IBM

Security Configuration in a TCP/IP Sysplex Environment

- Security perspective on Parallel Sysplex physical configuration
- Topography of an Internet connection and security measures
- Utilization of TCP/IP in a sysplex

Chris Rayns
Angie Boone
Patrick Kappeler
Bruno Lahousse
Alain Roessle

Redbooks

ibm.com/redbooks

International Technical Support Organization

Security Configuration in a TCP/IP Sysplex Environment

May 2003

SG24-6527-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page vii.

First Edition (May 2003)

This edition applies to Security Server for z/OS Version 1, Release 2, for use with the z/OS operating system, Security Server for z/OS Version 1, Release 3, for use with z/OS operating system, and Security Server for z/OS Version 1, Release 4, for use with z/OS operating system.

Contents

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

Trademarks

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

z/OS®
zSeries®
AIX®
CICS®
DB2®
ESCON®
eserver™
FICON™
GDPS®
IBM®
IMS™
Lotus Notes®
Lotus®
MVS™
Notes®
OpenEdition®
OS/390®
Parallel Sysplex®
Processor Resource/Systems Manager™
PR/SM™
Redbooks™
Redbooks(logo) ™
RACF®
RS/6000®
S/390®
SecureWay®
Sysplex Timer®
Tivoli®
VTAM®
WebSphere®

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

Today, many OS/390® installations are considering giving access to the services they are providing via an external network such as the Internet. The large majority of these installations nurtured their processes within the MVS™ and OS/390 “legacy” environment. Therefore, making their applications and data available from an outside TCP/IP network appears as quite a challenge both from a technical and, in some respect, cultural perspective.

Furthermore, installations that capitalized on the OS/390 Parallel Sysplex® technology to achieve stringent availability and scalability objectives via data sharing are now also facing the need to open connections between the outside world and the data and applications managed by their sysplex, without jeopardizing the security of their existing installation.

This IBM Redbook will help those Parallel Sysplex installations that are considering serving users over non-secure TCP/IP networks, such as the Internet, to achieve a broad understanding of the threats to their security. It offers configuration design considerations to keep the related risks at a minimum. The following areas are discussed:

- Utilization of z/OS® UNIX System Services and the relevant security setups
- The Parallel Sysplex physical configurations as scrutinized from the security aspect
- Exploitation and protection of the sysplex coupling mechanisms
- Utilization of TCP/IP in a sysplex and the associated security exposures
- Topography of an Internet connection and security measures

The team that wrote this redbook

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

Chris Rayns is a Security Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM® classes worldwide on all areas of security.

Angie Boone is an Advisory IT Networking Specialist in the United States. She has more than ten years of experience in the installation, implementation, and support of OS/390. She currently works on the TCP/IP customer support team in Raleigh.

Patrick Kappeler is a former computer specialist in the French Air Force and joined IBM in 1970 as a diagnostic programs designer. He has held several specialist and management positions as well as international assignments, all dealing with S/390® technical support. He joined the EMEA S/390 New Technology Center in Montpellier in 1996, where he now provides consulting and presale technical support in the area of e-business security.

Bruno Lahousse is an I/T Specialist in IBM EMEA Advanced Technical Support. He has 15 years of experience in IBM, working with S/390-based customers and systems. His areas of expertise include Parallel Sysplex, Remote Copy Services, and GDPS®. Bruno works on the GDPS Solution Team, where he provides consulting services on disaster recovery and continuous availability solutions for @server zSeries® customers.

Alain Roessle is an architect for the “IBM Design Center for e-business on demand,” in Montpellier, France. Before joining IBM in 2000, he worked for a large distribution company for 20 years. His areas of expertise

include CICS®, DB2®, and Parallel Sysplex. For the last two years he has worked on WebSphere® security issues.

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

George Baker, Worldwide Host Intergration Technical Sales, Raleigh

Travis Finucane, Software Engineer, ACE Team

Casey Cooley, Customer Support Specialist, Raleigh

Isabelle Grimalt, System Support (France), Montpellier

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 and/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 IBM 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 Internet note to:

  redbook@us.ibm.com

- Mail your comments to the address on page ii.

1

Review of z/OS operating system security

This chapter discusses issues related to the security of z/OS UNIX System Services. It provides an overview of RACF® and of the differences between UNIX System Services security and traditional UNIX security. There is also a discussion of how UNIX System Services can be configured to provide the level of platform security required by your installation.

1.1 The threats

We assume that the reader is already acquainted with the platform security implemented, for decades now, 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 “legacy” 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 to steal first the user identification and authentication data, presumably while they are 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 to get any services from the installation. A denial of Service attack can be performed by:
  - overloading the network and the system by 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 that 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 What is 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 are the objectives 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 connecting a Parallel Sysplex 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 controls functions will make 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 in focus this ISO 7498-2 definition of Security all along this book.

1.1.2 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 network(s) an installation is connected to, they also address transmission data confidentiality and integrity.

  Note: These mechanisms initially designed to just 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 Firewall Technologies and the Intrusion Detection Service.

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

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

1.2 Implementing security at the platform level

The intent of this chapter is by no means to go through all the details of the security designed and implemented in MVS at the platform (operating system) level; it is assumed that the reader is already familiar with this topic. Instead, we review the major design points in the approach, see how they have evolved further with OS/390 and z/OS, and explain how they fit in the context of two major technological breakthroughs: the Parallel Sysplex technology and the UNIX branding of OS/390.

1.2.1 The MVS security approach

One of the most important design points for MVS security has always been protecting the operating system from programs, and protecting programs from other programs. MVS was designed to provide multiprocessing and multiprogramming across multiple users, with the capability of processing large amounts of data. To do this, address spaces are created by giving an exclusive collection of virtual storage addresses to each user, thus achieving users isolation. Resources are accessed by programs running in these address spaces, and every task (a unit of work that is performed through the execution of a program) within an address space runs under a security environment based on an authenticated user ID. MVS has been designed to offload security management from the applications standpoint to an external security manager, accessible via the SAF interface. The users' identification and authentication data are managed by this external security manager. RACF (Resource Access Control Facility) is the IBM external security manager for MVS, OS/390, and z/OS.

RACF operates at the software level on the same premise as MVS system integrity which, itself, relies heavily on hardware protection mechanisms, such as supervisor and problem states, storage protection keys, and so forth. These hardware protection mechanisms are also complemented with MVS internal software security features, such as authorized programs and I/O channel program translation.

An understanding of the following terminology is important for our discussion:

- **Private address space**: Virtual storage consisting of a system area, a common area, and a private area. This virtual storage address is converted to a real storage address by means of dynamic address translation (DAT). Each user can address only one private area. OS/390 isolates each user from every other user in a private address space, thereby preventing one user from violating another user's address space. Before MVS allocates the real storage backing this private address space to another user, it clears any residual data from it, thus preventing access to its previous contents.
- **Multiple storage protection keys**: MVS protects information in real storage from unauthorized use by means of multiple storage protection keys. The storage key contains the protection key of the information owner and a fetch protection bit. The protection key protects the block of storage from unauthorized modification. The fetch protection bit protects the block of storage from an unauthorized attempt to read or fetch its contents. Unauthorized attempts to read or modify the contents of real storage are rejected, and a program exception interrupt is issued when such an attempt is detected. The protection key of '0' is only used for *Supervisor* state privileged control instructions and the I/O instructions. All other program instructions are executed in the *Problem* state using other keys.
- **Channel program translation**: OS/390 translates all user-written channel programs; that is, it replaces the channel program's virtual storage addresses with real addresses and builds a new, protected copy of the channel program. This ensures that these programs are unable to read or write information from or into the system's or other user's main storage areas. OS/390 also prefixes user-written channel programs with a `Set File Mask` command which prevents the channel program from reading or writing from or into another user's data set extents.

As mentioned previously, RACF is a software security mechanism for users having access to programs. Program control is a RACF function that allows it to consider load modules (a program that is called within another program) as protected resources. RACF references a table with PROGRAM class (controlled programs) entries. The table entries describe the programs and who can access them. Program libraries can be for public use or for limited private use. If the user is not authorized to execute the program, the system issues abend 306 or 806, ends the request, and records it in an audit log.

A program can also use a function called APF (Authorized Program Facility) to restrict system or user program access to sensitve functions in the system. It allows the system to fetch modules in authorized libraries to prevent programs from counterfeiting a module. Only modules marked authorized and that reside in an authorized library, and the modules that those modules invoke from authorized libraries, can run as authorized programs. An APF list identifies authorized libraries to the system. An APF list/table identifies authorized libraries to the system, and you protect these libraries in the APF list with RACF to reduce the possibility of an integrity exposure.

1.3 z/OS Security Server (RACF)

The z/OS Security Server is a package of security-related products for z/OS, some of them licensed. As of the writing of this book the z/OS Security Server contains:

- As licensed products: RACF, the DCE Security Server, the FTP proxy and SOCKS server of the OS/390 Firewall Technologies, and the OCEP (Open Cryptographic Enhanced Plugin) facility
- No longer licensed, starting with OS/390 2.10 and APAR OW47982:
  - Firewall Technologies Configuration GUI Client
  - IKE daemon (or ISAKMPD, or “key server”) used in the OS/390 VPN Firewall Technology to automatically install shared secret keys between the two ends of a VPN
- No longer licensed, starting with OS/390 2.8: the LDAP server
- Unlicensed product: Network Authentication Service (Kerberos)

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

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

1.3.1 Identification and authentication

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

- If the user is defined to RACF
- If the user has supplied a valid password, pass ticket, and a valid group name
- If the user's ID (UID) and group ID (GID) are valid on z/OS UNIX System Services (UID and GID are explained in 1.4.1, “Traditional UNIX security mechanisms” on page 8)
- If the user ID is in REVOKE status, which prevents a RACF-defined user from entering the system at all or entering the system with certain groups (if the user’s group connection is revoked)
- If the user can use the system on this day of the week and at this time of day (an installation can impose restrictions)
- If the user is authorized to access the terminal (which can also include day and time restrictions for accessing that terminal)
- If the user is authorized to access the application

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

1.3.2 Alternatives to passwords

In a z/OS environment, RACF allows alternatives to passwords, such as a pass ticket, to be used for authenticating users. In a z/OS UNIX System Services environment where users are also identified with numeric user identifiers (UIDs) and group identifiers (GIDs), these too may be used by RACF to control user access to system resources. Unlike user names or group names, these numeric IDs can be shared by more than one user or group, although sharing is not recommended.

In a client/server environment, RACF has the ability to map a client’s digital certificate to a RACF user ID, the digital certificate being stored in the RACF database, or mapped by a

certificate name filter rule. A digital certificate or digital ID, issued by a Certificate Authority, contains information that uniquely identifies the client.

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

1.3.3 Checking authorization

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

Figure 1-1 on page 6 shows how RACF uses the RACROUTE REQUEST=AUTH call to check authorization. The resource managers issue the other RACF calls in a similar manner.

