yes
}

IpGenericFilterAction          IpSec~LogYes
{
  IpFilterAction               IpSec
  IpFilterLogging              Yes
}

KeyExchangeOffer               KEO~1
{
  HowToEncrypt                 DES
  HowToAuthMsgs                MD5
  HowToAuthPeers               RsaSignature
  DHGroup                      Group1
  RefreshLifetimeProposed      480
  RefreshLifetimeAccepted      14 1440
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R
{
  HowToEncap                   Transport
  HowToEncrypt                 DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      14 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

IpDataOffer                    IPSec__Silver~R~2
{
  HowToEncap                   Transport
  HowToEncrypt                 3DES
  HowToAuth                    ESP Hmac_Sha
  RefreshLifetimeProposed      240
  RefreshLifetimeAccepted      14 480
  RefreshLifesizeProposed      None
  RefreshLifesizeAccepted      None
}

## NOTE -- Generated IpService IKE~Gen
IpService                      IKE~Gen
{
  Protocol                     UDP
  SourcePortRange              500
  DestinationPortRange         500
  Direction                    BiDirectional
  Routing                      Local
}

IpService                      All_other_traffic
{
  Protocol                     All
  Direction                    BiDirectional
  Routing                      Local
}

IpDynVpnAction                 IPSec__Silver
{
  Initiation                   Either
  VpnLife                      1440
  Pfs                          None
  IpDataOfferRef               IPSec__Silver~R
  IpDataOfferRef               IPSec__Silver~R~2
}
##
## Connectivity Rule 0 combines the following items:
##   Local data endpoint       0~ADR~1
##   Remote data endpoint      0~ADR~2
##   Topology                  HH
##   Requirement Map           VPNwithWindows
##     All_other_traffic         => IPSec__Silver

IpAddr                         0~ADR~1
```

```
{
  Addr                          9.100.193.234 6
}

IpAddr                          0~ADR~2
{
  Addr                          9.100.199.162 7
}

LocalSecurityEndpoint           0~LSE~4
{
  Identity                      X500dn CN=ITSO,OU=Montpellier,O=IBM,C=France
  LocationRef                   0~ADR~1
}

RemoteSecurityEndpoint          0~RSE~3
{
  Identity                      X500dn CN=ITSO,OU=Montpellier,O=IBM,C=France
  LocationRef                   0~ADR~2
}

KeyExchangeRule                 0~5
{
  LocalSecurityEndpointRef      0~LSE~4
  RemoteSecurityEndpointRef     0~RSE~3
  KeyExchangeActionRef          0
}

KeyExchangeAction               0
{
  HowToInitiate                 Main
  HowToRespond                  Either
  KeyExchangeOfferRef           KEO~1
  AllowNat                      No
}

## NOTE -- Generated IpFilterRule 0~6
IpFilterRule                    0~6
{
  IpSourceAddrRef               0~ADR~1
  IpDestAddrRef                 0~ADR~2
  IpServiceRef                  IKE~Gen
  IpGenericFilterActionRef      Permit~LogYes
}

IpFilterRule                    0~7
{
  IpSourceAddrRef               0~ADR~1
  IpDestAddrRef                 0~ADR~2
  IpServiceRef               All_other_traffic
  IpGenericFilterActionRef   IpSec~LogYes
  IpDynVpnActionRef          IPSec__Silver
}