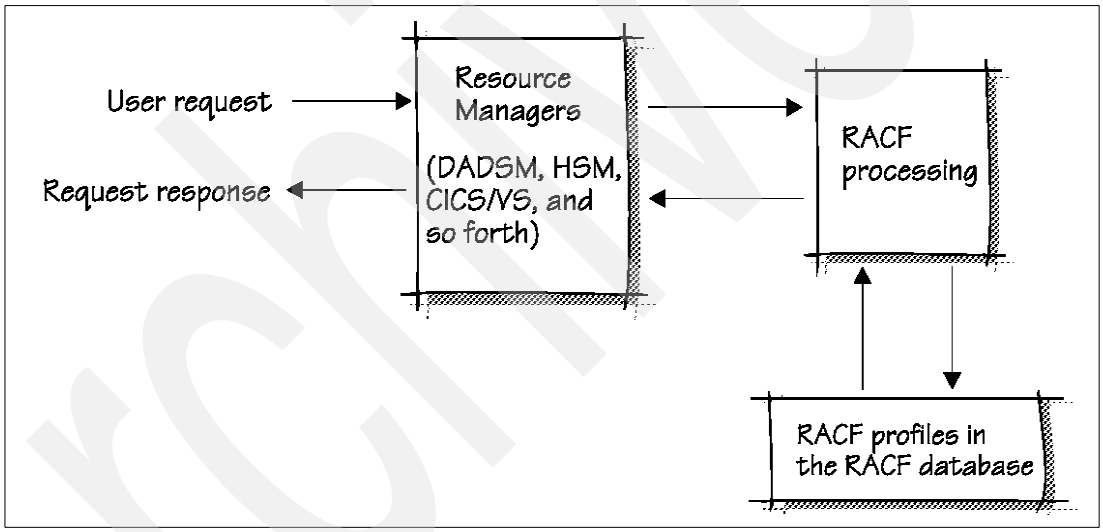


Figure 1-1 Example of RACROUTE REQUEST=AUTH processing

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

1. Checks the profiles to determine whether the user is authorized to access the resource. z/OS data sets are protected by profiles in the RACF class DATASET, but HFS files are protected differently. In the z/OS environment, an HFS itself is a z/OS data set protected by a DATASET profile that controls access to the data set. However, individual files and directories within the HFS are protected by means of permission bits, as described in 1.5.5, "Access permission to HFS files and directories" on page 16.
2. Checks the security classification of the user and data.
3. Gives the user access to the resource if any of the following conditions is satisfied:
   - The resource is a data set and the high-level qualifier is the user's user ID.
   - The user ID is in the access list with sufficient authority.
   - Any of the groups the user is connected to is in the access list with sufficient authority.
   - The universal access authority (UACC) or the access granted to ID(*) is sufficiently high.

1.3.4 RACF logging and reporting

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

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

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

1.3.5 RACF and z/OS UNIX System Services

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

UNIX System Services users are defined with RACF commands. When a job starts or a user logs on, the user ID and password are verified by existing z/OS and RACF functions. When an 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

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

1.4 Security in UNIX systems

UNIX and z/OS UNIX System Services Security manage user identities differently. Table 1-1 contrasts some aspects of how user names and identities are handled by UNIX systems and z/OS UNIX.

Table 1-1 UNIX security comparison

Category	UNIX	z/OS MVS	z/OS UNIX
User identity	Users are assigned a unique UID: 4-byte integer and user name	Users are assigned a unique user ID of 1 to 8 characters	Users are assigned a unique user ID with an associated UID
Security identity	UID	user ID	UID for accessing traditional UNIX resources and the user ID for accessing traditional z/OS resources
Login ID	Name used to locate a UID	Same as the user ID	Same as the user ID

Category	UNIX	z/OS MVS	z/OS UNIX
Special user	Multiple user IDs can be assigned a UID of 0	RACF administrator assigns necessary authority to users	Multiple user IDs can be assigned a UID of 0 or users can be permitted to BPX.SUPERUSER
Data set access	Superuser can access all files	All data sets controlled by RACF profiles	Superuser can access all HFS files; data sets controlled by RACF profiles
Identity change from superuser to regular user	Superuser can use the `su` command to switch into any UID	APF-authorized program can invoke SAF service to change identity	• If FACILITY class profile BPX.DAEMON is not defined, the superuser can `su` into any other UID • If BPX.DAEMON is defined, the superuser must know the password of the other user ID or have access to SURROGAT class profile BPX.SRV.userid
Identity change from regular user to superuser	The `su` shell command allows change if user provides root's password	No provision for unauthorized user to change identity	The `su` shell command allows change if the user is permitted to the BPX.SUPERUSER FACILITY class profile or if the user provides the password of a user with a UID(0)
Identity change from regular user to another regular user	The `su` shell command allows change if user provides password	No provision for unauthorized user to change identity	The `su` shell command allows change if user provides password
Terminate user processes	Superuser can kill any process	MVS operator can cancel any address space	Superuser can kill any process, UNIXPRIV class profile can authorize non-UID(0) users
Multiple logins	Users can log on to a single user ID multiple times	Users can only log on to TSO/E once per user ID	Users can `rlogin` or `Telnet` multiple times to a single user ID in the UNIX shell, and log on once to TSO/E at the same time
Login daemons	`inetd`, `rlogind`, `lm`, and `telnetd` process user requests for login; a process is created with the user identity (UID)	TCAS and VTAM® process user requests for logon; a TSO/E address space (process) is created with the user identity (user ID)	Users can log on to TSO/E or log in using one of the login daemons; in all cases, an address space is created with both an MVS identity (user ID) and a UID

1.4.1 Traditional UNIX security mechanisms

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

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

User ID and group ID

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

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

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

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

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

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

Access control lists (ACLs)

ACLs are a facility offered by some versions of UNIX, including AIX. The need for the functionality they provide is described in this section.

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

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

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

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

Auditing

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

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

1.5 OS/390 and z/OS UNIX System Services security

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

> **Note:** Other security products are available, but they are not covered here. This redbook exclusively refers to RACF, and statements made about OS/390 and z/OS UNIX System Services security take only RACF into account.

In OS/390, each user who may access the system has an identification called a user ID. For each user ID, a user profile is defined in class USER in the RACF database; the user profile contains all information about this user. The data elements for a user that are needed in the OS/390 UNIX environment, such as UID or home directory, are defined in the OMVS segment of the user profile, so the identity of UNIX users in z/OS is actually made of an MVS identity and a UNIIX identity. Each one of these identities is used for access control decisions based on the nature of the resource being accessed, as shown in Figure 1-2.

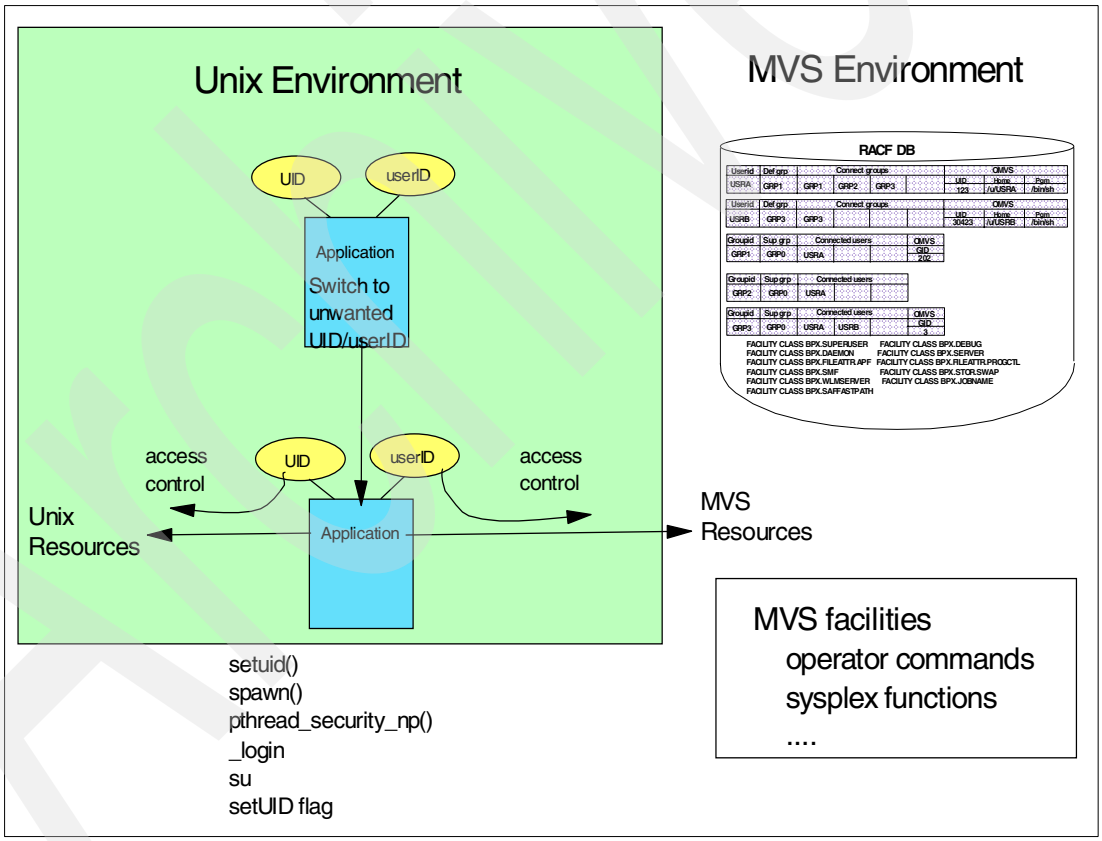


Figure 1-2 Shows a UNIX user in z/OS and how it is made up

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

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

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

In 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 system programmers. In this type of system, allowing a superuser to switch into the identity of any z/OS UNIX user could be a security risk.

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

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

1.5.1 UNIX-level security

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

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

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

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

> **Note:** We do not regard UNIX-level security to be adequate for a z/OS UNIX System Services environment and do not recommend running any system at this level of security. Consequently, the rest of this chapter assumes a system running with z/OS UNIX System Services security.

1.5.2 z/OS UNIX System Services-level security

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

This level of security is for customers with stricter security requirements and who need to have some superusers maintaining the file system, but who 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 large range of privileges the superuser is granted.

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

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

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

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

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

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

The controlled program environment

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

In a controlled program environment (also called a *program-controlled address space*), all programs that are loaded into the address space must meet one of the following conditions:

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

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

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

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

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

And an example of defining a module in the HFS as program-controlled is as follows:

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

Daemon authority

As mentioned previously, switching to another UID and user ID without first authenticating the user requires the daemon authority. Why is this so potentially dangerous that a special authority from RACF is required? A program that can switch into the identity of any UID has almost the same authority as a superuser in traditional UNIX. This could make a z/OS system lose accountability, at least as far as user IDs that have a UID defined are concerned. When a process with daemon authority switches into another UID without authenticating the user first, RACF will search for a user ID that this UID belongs to in the same way it resolves UIDs to user IDs in an `ls -l` display. The process will then run with the UID and the user ID determined by RACF. As a consequence, a process with daemon authority can switch into any RACF user ID that has a UID defined.

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

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

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

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

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

BPX.SERVER authority

As mentioned previously, when a server program issues the service `pthread_security_np`() to create a thread running under this user ID and UID, its user ID needs access to profile BPX.SERVER in class FACILITY.

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

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

- Before issuing `pthread_security_np`(), the server program authenticates the user ID and specifies the correct password.
  - The server's user ID needs READ access to BPX.SERVER; all authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread.
- The user ID of the server program has READ access to profile BPX.SRV.userid in class SURROGAT
  - The server's user ID needs READ access to BPX.SERVER. All authorization requests for resources accessed by the thread are checked against the user ID or UID of the thread.

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

1.5.3 Brief review of the z/OS UNIX user's dual identity

As discussed previously, changing the UNIX effective UID in a z/OS UNIX thread or process, also implies most of the time to change the MVS userID in order to keep the correspondence between the two IDs the same as described in the RACF data bas. Table 1-2 provides a synthesis of the UNIX functions which eventually change the MVS userID.

Table 1-2 UNIX UID and MNVS userID changes

USS function	Change Effective UID	Change MVS UserID	Additional Controls (RACF)[a]
setuid flag	Y	N	
su in shell	Y=0	N	userID access to BPX.SUPERUSER
su with userid and password	y	y	
su with userid without password	Y	Y	userID surrogate to target userID
setuid() or _login with authentication of target ID	Y	Y	clean address space
setuid() or _login without authentication of target ID	Y	Y	userID surrogate of target or (userID has UID(0) +access to BPX.DAEMON) + clean address space[b]
pthread_security_np with authentication of target ID	Y	Y	userID permitted to BPX.SERVER + clean address space
pthread_security_np without authentication of target ID	Y	Y	userID 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	userID surrogate of target or (userID has UID(0) + access to BPX.DAEMON) + clean address space

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

1.5.4 Why z/OS UNIX System Services is a more secure UNIX

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

- No /etc/passwd file.

  UNIX System Services relies on RACF for user authentication. This means that user information is not kept in /etc/passwd and /etc/security/passwd, respectively, and that all user administration is performed outside the z/OS UNIX System Services environment. It is easy to configure z/OS UNIX in a way that no access to the RACF database is possible from a UNIX process. This prevents brute-force password attacks against UNIX passwords and stops intruders from altering user information from within z/OS UNIX.

- Granularity of auditing and reporting.

  UNIX System Services and RACF provide comprehensive auditing, allowing numerous events to be audited. The auditing information is written to SMF data sets, which can be made inaccessible even to superusers. Reporting is based on an open architecture that allows the use of practically any reporting package. This provides better detection of suspicious events.

- Protection of daemon programs from modification and misuse.

  Programs that perform security-critical functions must run in a controlled program environment. A modification to a module in such an address space or an attempt to load a module from another, uncontrolled library will cause the program to fail.

- Superuser granularity control, a function that allows non-superusers authorized by RACF to use services that would otherwise require the user to be a superuser.

  This can reduce the number of superusers required in a system.

- The storage keys implemented in the S/390 and z900 hardware, together with the concept of address in z/OS, isolates applications from each other and from the operating system. A program that exceeds its allocated storage fails with a storage exception rather than overwriting storage areas of the operating system or other applications. This greatly reduces the possible damage that could be done by buffer overflows and similar problems.

- TCP/IP stacks, ports, and network addresses can be protected with RACF to prevent unauthorized users and programs from using them.

- The ability to assign different user identities (user IDs and UIDs) not only to processes but also to threads within a process allows access control in z/OS UNIX to be performed on a more granular basis than is possible in traditional UNIX environments. Specifically, programs such as the IBM HTTP Server for z/OS can run each thread under the client's identity. This allows the access authority of the client to be checked for all requests.

1.5.5 Access permission to HFS files and directories

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

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

The basic access algorithm can be deduced by referring to Figure 1-3.

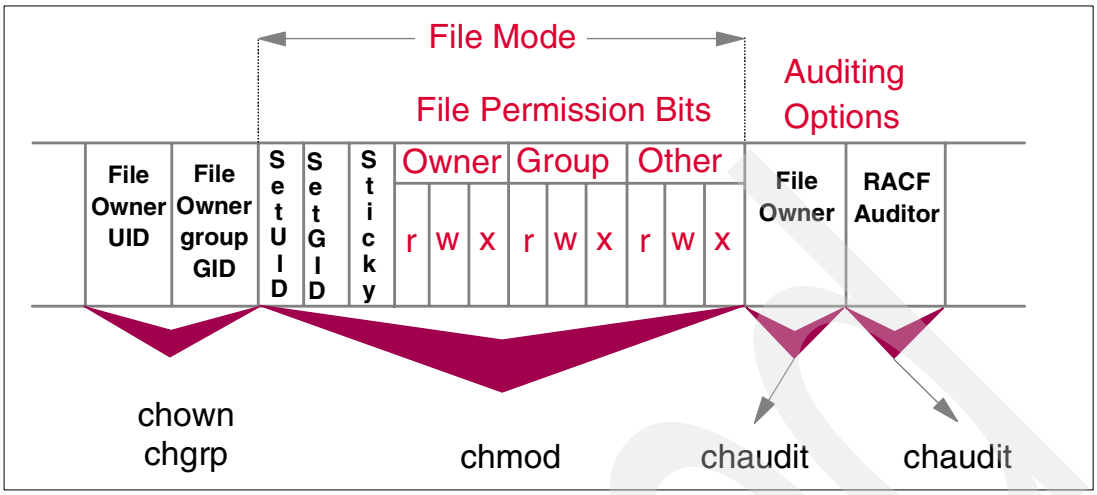


Figure 1-3 Hierarchical File System file security packet

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 permissions

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

- Owning group permissions

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

- Other permissions

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

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

Table 1-3 is an overview of types of access and the permissions granted by the accesses.

Table 1-3 File access types and permission bits

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

Notes®:

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

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

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

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

If RACF denies access, an error message will be logged to SYSLOG as shown in Example 1-1.

Example 1-1 Resource access authorization error

```
ICH408I USER(RAYNS ) GROUP(SYS1    ) NAME(CHRIS RAYNS IBM US)
  /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 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 are only used to hold auditing options, so don't look for any profiles in this class. FSOBJ means *file system objects*, in other words, files. DIRACC would mean read/write access to a directory while DIRSRCH would appear if a directory search failed. For a description of all classes used, see Appendix A in *z/OS V1R3.0 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 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.

1.5.6 Displaying files and directories

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

Example 1-2 Output of `ls -l` *command*

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

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

Class UNIXMAP and VLF classes IRRUMAP and IRRGMAP

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

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

Application Identity Mapping (AIM)

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

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

1.5.7 UID/GID assignment to a process

As mentioned earlier, the first use of a z/OS UNIX service causes the z/OS address space to be *dubbed* into a z/OS UNIX process. The User Security Packet (USP) is an important control

block created by dubbing. Among the information it holds are the UIDs relevant to the process:

- **Real UID:** At process creation, the real UID identifies the user who has created the process.
- **Effective UID:** Each process also has an effective UID. The effective UID is used to determine the owner access privileges of a process.
- Normally this value is the same as the real UID. It is possible, however, for a program that resides in the Hierarchical File System to have a special flag set that, when this program is executed, changes the effective UID of the process to the UID of the owner of the program. A program with this special flag set is said to be a set-user-ID program. This feature provides additional permissions to users while the set-user-ID program is being executed.
- **Saved UID:** Used to save the effective UID of a process. When a z/OS UNIX user tries to change the UID, the saved_UID field is checked to see if the UID is the original one.
- **Real GID:** At process creation, the real GID identifies the group of the user for which the process was created.
- **Effective GID:** Each process also has an effective group. The effective GID is used to determine the group access privileges of a process. Normally this value is the same as the real GID. Some programs, however, have a special flag set that, when the program is executed, changes the effective GID of the process to the GID of the owner of this program. A program with this special flag set is said to be a set-group-ID program. Like the set-user-ID feature, this provides additional permission to users while the set-group-ID program is being executed.
- **Saved GID:** Used to save the effective GID of a process. When a z/OS UNIX user tries to change the GID, the saved GID is checked to see if the GID is the original one.

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

1.5.8 Defining UNIX System Services users

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

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

Then define the new user:

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

Note: For the UID, any value up to 2,147,483,647 can be used. However, UID(0) should only be assigned to superuser IDs. To avoid the task of manually keeping track of all the UIDs that are in use, we recommend using standard procedures to define users without OMVS segments, and using the TSO ISHELL to add the OMVS segment to the user ID. The TSO ISHELL will keep track of the UIDs it has issued and will assign the next available UID to a new user. Figure 1-4 shows how to invoke the appropriate ISHELL function.

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
-----------------------------------------
                              OpenMVS ISPF S

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on

Return to this panel to work with a differ

   /u/boche
   ____________________________________
   ____________________________________
   ____________________________________
```

```
1 _1. *User...
   2. *User list...
   3. *All users...
   4. *All groups...
   5. *Permit field access...
   6. *Character Special...
   7. Enable superuser mode(SU)
```

```
Some choices (*) require
superuser or the "special"
attribute for full function, or
both
```

Figure 1-4 The setup drop-down menu in IDPF ISHELL

Select **Setup** -> **1. *User...** to bring up the window shown in Figure 1-5; this assigns the next free UID to an existing user and also allows you to set the other OMVS segment parameters.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help

                    Set Up a Current TSO/E User

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

 User ID . . . . . . . . . . newuser
 Home directory  . . . . . /u/newuser
 Initial program . . . . . /bin/sh

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

Figure 1-5 ISHELL window to assign a UID and other OMVS segment parameters to a user

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

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

- Execute permission to the “/” and “/u” directories
- Read, write, and execute permission to the /u/user01 directory

If you do not give user01 sufficient permission, any attempt to log in to the z/OS UNIX shell will fail.

Note: Defining /u/userid as the home directory for a user is a frequently used convention, but any valid directory name can be defined as a user's home directory.

1.5.9 Default user

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

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

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

- UID

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

- HOME

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

  – Define a home directory where the default user does not have write permission, such as the root (/) directory.

  – Define the HOME directory as the /tmp directory, which is designed for temporary files created by different users.

- PROGRAM

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

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

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

The following is an example of the setup required for a default user and group:

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

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

2. Define a default GROUP and USER:

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

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

3. Refresh the FACILITY class profile:

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

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

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

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

Note: Because many z/OS UNIX processes now run with the default user's UID, the number of allowed processes per UID as defined in BPXPRMxx with the MAXPROCUSER statement may be too low.

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

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

Note: If you allow users to access UNIX System Services resources using the default UID and GID, be aware that these users can access all "world-readable" HFS files and directories. In an HFS environment, it is often not feasible to avoid world-readable files because of the limited possibilities offered by the POSIX permission bits. Therefore, the use of the default UID and GID requires careful planning and risk assessment.

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

1.5.10 Superuser

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

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

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

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

Then define the superuser:

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

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

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

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

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

Table 1-4 shows the resource names that are used in class UNIXPRIV and lists the privileges associated with each resource.

Table 1-4 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

Resource name	z/OS UNIX privilege	Access required
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.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
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 on 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 class UNIXPRIV to make the profiles resident in storage. Therefore, any addition or modification to profiles in this class requires the profiles to be refreshed using the command:

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

1.5.11 Started task user IDs

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

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

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

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

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

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

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

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

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

Figure 1-6 is a sample of the output produced by the RLIST command.

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

Figure 1-6 RLIST output for STARTED class profile

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

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

1.5.12 FACILITY class profile BPX.SUPERUSER

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

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

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

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

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

Issue the following RACF commands:

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

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

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

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

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

A daemon program that wants to make use of the authority to BPX.SUPERUSER to gain superuser status needs to invoke the `seteuid`() service. Therefore, using a non-UID(0) user ID with access to BPX.SUPERUSER instead of a superuser user ID for a daemon or server program is a solution only if the program does, indeed, support this function.