KeyExchangePolicy
{
  AllowNat                   No
  KeyExchangeRuleRef         0~5
}
```

A few remarks on this policy:

1 and 2 are specified at the beginning of the combined policy to inhibit pre-decapsulation of the IPSec packets and to allow creation of on demand tunnels.

3 The rule for IKE daemon traffic, 0~6, is placed at the top of the filtering rules to ensure that ISAKMP/IKE negotiations could freely proceed.

4 The same goes for the rules specifying which traffic is routed through the tunnel.

5 This is where the VPN policy information begins.

6 This address is the distributed DVIPA.

7 This is the workstation IP address that we had for the test.

Note: The IP addresses in 7 is important. It needs to be narrowly defined so that only the appropriate hosts are permitted to match this rule. This is especially important since these rules appear first in the hierarchy.

Windows XP client changes for the Sysplex Distributor configuration

The workstation's policy file had to be updated to specify the sysplex distributor's address as the target. This was done using MMC and by opening the properties of the policy and editing the filter for the VPN. No other changes were necessary.

Proving Sysplex-Wide Security Association (SWSA) is working

SWSA uses the coupling facility structure EZBDVIPA to make the negotiated security association available to the distributed DVIPA hosts. By thus allowing the tunnel's end to be located at the distributed DVIPA, whatever target member of the sysplex distributor is acting as the DVIPA host. Example G-8 shows the specification of the EZBDVIPA structure in the CFRM couple dataset.

Example: G-8 EZBDVIPA specification

```
STRUCTURE NAME(EZBDVIPA)  (4)
                    SIZE(4096)
                    INITSIZE(1024)
                    REBUILDPERCENT(30)
                    PREFLIST(CF02,CF01)
```

Using the configuration file identified in Example G-7 on page 289, an FTP request issued from the Windows XP workstation and targeting the sysplex distributor address was attempted. The FTP was successful, and the following message was seen in the syslogd log for the distributing host:

```
EZD0818I Tunnel added: 09/07/2005 10:19:36.41 vpnaction= IPSec__Silver
tunnelID= Y10  AHSPI= 0  ESPSPI= 4104364355
```

This message was simultaneously reflected to all target hosts' syslogd output. However, messages related to the use of the tunnel were only reflected on the host that is the actual target for the connection. For example, the first connection established looked like the one shown in Example G-9 when using the command:

```
DISPLAY TCPIP,TCPIP,NETSTAT,VCRT
```

Example: G-9 One distributed VPN connection

```
EZZ2500I NETSTAT CS V1R7 TCPIP 709
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR       DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
9.100.193.234     00021  9.100.199.162    01348  10.1.2.3
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

The destination XCF IP address 10.1.2.3 is host MVSIV. When viewing the syslogd output on that host, traffic in the tunnel produced messages such as below:

```
EZD0814I Packet permitted: 09/07/2005 10:19:59.51 filter rule= 0~7 ext= 2
sipaddr= 9.100.199.162 dipaddr= 9.100.193.234 proto= tcp(6) sport= 1348 dport=
21 -= Interface= 10.1.1.3 (I) secclass= 2 dest= local len= 48 vpnaction=
IPSec__Silver tunnelID= Y10
```

Note that the tunnel ID is Y10. While using this tunnel, no syslogd messages were recorded in the other sysplex distributor target member MVSIV. Establishing a second FTP session from the client workstation resulted in this second connection being distributed to MVSIS. A display at the distributing host now looks like Example G-10.