1.5.13 FACILITY class profile BPX.DAEMON

The profile BPX.DAEMON is very important to the concept of z/OS UNIX-level security and the overall system security in z/OS, as we discussed previously. If you are not familiar with

identity switching, daemon authority, and the controlled program environment, review 1.5.2, “z/OS UNIX System Services-level security” on page 11.

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

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

Note: Activating z/OS UNIX-level security in a system that did not use this level of security before could cause some daemon programs to fail. Before defining this profile or the FACILITY class profile BPX.SERVER, you should make sure that all daemons that use security-critical services are loaded from a controlled program environment and that appropriate authority to SURROGAT class profiles has been established. For details see “The controlled program environment” on page 13.

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

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

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

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

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

User ID BPXROOT

In “Daemon authority” on page 14, we explain the use of user ID BPXROOT. This user ID should not be used to log on to any interactive services, so it is appropriate to define it as a protected user ID (using the NOPASSWORD parameter). To define the BPXROOT user ID, enter:

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

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

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

1.5.14 Additional BPX.* FACILITY class profiles

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

- 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 can execute in program-controlled address spaces.

- BPX.FILEATTR.SHARELIB

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

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

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

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

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

- BPX.SERVER

  This FACILITY class profile is explained in detail in "BPX.SERVER authority" on page 14.

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

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

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

- 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.

1.5.15 Programs in the Hierarchical File System

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

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

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

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

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

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

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

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

Processes that use security-critical functions need to run in a controlled program environment (for details see “The controlled program environment” on page 13) to make sure that all programs in the address space are loaded from a trusted source.

The sticky bit

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

UNIX System Services uses the sticky bit to bypass loading of an executable from the Hierarchical File System. If an executable file has the sticky bit turned on, as shown in Example 1-3, the executable file in the HFS will not be used, but z/OS will instead turn to the standard MVS search order to look for a copy of the executable file in an MVS load library.

Example 1-3 Sticky bit

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

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

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

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

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

Extended attributes

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

- Mark an executable file as program-controlled

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

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

- **extattr +p** marks an executable file as program-controlled. See "BPX.FILEATTR.PROGCTL" on page 29 for the authority needed to set this attribute.
- **extattr +a** marks an executable file as APF authorized. Note that the program needs to be linked with AC=1 if it is to be invoked as a job step program. See "BPX.FILEATTR.APF" on page 29 for the authority needed to set this attribute.
- **extattr +l** marks an executable file as a system-shared library object. The program will be loaded into the kernel-shared library region data space. See "BPX.FILEATTR.SHARELIB" on page 29 for the authority needed to set this attribute.
- **extattr +s** marks an executable file as a program that may be loaded into a shared address space. Whether the program will actually be loaded into a shared address space depends on the setting of the environment variable _BPX_SHAREAS.

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

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

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

1.5.16 z/OS UNIX kernel address space

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

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

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

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

1.5.17 z/OS UNIX security considerations for TCP/IP

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

TCP/IP address spaces

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

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

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

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

VARY TCP/IP command

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

If the installation has activated class OPERCMDS in RACF, profiles should be defined to control the use of the VARY TCPIP command. The resource names shown in Table 1-5 are in use.

Table 1-5 RACF OPERCMDS resources for the VARY.TCPIP command

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

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

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

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

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

1.5.18 IBM-supplied daemons

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

z/OS UNIX supplies these daemons:

- InetD, the Internet daemon
- rlogind, the remote login daemon

- cron, the batch scheduler
- lm, the Communications Server login monitor
- uucpd, the UUCP daemon

The InetD, rlogind, cron, lm, and uucpd programs reside in the HFS directory /usr/sbin. They have the sticky bit turned on.

InetD

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

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

If InetD is started as a started procedure, a profile in class STARTED is needed to assign a user ID to InetD.

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

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

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

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

telnetd, RSHD, REXECD, and rlogind

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

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

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

Both the telnetd process and the shell process execute under the end user ID, and not the original daemon user ID. The shell process has the telnetd process as parent, the telnetd

process has the InetD process as parent, and InetD in turn has the highest level process of all, the BPXOINIT process, as parent.

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

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

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

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

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

Routing daemons

On z/OS IP there are two routing daemons:

- ORouteD

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

- OpenEdition® Multiprotocol Route Application (OMPROUTE)

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

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

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

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

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

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

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

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

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

1.5.19 MVS sockets server applications

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

1.5.20 Summary

Table 1-6 gives a summary of all the definitions you have to create to run UNIX System Services servers with MVS-like security.

Table 1-6 Overview of processes, UIDs and RACF FACILITY class profiles

Process/Job	UID	BPX.SERVER access	BPX.DAEMON access
OMVS	0	NONE	READ
TCP/IP system address space	0	NONE	NONE
InetD	0	NONE	READ
syslogd	0	NONE	NONE
FTPD	0	NONE	NONE
Client User	Not 0	NONE	NONE
Web Server	0	READ	NONE
Web User	Web User not 0, not UID of administrator, not in same group	NONE	NONE

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

You may also wish to investigate the sample job that will assist you in performing some of these tasks. The job can be found in member EZARACF, shipped in hlq.SEZAINST, which contains sample RACF commands for most TCP/IP-related security functions.

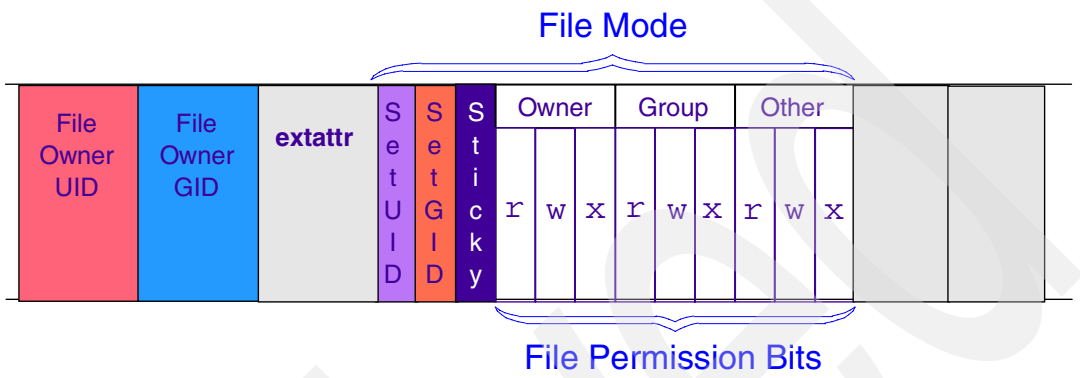
1.6 Access control list (ACL) support for z/OS 1.3

Traditionally, the authorization checking for access to z/OS UNIX files and directories in a file system has been done using the file security packet (FSP), as shown in Figure 1-7 on page 37.

The FSP is stored in the file system as part of the attributes of a file or directory and is created when the file directory is created. If a security authorization is needed for a file or

directory, the security packet is passed to the security product for authorization checking. The level of authorization for the file or directory through the FSP allows specification of file permission bits for file owner (user), group owner, and others, but cannot permit or restrict the access to other specific users and groups.

HFS and zFS files are currently protected with POSIX permission bits that are contained within the FSP in the file system (not in RACF).



Access	Permission for File	Permission for Directory
Read (r)	Permission to read or print the contents.	Permission to read, but not search, the contents.
Write (w)	Permission to change, add to, or delete from the contents.	Permission to change, add, or delete directory entries.
Execute (x)	Permission to run the file. This permission is used for executable files.	Permission to search the directory.

Figure 1-7 File security packet (FSP)

1.6.1 File access authorization checking

Both the ACL and FSP are maintained by the Physical File System (PFS). When a security decision in needed, the file system calls the security product, supplying the ACL, if present, and the FSP. If the security product supports ACLs, it applies its own rules to the file access request.

RACF uses the permission bits, the ACL, and the following UNIXPRIV class profiles to determine whether the user is authorized to access the file with the requested access level:

- SUPERUSER.FILESYS
- RESTRICTED.FILESYS.ACCESS
- SUPERUSER.FILESYS.ACLOVERRIDE

Once RACF has checked the authorization, RACF returns control to the file system.

SUPERUSER.FILESYS profile

This profile in the UNIXPRIV class was introduced in OS/390 V1R8. The UNIXPRIV class provided the capability to assign specific superuser functions to a user or group when you give a user or group either a:

- UID of 0

- BPX.SUPERUSER profile

Either of these gives a user or group access to all UNIX functions and resources. A BPX.SUPERUSER profile allows you to request that you be given such access, but you do not have the access unless you make the request. So, instead of giving a user or group access to all functions a superuser has, the UNIXPRIV class provides profiles that allows access to a specific superuser functions.

The SUPERUSER.FILESYS profile in the UNIXPRIV class has three access levels that allow access to z/OS UNIX files as follows:

READ Allows a user to read any local file, and to read or search any local directory.

UPDATE Allows a user to write to any local file, and includes privileges of READ access.

CONTROL/ALTER Allows a user to write to any local directory, and includes privileges of UPDATE access.

RESTRICTED attribute

You can define a restricted user ID by assigning the RESTRICTED attribute through the ADDUSER or ALTUSER command, as follows:

```
ALTUSER RSTDUSER RESTRICTED
```

User IDs with the RESTRICTED attribute cannot access protected resources they are not specifically authorized to access. Access authorization for restricted user IDs bypasses global access checking. In addition, the UACC of a resource and an ID(*) entry on the access list are not used to enable a restricted user ID to gain access.

Note: The RESTRICTED attribute does not prevent users from gaining access to z/OS UNIX file system resources unless you take certain steps, as shown in the next section.

However, the RESTRICTED attribute has no effect when a user accesses a z/OS UNIX file system resource; the file's “other” permission bits can allow access to users who are not explicitly authorized. To ensure that restricted users do not gain access to z/OS UNIX file system resources through “other” bits, you must use the new UNIXPRIV profile, RESTRICTED FILESYS.ACCESS.

1.6.2 New UNIXPRIV profiles with z/OS V1R3

The three new UNIXPRIV profiles are:

- **RESTRICTED.FILESYS.ACCESS**

  This profile in the UNIXPRIV class controls the access to file system resources for restricted users based on the “other” permission bits.

- **SUPERUSER.FILESYS.ACLOVERRIDE**

  This profile allows RACF to force the use of ACL authorizations to override a user’s SUPERUSER.FILESYS profile authority.

- **SUPERUSER.FILESYS.CHANGEPERMS**

  This profile 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.

RESTRICTED.FILESYS.ACCESS profile

This profile specifies that RESTRICTED users cannot gain file access by virtue of the “other” permission bits. Checking for this new profile, RESTRICTED.FILESYS.ACCESS, is done for RESTRICTED users regardless of whether an ACL exists, so this function can be exploited regardless of whether you plan to use ACLs or not. You can define the profile as follows:

```
RDEFINE UNIXPRIV RESTRICTED.FILESYS.ACCESS UACC(NONE)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

Note: Using UACC(READ) on the RESTRICTED.FILESYS.ACCESS profile does not work since, by definition, a RESTRICTED user cannot be granted access via a UACC. This would be a meaningless thing to do given that you simply would not define RESTRICTED.FILESYS.ACCESS if you want “other” bits to be checked for RESTRICTED users.

For exception cases, permit the RESTRICTED user (or one of its groups) to the RESTRICTED.FILESYS.ACCESS profile as follows:

```
PERMIT RESTRICTED.FILESYS.ACCESS CLASS(UNIXPRIV) ID(RSTDUSER) ACCESS(READ)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

This does not grant the user access to any files. It just allows the “other” bits to be used in access decisions for this user.

Note: SUPERUSER.FILESYS still applies to RESTRICTED users regardless of the existence of the RESTRICTED.FILESYS.ACCESS profile.

SUPERUSER.FILESYS.ACLOVERRIDE profile

Any user who is not a superuser with UID(0) or the file owner, and is denied access through the ACL, can still access a file system resource if the user has sufficient authority to the SUPERUSER.FILESYS resource in the UNIXPRIV class. Therefore, to prevent this, you can force RACF to use your ACL authorizations to override a user's SUPERUSER.FILESYS authority by defining the following profiles:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.ACLOVERRIDE UACC(NONE)
SETROPTS RACLIST(UNIXPRIV) RACLIST
```

Note: This describes the relationship between the existing SUPERUSER.FILESYS profile and the new SUPERUSER.FILESYS.ACLOVERRIDE profile. Either profile could get checked for a file; it depends upon the presence of an ACL for the file, and the contents of the ACL for granting access.

For exception cases, permit the user or group to SUPERUSER.FILESYS.ACLOVERRIDE with whatever access level would have been required for SUPERUSER.FILESYS as follows:

```
PERMIT SUPERUSER.FILESYS.ACLOVERRIDE CLASS(UNIXPRIV) ID(ADMIN) ACCESS(READ)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

SUPERUSER.FILESYS authority is still checked when an ACL does not exist for the file. This should be done for administrators for whom you want total file access authority. That is, you do not want anyone to deny them access to a given file or directory by defining an ACL entry for them with limited or no permission bit access.

Important: The intent of these new profiles is to allow ACLs to behave as much as possible like RACF profile access lists. The new profiles are provided to avoid changing the default behavior, as that could introduce compatibility issues with previous releases.

For more information about the new RESTRICTED.FILESYS.ACCESS, SUPERUSER.FILESYS.ACLOVERRIDE, and SUPERUSER.FILESYS.CHANGEPERMS UNIXPRIV profiles, refer to *z/OS V1R3 Security Server RACF Security Administrator's Guide*, SA22-7683.

SUPERUSER.FILESYS.CHANGEPERMS profile

As an enhancement to superuser granularity, when using the `chmod` command, a RACF service (IRRSCF00) has been updated to check the caller's authorization to the resource SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class if the caller's user ID is not either:

- UID(0)
- The owner of the file
- BPX.SUPERUSER

If the user executing the `chmod` command has at least READ authority to the resource, the user is authorized to change the file mode in the same manner as a user having UID(0).

This profile 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.

1.6.3 ACL overview

ACLs have existed on various UNIX platforms for many years, but with variations in the interfaces. ACL support in z/OSV1R3 is based on a POSIX standard that was never approved, and other UNIX implementations.

In the POSIX standard, two different ACLs are referenced as follows:

- Base ACL entries are permission bits (owner or group) that refer to the FSP.
- Extended ACL entries are ACL entries for individual users or groups, such as the permission bits that are stored with the file, not in RACF profiles.

See "ACL entries" on page 41 for a description of these entries.

Access control lists (ACLs) are introduced in z/OS V1R3 as a way to provide a greater granularity for access to z/OS UNIX files and directories. z/OS V1R3 provides support for access control lists (ACLs) to control access to files and directories by individual users (UIDs) and groups (GIDs). ACLs are now created and checked by RACF. ACLs are created, modified, and deleted by using either of the following:

- **`setfacl`** shell command
- ISHELL interface

To display ACLs, use the `getfacl` shell command. The HFS and zFS file systems support ACLs.

> **Attention:** Before you can begin using ACLs, your security product must support ACLs.

Migration considerations

For migration, you need to upgrade any nodes that share HFS or zFS to z/OS V1R3 or, at a minimum, apply the compatibility APAR OW49334 to the down-level systems.

1.6.4 Security product and ACLs

ACLs are used together with the permission bits in the FSP to control the access to z/OS UNIX files and directories by individual users and groups. An ACL is mapped by the SAF IRRPFACL macro, as shown in Figure 1-8 on page 41, where the set of user entries is followed by the set of group entries. The entries are sorted in ascending order by UID and GID in order to help optimize the access checking algorithm. It consists of a list of entries (with a maximum of 1024) where every entry has information about the type (user or group), identifier (UID or GID), and permissions (read, write, and execute) to apply to a file or directory.

ACL entries

There are two types of ACL entries as follows:

- **Base ACL entries**

  These entries are the same as permission bits (owner, group, other) that have always existed with z/OS UNIX files and directories. You can change the permissions using the `chmod` or the new `setfacl` commands. They are not physically part of the ACL, although you can use the `setfacl` command to change them and the `getfacl` command to display them.

- **Extended ACL entries**

  These entries are for individual users or groups and, like the permission bits, they are stored with the file, not in RACF profiles. Each ACL type (access, file default, directory default) can contain up to 1024 extended ACL entries. Each extended ACL entry specifies a qualifier to indicate whether the entry pertains to a user or a group, the actual UID or GID itself, and the permissions being granted or denied by this entry. The allowable permissions are read, write, and execute. As with other UNIX commands, the `setfacl` command allows the use of either names or numbers when referring to users and groups.

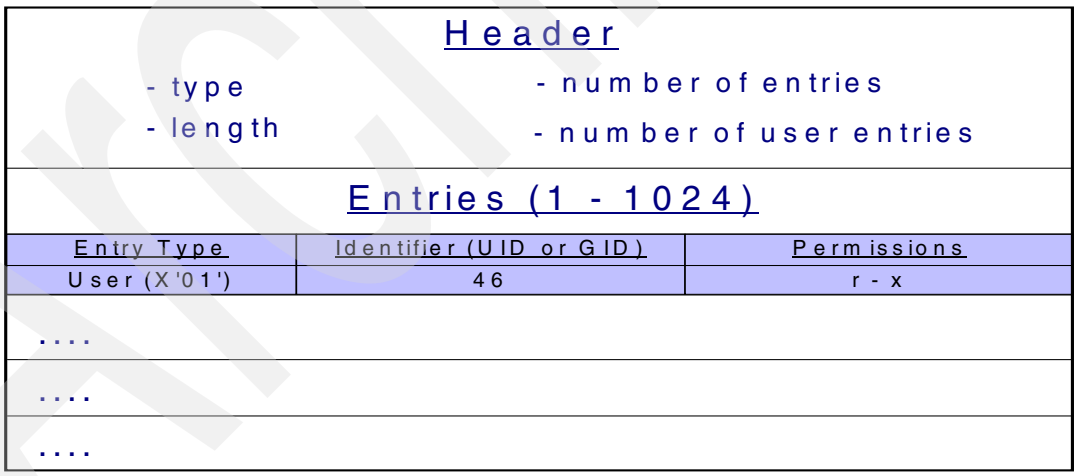


Figure 1-8 Access control list table

There is no such thing as an empty ACL. If there is only one entry and it is deleted, the ACL table is automatically deleted.

For more information on UNIX System Services enhancements in z/OS 1.3, see the appropriate product documentation.

1.7 Enhancements for UID/GID support in z/OS 1.4

Enhancements in z/OS V1R4 have been made in the way that UIDs and GIDs can be assigned by RACF. They can be automatically assigned to new users and prevented from being shared or allowing to be shared.

New keywords have been added to the SEARCH command in order to determine the set of users assigned to a given UID or the groups assigned to a given GID.

The RACF UNIXMAP class is currently used to allow the system to quickly look up a user ID from a UID, or a group name from a GID. An enhanced RACF database organization can now perform this conversion quickly without requiring the mapping profiles in the UNIXMAP class. If desired, you should use the IRRIRA00 utility to convert the RACF database to the new organization.

1.7.1 RACF database and AIM

To use a new RACF profile, SHARED.IDS, and the new SEARCH keywords, the RACF database must have application identity mapping (AIM) stage 2 or 3 implemented. You can convert your RACF database to stage 3 of application identity mapping using the IRRIRA00 conversion utility. See the *z/OS Security Server RACF System Programmer's Guide*, SA22-7681 for information about running the IRRIRA00 conversion utility.

If your installation is new to RACF and you are not running any releases prior to OS/390 Version 2 Release 10, you will automatically take advantage of application identity mapping at the stage 3 level without running the IRRIRA00 conversion utility, and you will not need to use VLF and UNIXMAP to achieve improved performance.

IRRIRA00 utility

This utility was new in OS/390 V2R10. It converts an existing RACF database to application identity mapping functionality using a four-stage approach.

With a database created at OS/390 Release 10 or higher, RACF's application identity mapping uses an alias index to associate user and group names with:

- Lotus® Notes® for z/OS
- Novell Directory Services for OS/390
- z/OS UNIX identities

RACF typically uses mapping profiles for this purpose with databases created before Release 10. However, you can use the IRRIRA00 utility to convert the database to work with an alias index instead. The conversion consists of a series of stage transitions from zero to three. RACF uses the database mapping information differently for each stage.

> **Note:** Systems with an older release are not aware of an alias index or the application identity mapping conversion stages, so you should use care when sharing a database between older systems and a system at Release 10 or higher.

AIM stage 3

In stage 3, RACF locates application identities (such as UIDs and GIDs), for users and groups by using an alias index that is automatically maintained by RACF. This allows RACF to more efficiently handle authentication and authorization requests from applications such as z/OS UNIX than was possible using other methods, such as the UNIXMAP class and VLF. Once your installation reaches stage 3 of application identity mapping, you will no longer have

UNIXMAP class profiles on your system, and you can deactivate the UNIXMAP class and remove VLF classes IRRUMAP and IRRGMAP.

> **Important:** Associating RACF user IDs and groups to UIDs and GIDs has important performance considerations. If your installation shares the RACF database with systems running releases prior to OS/390 Version 2 Release 10, it is important to use the VLF classes IRRUMAP and IRRGMAP and the UNIXMAP class to improve performance by avoiding sequential searches of the RACF database for UID and GID associations.
>
> If your installation shares the RACF database with only systems running z/OS, or OS/390 Version 2 Release 10 or above, you may be able to achieve improved performance without using UNIXMAP and VLF. However, before you can avoid using UNIXMAP and VLF, you need stage 3 of application identity mapping by running the IRRIRA00 conversion utility.

1.7.2 Search enhancements to map UIDs and GIDs

The use of the new SEARCH keywords, USER and GROUP, requires at least AIM stage 2 installed, and it does not require any particular authority. When a UID or GID is specified, all other keywords, except CLASS, are ignored. The RACF search command is as follows:

```
SEARCH CLASS(USER) UID(0)
```

Using the UID keyword, as shown in Example 1-4, you will obtain a list of all the users assigned to the UID specified.

Example 1-4 Display all users with a UID=1

```
SEARCH CLASS(USER) UID(1)
LDAPSRV
MSYSLDAP
```

Using the GID keyword, as shown in Example 1-5, you will obtain a list of all the groups assigned to the GID specified.