Example: G-10   Two distributed VPN connections

```
EZZ2500I NETSTAT CS V1R7 TCPIP 834
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST IPADDR      DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
9.100.193.234    00021  9.100.199.162    01349  10.1.2.2
9.100.193.234    00021  9.100.199.162    01348  10.1.2.3
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

When data traffic occurred on this second FTP session, the EZD0814I messages only appeared in the MVSIS syslogd.

There is obviously no need to test the sysplex distributor function, but a shutdown of the distributing host (MVSIT) allowed both FTP sessions to remain active. A restart of MVSIT also was nondisruptive.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 304. Note that some of the documents referenced here may be available in softcopy only.

- *Security Configuration in a TCP/IP Sysplex Environment*, SG24-6527
- *Putting the Latest z/OS Security Features to Work*, SG24-6540
- *z/OS 1.6 Security Services Update*, SG24-6448
- *Implementing VPNs in a z/OS Environment*, SG24-6530

Other publications

These publications are also relevant as further information sources:

- *z/OS ICSF Overview,* SA22-7519
- *z/OS ICSF System Programmer's Guide,* SA22-7520
- *z/OS Cryptographic Services System Secure Sockets Layer Programming*, SC24-5901
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS security Server RACF Command Language Reference,* SA22-7687
- *z/OS Security Server RACF System Programmers Guide,* SA22-7681
- *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 MVS Setting Up a Sysplex,* SA22-7625
- *z/OS UNIX System Services Planning,* GA22-7800
- *z/OS MVS Planning; Workload Management,* SA22-7602

- *System Programmer's Guide to: Workload Manager*, SG24-6472

Online resources

- The RACF-L discussion list at listserv@listserv.uga.edu

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols
$SYSSECA 89
$SYSSECR 89
&SYSCLONE 218
/etc/services 143
_BPXK_SETIBMOPT_TRANSPORT 199
_CEE_ENVFILE 278
_login 67

Numerics
9037-002 118

A
Access control 2
Access Control Lists (ACLs) 85
 inheritance 85
Access Controls 47
Acess rules 52
AF_INET socket 143
AF_UNIX socket 143
aggregate 79
AIM 64
Application Transparent TLS 154
ARM 30
ASSIZE 216
AT-TLS 154
authentication 2
AUTOGID keyword 64
AUTOID keyword 64
AUTOMOVE 97, 99
 EXCLUDE 100, 139
 INCLUDE 100, 139

B
BASIC program security mode 68
BPX.CONSOLE 103
BPX.DAEMON 66, 68
BPX.DAEMON.HFSCTL 103
BPX.DEBUG 71, 74, 103
BPX.DEFAULT.USER FACILITY 60
BPX.DEFAULTUSER 80
BPX.FILEATTR.APF 71, 103
BPX.FILEATTR.PROGCTL 71, 104
BPX.FILEATTR.SHARELIB 71
BPX.MAINCHECK 66, 68–69
BPX.NEXT.USER 64
BPX.SERVER 71, 104
BPX.SMF 105
BPX.STOR.SWAP 105
BPX.SUPERUSER 71
BPX.WLMSERVER 105
BPXMCDS 30
BPXPRMxx 95
BPXTCAFF 200

C
CERTAUTH 279
CFRM 30
change-cipher-spec 267
CHOWN.UNRESTRICTED 72
CINET 199, 217
CKDS 121
 in-storage copy 123, 127
 REFRESH 123
Common INET (C-INET) 199
Communications Server Security Level 3 12
CONSOLxx 40
 LOGON 40
Controlled write-down. 52
coupling facility 25
 shutdown 25
CPUTIME 216
CSFEUTIL 123

D
DAC 47
Data confidentiality 2
Data integrity 2
Data sharing 27
default policy 223
DeMilitarized Zone 147