Example 1-5 Display all groups with a GID=1

```
SEARCH CLASS(GROUP) GID(1)
OMVSGRP
```

RACF SEARCH panel

If you prefer to use RACF panels, the SEARCH panel for both user and group profiles has been modified to support this new enhancement on the SEARCH command, as shown in Figure 1-9 on page 44, with the UID selection criteria.

```
                         RACF - SEARCH FOR USER PROFILES
COMMAND ===>

ENTER OPTIONAL SELECTION CRITERIA:

   MASK1    ________     Selects profiles with names that begin with the
                         specified character string.

   MASK2    ________     Selects profiles with names that contain the
                         specified string somewhere after MASK1.

   FILTER   ________     Selects profiles with names that match the
                         specified string.

   AGE      ____         Selects users that have not accessed the system
                         in the number of days specified.

   USERID   ________     Selects the profiles this user is authorized to see
                         (administrators only).

   UID      1_________   Selects profiles which have this UID defined in
                         the OMVS segment (other options will be ignored).
                         Valid values are 0 - 2147483647.

   ___ Enter YES to generate a TSO clist
       (Command Direction is inactivated for SEARCH with clist)

   ___ Enter YES to specify additional SEARCH criteria
```

Figure 1-9 RACF search for user profiles panel

To search for UID(1) users, enter a 1 for the UID, as shown in Figure 1-9; the response is shown in Figure 1-10.

```
 BROWSE - RACF COMMAND OUTPUT----------------------- LINE 00000000 COL 001 080
 COMMAND ===>                                            SCROLL ===> HALF
******************************** Top of Data *********************************
LDAPSRV
MSYSLDAP
******************************* Bottom of Data ********************************
```

Figure 1-10 RACF search response for UID(1)

1.7.3 Shared UID prevention

To prevent several users from having the same UID number, a new RACF SHARED.IDS profile has been introduced in the UNIXPRIV class. This new profile acts as a system-wide switch to prevent assignment of a UID which is already in use.

The use of the SHARED.IDS profile requires AIM stage 2 or 3 and is uniqueness guaranteed within the scope of GRSplex.

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
```

Once the SHARED.IDS profile has been defined and the UNIXPRIV class refreshed, it will be not allow a UID to be assigned if the UID is already in use. The same is true for GIDs: 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 new functionality does not affect pre-existing shared UIDs. They remain as shared once you install the new support. If you want to eliminate sharing of the same UID, you must clean them up separately. The release provides a new IRRICE report to find 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 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)
```

Note: To specify the SHARED operand, you must have the SPECIAL attribute or at least READ authority to the SHARED.IDS profile in the UNIXPRIV class.

1.7.4 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. Previously, this was a manual process and guaranteed the uniqueness of the UID and GID for every user.

Using a new AUTOUID keyword with the ADDUSER and ATLUSER commands, an unused UID will be 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:

- 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 2 qualifiers separated by a forward slash (/). The qualifier on the left of the slash character specifies 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 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 example

In the following example, we have defined the APPLDATA for a range of values from 5 to 70000 for UIDs and from 3 to 30000 for GIDs. USERA and USERB are added using the automatic assignment of UID. The range of automatic UID assignment starts with 5, so USERA is assigned to UID(5), which was free. UID(6) and UID(7) were already assigned before we started our example, so the first following UID free is 8. USERB is assigned to UID(8).

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('5-70000/3-30000')

ADDUSER USERA OMVS(AUTOUID)
IRR52177I User USERA was assigned an OMVS UID value of 5.

ADDUSER USERB OMVS(AUTOUID)
IRR52177I User USERB was assigned an OMVS UID value of 8.
```

RACF extracts the APPLDATA from the BPX.NEXT.USER and parses out the starting value. It checks if it is already in use and if so, the value is incremented and checked again until an unused value is found. Once a free value is found, it assigns the value to the user or group and replaces the APPLDATA with the new starting value, which is the next potential value or the end of the range.

In our example, that means that if UID(6) becomes free after UID(7) is assigned to USERB, RACF will start checking from UID(8) in the next assignment so, it will not assign UID(6). But

you can change the APPLDATA and modify the starting value. The APPLDATA can be changed using the following command:

```
RALTER FACILITY BPX.NEXT.USER APPLDATA('2000/500')
```

Note: Automatic assignment of UIDs and GIDs fails if you specify a list of users to be defined with the same name or if you specify the SHARED keyword. Also, AUTOUID or AUTOGID is ignored if UID or GID is also specified.

APPLDATA examples

Here are some examples of correct and incorrect APPLDATA specifications:

Good data 1/0
1-50000/1-20000
NOAUTO/100000
/100000
10000-20000/NOAUTO
10000-20000/

Bad data 123B
/
2147483648 /* higher than max UID value */
555/1000-900

If you have an incorrect specification and attempt to use AUTOUID on an ADDUSER command, the following message is issued:

```
IRR52187I Incorrect APPLDATA syntax for the BPX.NEXT.USER profile.
```

Automatic assignment with RACF panel

You may use the RACF panels to define the OMVS segment. Figure 1-11 on page 47 indicates how to use the automatic assignment by using the AUTOUID field.

```
                         RACF - CHANGE USER JANE
                         OMVS PARAMETERS
COMMAND ===>

 Delete ALL OMVS information      (NOOMVS)  ___    Enter YES to DELETE

    -- OR --

 Choose to CHANGE or DELETE, then press ENTER.
                                                              More:     +
 Specify new User Identifier         (UID)  __________  0 - 2147483647
 Allow shared use of this UID     (SHARED)  _           Enter any character
    -- or --
 Assign a unique UID             (AUTOUID)  _           Enter any character
    -- or --
 Delete User Identifier            (NOUID)  _           Enter any character

 Change Initial Path Name           (HOME)  _           Enter any character
 Delete Initial Path Name         (NOHOME)  _           Enter any character

 Change Program Path Name        (PROGRAM)  _           Enter any character
 Delete Program Path Name      (NOPROGRAM)  _           Enter any character

 Specify CPU Time             (CPUTIMEMAX)  __________  7 - 2147483647
 Delete  CPU Time           (NOCPUTIMEMAX)  _           Enter any character

 Specify Address Space Size    (ASSIZEMAX)  __________  10485760 -
                                                           2147483647
 Delete  Address Space Size  (NOASSIZEMAX)  _           Enter any character
```

Figure 1-11 RACF panel to set OMVS parameters for automatic assignment

Automatic assignment in an RRSF configuration

In an RRSF configuration, shown in Figure 1-12, in order to avoid UID and GID duplications, use non-overlapping APPLDATA ranges.

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 V1R4.0 Security Server RACF Security Administrator's Guide*, SA22-7683.

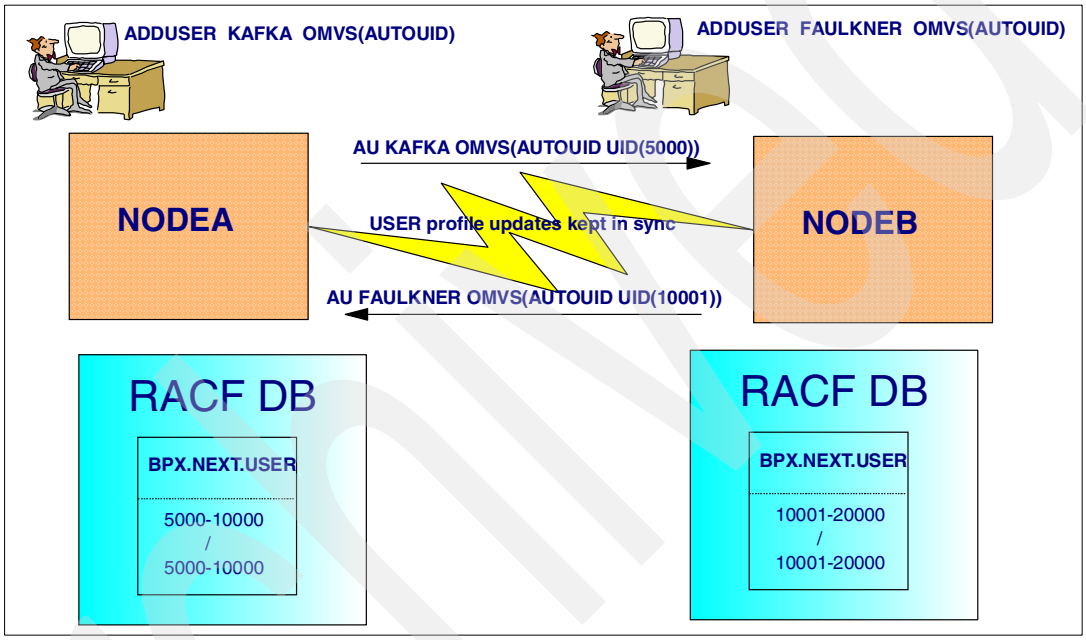


Figure 1-12 Automatic assignment in an RRSF configuration

1.7.5 Group ownership option

The behavior for assigning file group ownership without using the new UNIXPRIV profile available with zOSV1R4, shown in Figure 1-13, is as follows:

- The owning UID is taken from the process' effective UID.
- The owning GID is taken from the parent directory.

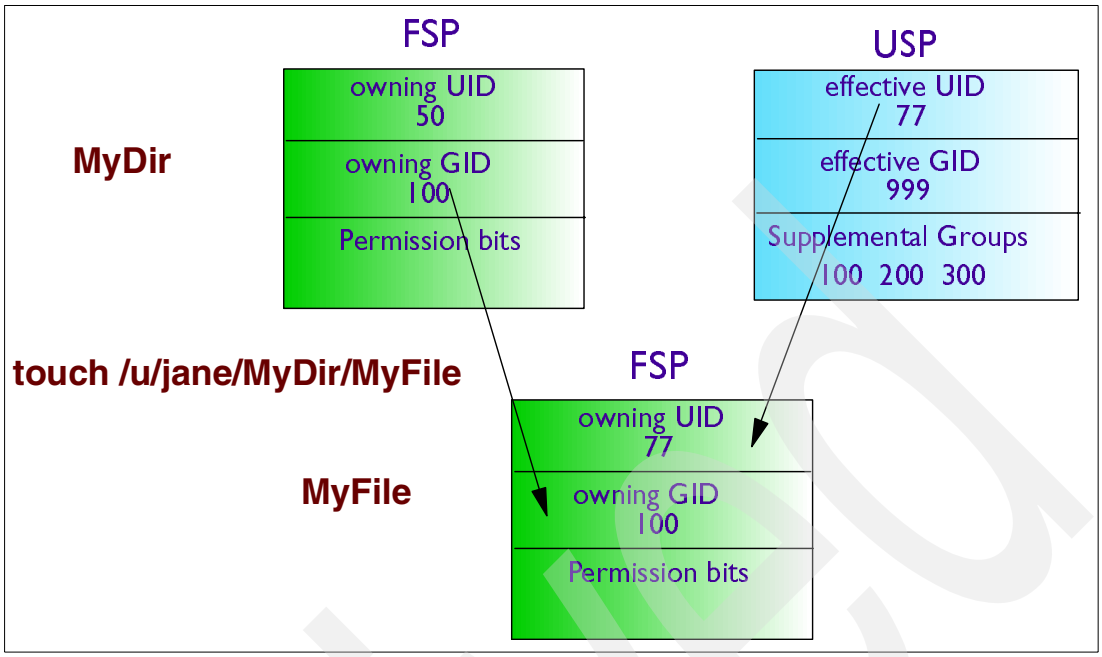


Figure 1-13 Group ownership example without UNIXPRIV profile

New group ownership option

The new group ownership option introduced in z/OS V1R4 allows you to specify that the group owner of a new file is to come from the effective GID of the creating process.

To enable this new option, the following steps must be done:

1. Define A FILE.GROUPOWNER.SETGID profile in the UNIXPRIV class and do a refresh of the UNIXPRIV class:

```
RDEFINE UNIXPRIV FILE.GROUPOWNER.SETGID
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

2. Turn off the set-gid bit for the directory (it is the default) if is not already off:

```
chmod g+s /u/jane/MyDir
```

Once the new group ownership option is enabled, the new behavior is as shown in Figure 1-14.