demilitarized zone (DMZ) 196
Denial of Service 3, 132
 multiple packet DoS 132
 single packet DoS 132

Denial of Service Attack 3
DES 276
DestinationPortRange 228
DIRACC 84
Direction 228
DIRSRCH 84
Discretionary Access Control 47
Discretionary Access Control (DAC) 47
Distributed Dynamic VIPA 109
DMZ 147, 150, 196
Dominance 50
dual-stack 198
dubbing 62
DVIPA 178
dynamic VIPA 178
dynamic XCF 180
DYNAMICXCF 180

E
ECS 21
EIM 10
ELEMBIND=CURJOB 37
e-mail address 277
EMCS 42
ENHANCED program security mode 68
ENHANCED-WARNING program security mode 68
Enterprise Identity Mapping 10
environment 199
environment variable
  _BPXK_SETIBMOPT_TRANSPORT 199
Equivalence 51
ESP 276
ETR 118
extended attributes 71
External load balancer 193
External Time Reference 118
Extranet 150
EZB.IPSECCMD 219
EZB.NETACCESS 135
EZBDVIPA 185
EZBDVIPA structure 300
EZBTCPCS 181

F
file mode bits 82
File Security Packet 81
File Security Packet (FSP) 84
filter policy 223
filter rules 225
firewall 152
Firewall Technologies 8
FIRSTHIT 37
FSOBJ 84
FSP 81
FSSEC class 86
FTP 11
Fully Qualified Domain Name, 277

G
Generic Routing Encapsulation 184
generic server 199
GLOBALCONFIG 146
GRE 184
GRS 22
GSK.SIDCACHE 259
GSK.SIDCACHE. 259
GSK_SYSPLEX_SIDCACHE 193, 259
GSK_V3_SESSION_TIMEOUT 256
GSK_V3_SIDCACHE_SIZE 256
GSKSRVGP 192
GSKSRVR 192, 258–259
GUI 159, 271

H
Hardware Management Console 114
HCR7730 121
HFS compatibility mode 79
HIPAA 47
hipersockets 181
HMC 18
  Domain Security 114
  ESCON Director Control programs 114
  Licensed Internal Code 115
Host-Based Intrusion Detection 169

I
ICHRDSNT 23
ICMP 228
ICSF 5
Identification 2
IDS 159, 170
  Attack 172
  event 173
  IDS probes 169

policy 171
Scan detection 171
Traffic regulation 172
IEFSSNxx 26
IKE 157, 159
IKE daemon 271
IKED 159
IKED_CTRACE_MEMBER 279
IKED_FILE 278
implicit 276
IMWEBSRV 262
inbound direction 151
Internet Key Exchange 157
Intranet 150
intrusion 132
Intrusion detection 168
intrusion detection system (IDS) 169
IOEFSPRM 79
IP datagram 132
IP Filtering 150
IP filtering default policy 222
IP filtering filter policy 222
IPCONFIG 222–223, 271
IPDestAddr 228
IpFilterAction 228
IpFilterGroup 228
IpFilterPolicy 228
IpFilterRule 228
IPSEC 225, 272
IPsec 160
ipsec 137, 219, 223, 232, 284, 287
IPSEC statement 222
IPSec VPN 153
IPSECRULE 225
IPSECURITY 222–223, 271–272
IpService 228
IPSourceAddr 228
IRR.DIGTCERT.LISTRING 279
IRRXCF00 23
ISAKMP 161
ISAKMP/IKE 276–277
ISO 17799 19
ISO Standard 7498-2 2
IUTSAMEH 181
IVTPRM00 146
IXCARM.elemtype.elemname 38
IXCMIAPU 30, 34
IXGLOGR 35
IXLCONN 28–29
IXLFORCE 28
IXLREBLD 28

J
JES2 21

K
Key Part Import 126
Key Record Create 126
Key Record Delete 126
Key Record Write 126
KGUP 123

L
LDAP Directory Server 9
Load Operational Key 126
LOGDISABLE 225
LOGENABLE 225
LOGIMPLICIT 225
LOGON 40
AUTO 41
OPTIONAL 41
REQUIRED 40
LOGOPTIONS (ALWAYS) 88
LOGR 30
LOGSTRM 37

M
MAC 48
MAC Access 51
Read/Write test 52
Read-Only test 51
Write-Only test 51
MAC principle 53
MAIN 69
Mandatory Access Control 48
Mandatory Access Control (MAC) 48
MCS console 40
MD5 276
MEMLIMIT 216
MLACTIVE 92
MLS 46
Access Controls 47
Access rules 52
Controlled write-down 52
Disjoint 50
Dominance 50

Equivalence 50
SECLABELs 54
MMC 280–281
MODIFY 283, 288
MOUNT 78
Multilevel Security 46, 54, 88
SECLABELs 54
MVS 59
MVS/ESA 48
MVSADMIN.XCF. 31

N
NETSTAT 136
Network Address Translation 156, 163
Network Authentication Service 10
Network Configuration Assistant 159, 270
Network Security Configuration Assistant 272
Network-based Intrusion Detection 169
NOAUTO 65
NOAUTOMOVE 99
NODATAGRAMFWD 197
Non-repudiation 2

O
OCEP 6
OCSF 6
OMPROUTE 11
OMVS 60
OMVS couple data set 93
ONLYAT keyword 66
OpenSSH for z/OS 12
OPERCMDS 41
OPERPARM 42
OS/2 115
OSA 182
OSPF 11, 181, 272
OSPF MD5 11
outbound direction 151

P
PADS 69
PAGENT 154, 157, 223, 241, 282
MODIFY 230
policy agent 216
REFRESH 230
pasearch 137, 157, 240, 287
personal certificate 279
PKCS7DER 281
PKDS 122
PKITP 6
Policy Agent 154
port 500 276
pre-defined customer security proposals 276
Pthread_security_np() 67
public key cryptography 187

Q
QoS 158

R
RACF 21, 47–48
BPX.DEBUG 71, 74, 103
BPX.FILEATTR 71
BPX.FILEATTR.APF 71, 103
BPX.FILEATTR.PROGCTL 104
BPX.FILEATTR.SHARELIB 71, 104
BPX.SERVER 71, 104
BPX.SMF 105
BPX.STOR.SWAP 105
BPX.SUPERUSER 71
BPX.WLMSERVER 105
data sharing mode 26
non-data sharing 26
UNIXPRIV class 72
CHOWN.UNRESTRICTED 72
SUPERUSER.FILESYS 72
SUPERUSER.IPC.RMID 73
SUPERUSER.PROCESS 73
SUPERUSER.SETPRIORITY 74
RACF user revocation 204
Redbooks Web site 304
Contact us xiii
RefreshLifetimeAccepted 276
RefreshLifetimeProposed 283
Remote Support Facility 117
Resource sharing 21
RESTRICTED attribute 87
RESTRICTED.FILESYS.ACCESS 87
RESTRICTLOWPORTS 136
RFC 1701 184
RFC 2246 164
RFC 2328 11
RFC 2401 160
RFC 2408 161
RFC 2409 161

root file system 94
ROUTE 44
ROUTE command 18
Routing 229
RSF 117
rshd 11
RVARY 24, 26
  ACTIVE 24
  DATASHARE 24
  INACTIVE 24
  NODATASHARE 24
  SWITCH 24

S
SECLABEL 57
SECLBYSYSTEM 98
SECLEVEL 57
Secure Socket Layer 164
Security 49
security association 161
Security class 184
security class 224
Security Label
  Dominance 50
  Equivalence 51
Security Label diagram 50
Security Labels 49
Security Level 49
security policy database 151
SecurityClass 229
sendmail 11
SERVAUTH 134, 219
  NETACCESS 134
  PORTACCESS 135
session-ID 187
SETOMVS 97
SETR MLS 57
SETROPS
  NORACLIST 24
SETROPTS 24
  GENERIC 24
  GLOBAL 24
  MLFSOBJ 90
  RACLIST 24
  WHEN(PROGRAM) 24
setUID 67
Setuid() 67
SFM 30
SHARED 64
shared HFS 93
SHARED keyword 63
shared secret key cryptography 187
SHARED.IDS 63
SHARED.IDS profile 63
SHAREOPTIONS 122
snap-ins 281
SNMPv3 12
Socks server 156
SourcePortRange 228
Spawn() 67
SPD 151
SSL/TLS 153
stack affinity 199
STDENV DD 278
STP 118
su 67
subplexes 204
SUPERUSER 61
SUPERUSER.FILESYS 72, 87
SUPERUSER.FILESYS.ACLOVERRIDE 87
SUPERUSER.FILESYS.CHANGEPERMS 85
SUPERUSER.FILESYS.MOUNT 72
SUPERUSER.IPC.RMID 73
SUPERUSER.PROCESS 73
SUPERUSER.SETPRIORITY 74
Support Element 114
SWSA 181, 185, 299
symmetric key 276
SYS1.PARMLIB(CONSOLxx) 20
SYSICSF 125
SYSLOGD 142
  facility 143
  isolation 144
  priority code, 143
syslogd 142
  /etc/services 143
  AF_INET socket 143
  AF_UNIX socket 143
  isolation 144
  security 142
SYSM.TCPCFG.POLICY.VPN 273
SYSMULTI 92
SYSNAME 97
Sysplex Distributor 178
Sysplex distributor 179
sysplex distributor 269
Sysplex support 122

Sysplex Wide Security Association 181