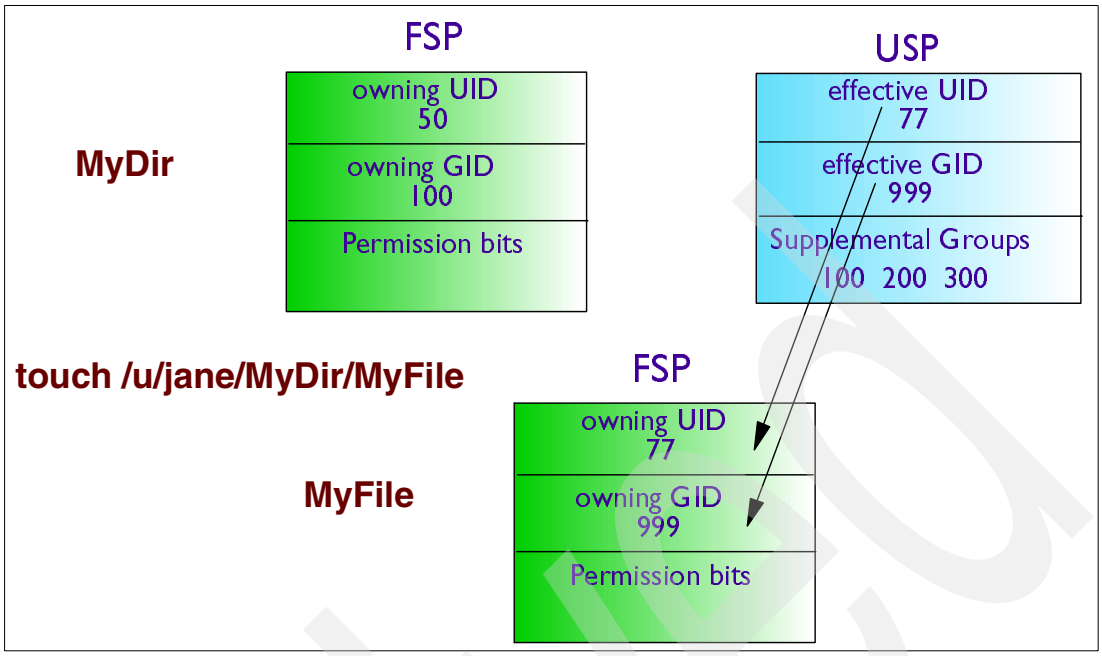


Figure 1-14 New group ownership option with UNIXPRIV profile

Note: When a new file system is mounted, you must turn on the set-gid bit of its root directory if you want new objects within the file system to have their group owner set to that of the parent directory.

2

Overview of Parallel Sysplex technologies

This chapter provide an overview of hardware and software technologies that are used in a Parallel Sysplex, concentrating on sysplex environment descriptions and definitions.

To get a sense of the evolution—from a single system uniprocessor, through tightly coupled multiprocessors and loosely coupled configuration, to the sysplex—refer to Chapter 1, “Introduction to a Sysplex,”, in *z/OS Parallel Sysplex Overview: Introducing Data Sharing and Parallelism in a Sysplex,* SA22-7661.

MVS provides several system commands to manage and operate a sysplex, of which we use some in this chapter. *z/OS MVS System Commands*, SA22-7627 contains a complete description of all system commands and their parameters.

2.1 Parallel Sysplex definition

A *sysplex* is a collection of MVS systems that cooperate, using certain hardware and software products, to process work. All systems in a sysplex share the same sysplex name and a sysplex couple data set, which holds control information and records status information for the sysplex.

If the systems in a sysplex are all using CTC connections for communication, and no coupling facility, the sysplex is called a *base sysplex*.

In contrast, a *Parallel Sysplex* is a collection of MVS systems, all of which have access to the same one or more coupling facilities. A coupling facility enables parallel processing and improved data sharing for authorized programs running in the sysplex.

If you want to enhance the capability of your enterprise to recover from disasters and other failures, you also can use a *Geographically Dispersed Parallel Sysplex* (GDPS). A GDPS is a multi-site application-availability solution that provides the capability to manage remote copy configuration and storage subsystems, automates Parallel Sysplex operational tasks, and performs failure recovery from a single point of control, thereby improving application availability.

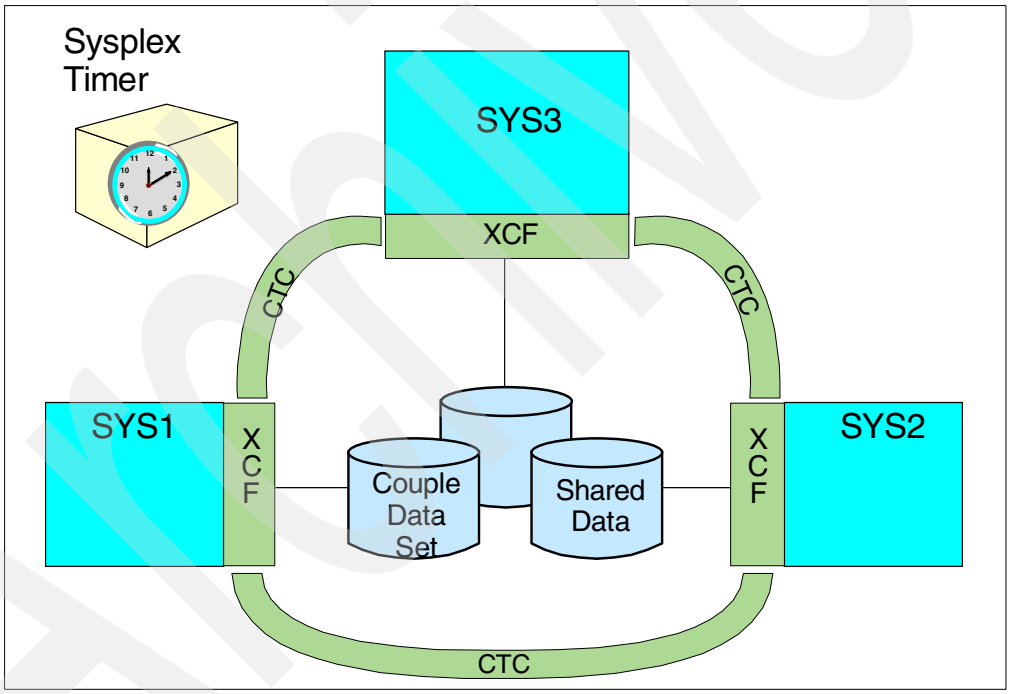


Figure 2-1 Sysplex structure overview

2.1.1 Hardware

Certain hardware is necessary to run a Parallel Sysplex. In this section we introduce some of the hardware used in a Parallel Sysplex.

Coupling facility

To run a Parallel Sysplex, you need a *coupling facility*. The coupling facility provides high-speed caching, list processing, and a locking function, and enables high-performance multisystem data sharing.

A coupling facility is defined as a special logical partition on either a 9674 Coupling Facility, a 9672 Parallel Sysplex Enterprise Server, a zSeries 900 server, or a zSeries 800 server.

The level of CFCC code that is loaded into the coupling facility determines what functionality is available. A complete and up-to-date list of coupling facility details and enhancements introduced with each CFLEVEL can be found at the following Web site:

http://www.ibm.com/servers/eserver/zseries/pso/cftable.html

Note: Higher CFLEVELs might require higher versions of your operating system. For example, CFLEVEL 12 requires z/OS V1.4 to take advantage of all enhancements. Also note that when migrating to higher CFLEVELs, list, lock, and cache structure sizes might need to be increased to support new function.

To display all the coupling facilities in a sysplex, z/OS provides the D CF command. The same command used with parameter CFNAME= allows you to list only specific coupling facilities.

Figure 2-2 shows the output of the command D CF,CFNAME=CF05. The output contains the coupling facility name, the CFLEVEL and CFCC release, and the path and device information. The space utilization shows allocated storage for structures and dump space.

```
D CF,CFNAME=CF05
IXL150I  17.14.35  DISPLAY CF 583
COUPLING FACILITY 009672.IBM.02.000000050822
                  PARTITION: F  CPCID: 00
                  CONTROL UNIT ID: FFFE
NAMED CF05
COUPLING FACILITY SPACE UTILIZATION
 ALLOCATED SPACE                  DUMP SPACE UTILIZATION
  STRUCTURES:     339968 K         STRUCTURE DUMP TABLES:        0 K
  DUMP SPACE:       2048 K                   TABLE COUNT:        0
 FREE SPACE:      691456 K        FREE DUMP SPACE:            2048 K
TOTAL SPACE:     1033472 K       TOTAL DUMP SPACE:            2048 K
                                MAX REQUESTED DUMP SPACE:        0 K
   VOLATILE:         YES          STORAGE INCREMENT SIZE:      256 K
    CFLEVEL:           9
    CFCC RELEASE 09.00, SERVICE LEVEL 01.14
    BUILT ON 05/10/2002 AT 15:32:00

COUPLING FACILITY SPACE CONFIGURATION
                             IN USE          FREE         TOTAL
  CONTROL SPACE:           342016 K      691456 K     1033472 K
  NON-CONTROL SPACE:             0 K           0 K           0 K

  SENDER PATH       PHYSICAL            LOGICAL       CHANNEL TYPE
          00         ONLINE              ONLINE        CBS
          02         ONLINE              ONLINE        CBS

  COUPLING FACILITY DEVICE     SUBCHANNEL     STATUS
                     FFFC         229D        OPERATIONAL/IN USE
                     FFFD         229E        OPERATIONAL/IN USE
                     FFFE         229F        OPERATIONAL/IN USE
                     FFFF         22A0        OPERATIONAL/IN USE
```

Figure 2-2 Sample output of command D CF,CFNAME=CF05

Coupling facility channels

There must be at least two operational signaling paths (one inbound and one outbound path) between each of the systems in the sysplex. These connection links can be:

- ESCON® or FICON™ channels between the CPCs

  or

- Coupling facility channels, which provide high-speed connectivity between the coupling facility and the CPCs that use the coupling facility

Processors

Selected models of S/390, z900, or z800 processors can take advantage of a sysplex. A single system is referred to as a Central Electronic Complex (CEC) or a Central Processor Complex (CPC).

Sysplex timer

If systems in a sysplex run on different processors, a *sysplex timer* is required to synchronize the TOD clocks.

Additional hardware

ESCON or FICON channels enhance data access and communication in the sysplex. Directors add dynamic switching capability for both ESCON and FICON channels. ESCON- or FICON-attached control units and I/O devices in a sysplex provide the increased connectivity necessary among a greater number of systems.

Figure 2-3 shows a hardware configuration of a Parallel Sysplex using an external coupling facility as well as an LPAR in one of the CPCs acting as a coupling facility.