SYSPLEXCKDS 125
System Logger 21
System logger 32
System SSL 7, 165–166
SYSZCKDS 126
SYSZCKT 126

T
TCP/IP 146
  CSM 146
  memory 146
  stack hardening 147
TCPIMAGE 217
TCSEC 48
Telnet 11
TLS 187
  abbreviated handshake 190
  handshake 187
  handshake protocol 187
  record protocol 187
  session-ID 187
  session-ID caching 190
  Sysplex session-ID caching 191
TN3270 11
Traffic Regulation Monitoring 159
Triple-DES 276
TRMD 159, 173, 225, 241
TRMDSTAT 172
TRUSTED 37
trusted computing base 48
TSOAUTH 20
TTLSEnabled 255
tunnel ID 300

U
UDP 276
UID 60
UID(0) 61
UNIX 59
UNIX System Services
  extended attributes 71
  superuser 71
  superuser granularity 71
UNIXPRIV 20
UNIXPRIV class 63
UNIXPRIV RACF class 72
  SUPERUSER.IPC.RMID 73
  SUPERUSER.PROCESS 73
  SUPERUSER.SETPRIORITY 74
User Security Packet 81
USP 81

V
VERSION 95
version file system 94
VIPA 178
VIPA devices 225
VIPADEFINE 178
VIPADISTRIBUTE 109
VIPADYNAMIC 178
VIPADYNAMIC BACKUP 178
VIPADYNAMIC VIPAROUTE 180
VIPAROUTE 181, 185
Virtual IP Address 178
Virtual Private Network 154
VPN 154
  Encapsulation mode 163
  symmetric encryption 161

W
WEBSRV 262
Windows XP workstation 280
WLM 30, 107, 113, 121
  Classification rules 111
  policy 110
  Report classes 111
  resource group 140
  Resource Groups 111
  service classes 110
  service definition 110
  service policies 110
  workload 110

X
X.500 distinguished name 277
XCFAS 22–23, 37

Z
z 48
z/OS Cryptographic Services 5
z/OS Firewall Technologies 154
z/OS IDS Policy 171
  Attack detection, reporting, and prevention 172
  Traffic regulation for TCP connections and UDP

receive queues 172
z/OS Integrated Security Services 8
z/OS Intrusion Detection Services 170
z/OS PKI Services 7
z/OS Security Level 3 10
z/OS Security Server 8, 47
zFS 78

IBM

Redbooks

Sysplex eBusiness Security z/OS V1R7 Update

Sysplex eBusiness Security z/OS V1R7 Update

Redbooks

Exploiting the sysplex advantages in an exposed environment

Showing the newest z/OS V1R7 Security features at work

The many things to take into consideration

This IBM Redbook provides an overview of the z/OS Security setups for Parallel Sysplex installations that are considering serving users locally or over non-secure TCP/IP networks. It provides insight into what can be done to minimize the risks in such contexts by addressing the following operating environments:

- Parallel Sysplex (as a stand-alone system) security.
- One member of the Sysplex is exposed to a non-secure network.
- All members of the Sysplex can be reached from the non-secure network.

We use a simple Sysplex configuration running at z/OS 1.7, with the capability of testing workload distribution among the Sysplex members. The basic Security features of z/OS are tested in this environment: SSL/TLS with session ID sharing, Kerberos Key Distribution Center, Communications Server Intrusion Detection Services, and IPSec VPNs with Sysplex Wide Security Association. Other areas of investigation are the potential consequences of resource sharing with members being connected to non-secure networks and what protections are available in terms of z/OS mechanisms and Sysplex configuration best practices.

SG24-7150-00 ISBN 0738494801