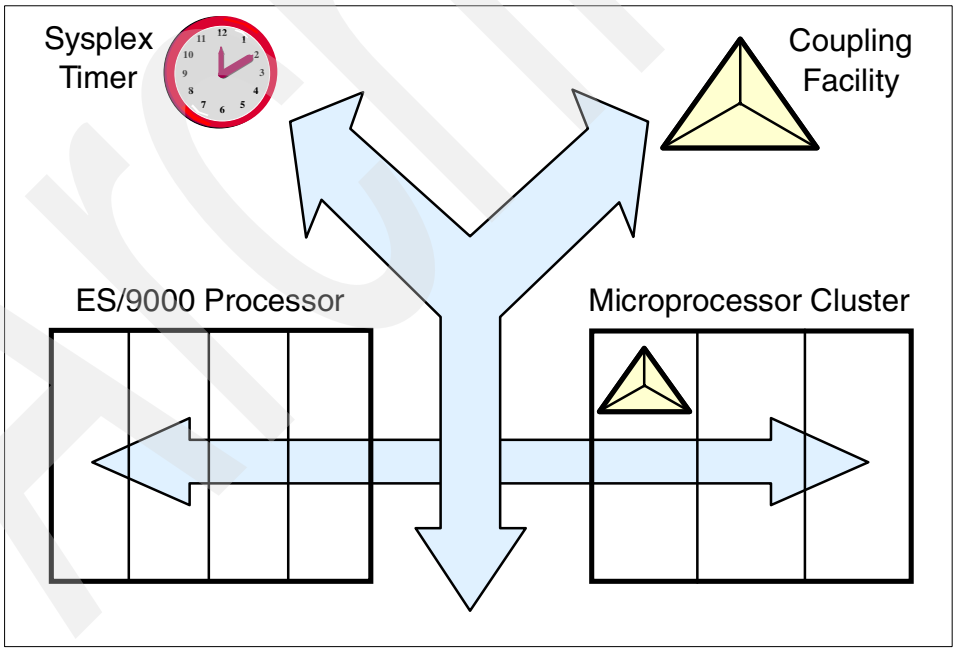


Figure 2-3 Sample configuration of hardware components used by a Parallel Sysplex

2.1.2 Software

At the base operating system level, OS/390 or z/OS operating systems provide the resource management of the coupling facility. MVS is the platform for simplified systems management of a sysplex, including configuration management, availability management, workload management, and single-image operations. Communication in a Parallel Sysplex uses two important components of z/OS:

- The XCF component of MVS provides coupling services for simplified multisystem management. Applications on one member of a sysplex use the XCF services of MVS to communicate with applications on the same system or on other systems.
- The XES component of MVS enables applications and subsystems to take advantage of the coupling facility.

Other components of the base operating system and several products running on these platforms are capable of using sysplex services. Among them are:

- JES2 or JES3 controlling job queues and dispatching work in a sysplex
- DFSMS enabling placement and storage management of data for MVS
- Networking software, which includes VTAM and TCP/IP
- z/OS SecureWay® Security Server RACF
- Database managers DB2 and IMS™ DB
- Transaction managers CICS TS and IMS TM
- WebSphere MQ for messaging and queuing

A sample configuration of a Parallel Sysplex using a variety of software is shown in Figure 2-4.

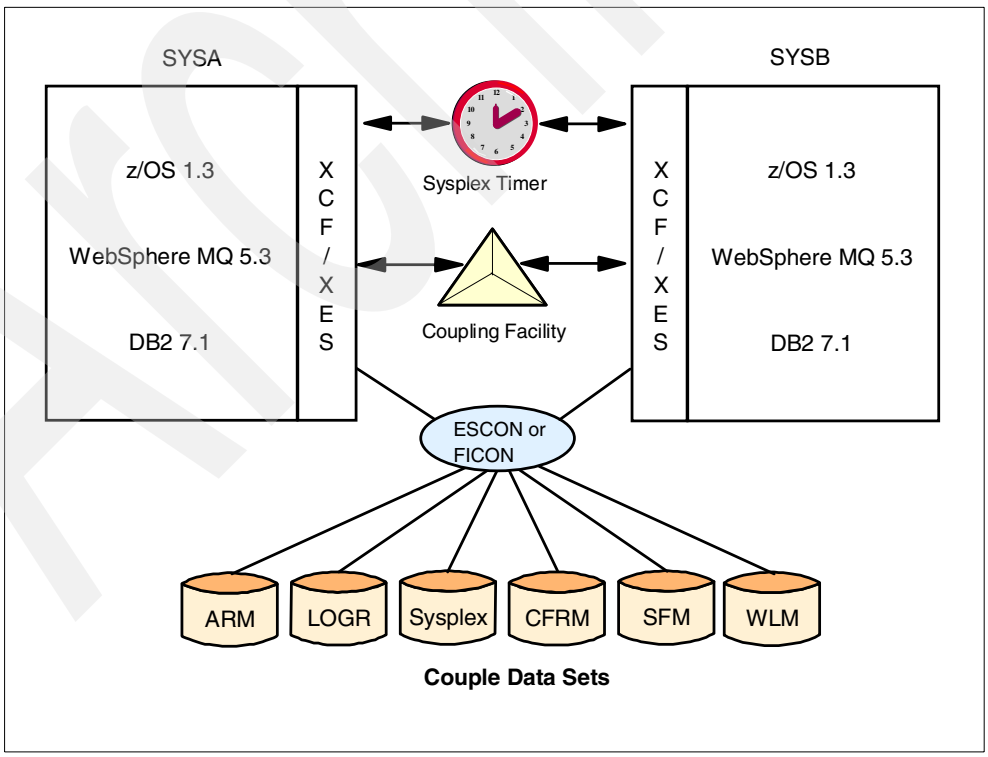


Figure 2-4 Parallel Sysplex configuration with software using sysplex functionality

2.1.3 SYS1.PARMLIB members used for sysplex setup

SYS1.PARMLIB members contain parameter values that MVS uses as input during system initialization to define the characteristics of the system. There are several SYS1.PARMLIB members containing sysplex-relevant parameters.

For an MVS system to run in a sysplex you have to update the following SYS1.PARMLIB members:

- IEASYSxx contains parameter values that control the initialization of an MVS system.
- COUPLExx contains values that are used to set up a sysplex. These values include, among other definitions, the name of the sysplex and the sysplex couple data sets, the name and type of other couple data sets, and the path definitions for sysplex communication.

Certain systems running in a PR/SM™ partition can specify values for the XCFPOLxx parmlib member.

Depending on the way you set up your sysplex, you might also need to change values in the following SYS1.PARMLIB members:

- IEASYMxx, where you can specify system symbols
- GRSCNFxx and GRSRNLxx, which contain definitions regarding the GRS
- CLOCKxx, which holds information on how to initialize the TOD clock during system initialization
- CONSOLxx, which contains console configuration values

There is planning information for these members in *z/OS MVS Setting Up a Sysplex,* SA22-7625 and corresponding reference information in *z/OS MVS Initialization and Tuning Reference,* SA22-7592.

2.1.4 Couple data sets

A sysplex requires at least one sysplex couple data set to store general information and status information about its members. In addition, the sysplex couple data set contains information about the XCF groups and members running in a sysplex.

The sysplex couple data set does not contain any policy specification. Depending on the policies you want to define, you might need to create additional couple data sets for CFRM, SFM, ARM, LOGR, and WLM policies.

To avoid a single point of failure, you should always use a primary and an alternate couple data set, which should be defined on a different device and control unit. Besides preventing a single point of failure, you can manage your couple data sets in a nondisruptive way in order to expand the size of the primary couple data set, or to move the couple data set to a different volume. The command SETXCF COUPLE allows you to make these changes dynamically.

Depending on the designated type of a couple data set, the couple data set must reside on a volume that is shared by some or all of the MVS systems in the sysplex. All sysplex couple data sets must be stored on DASDs shared by all the systems in the sysplex.

The couple data set format utility IXCL1DSU allows you to create couple data sets. The control statements for the utility program follow standard conventions for JCL statements. The utility has two levels of control statements:

- With DEFINEDS, the primary control statement, you identify the couple data set you want to format.

- With DATA TYPE, you specify the type of data to be supported in the couple data set. Data types can be sysplex data, ARM data, CFRM data, SFM data, WLM data, LOGR data, or z/OS UNIX System Services data.

Defining a sysplex couple data set

Figure 2-1 shows a sample of the IXCL1DSU utility to format a sysplex couple data set.

Example 2-1 Sysplex couple data set definition using IXCL1DSU

```
//IXCSYSDF JOB
//SYSDF001 EXEC PGM=IXCL1DSU
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  DEFINEDS SYSPLEX(WTSCPLX1)
           DSN(SYS1.XCF.CDS00) VOLSER(TOTDS0)
           MAXSYSTEM(16)
           CATALOG
  DATA TYPE(SYSPLEX)
           ITEM NAME(GROUP) NUMBER(200)
           ITEM NAME(MEMBER) NUMBER(203)
           ITEM NAME(GRS) NUMBER(1)
  DEFINEDS SYSPLEX(WTSCPLX1)
           DSN(SYS1.XCF.CDS04) VOLSER(TOTDS6)
           MAXSYSTEM(16)
           CATALOG
  DATA TYPE(SYSPLEX)
           ITEM NAME(GROUP) NUMBER(200)
           ITEM NAME(MEMBER) NUMBER(203)
           ITEM NAME(GRS) NUMBER(1)
/*
```

The data type for the sysplex couple data set must be SYSPLEX. For a sysplex couple data set, the following parameters are valid:

- SYSPLEX names the sysplex for which this couple data set is formatted
- DSN specifies the data set name of the couple data set
- VOLUME specifies the volume where the couple data set should reside
- CATALOG enforces the data set to be cataloged at allocation time
- MAXSYSTEM is the maximum number of systems in the specified sysplex

For the DATA TYPE(SYSPLEX), valid item names are GROUP and MEMBER. These parameters specify the maximum number of XCF groups and XCF group members for each group.

An *XCF group* is a set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex. Communication between group members on different systems occurs over the signaling paths that connect the systems; on the same system, communication between group members occurs through a local signaling service. To prevent multisystem applications from interfering with one another, each XCF group name in the sysplex must be unique.

In addition, the `ITEM NAME(GRS)` defines that this couple data set supports global resource serialization.

Defining couple data sets that contain policy data

The rules and actions that a sysplex is to follow are defined in *policies*. These allow MVS to manage specific resources in line with the requirements of a specific Parallel Sysplex setup. For a policy to be active in a sysplex, the couple data set containing this policy must be accessible to all systems that need to use it.

Except for the System Logger, the couple data sets can contain more than one administrative policy of more than one type. We recommend you use different couple data sets for different policy types.

Only one policy for each type can be active at any one time. The active couple data set contains both the administrative and the active policies. It also holds status data about the defined policies, and real-time information about the sysplex and the resources being managed by the particular policy.

Before you can define and activate a policy, you first have to format a couple data set with the utility IXCL1DSU, which is also used to format a sysplex couple data set. For a list of valid parameters of each type of couple data set, refer to Appendix B, "Format Utility for Couple Data Sets," in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

To define administrative policies, MVS provides the administrative data utility IXCMIAPU. IXCMIAPU allows you to create or delete policies, and to obtain a report of the contents of the policy data for all administrative policies defined in the couple data set. Example 2-2 shows a sample JCL to define an administrative policy in the couple data set for CFRM and two coupling facilities.

Example 2-2 CFRM policy and coupling facilities definition using IXMIAPU

```
//IXCCFRM  JOB
//CFRM0001 EXEC PGM=IXCMIAPU
//SYSPRINT DD  SYSOUT=A
//SYSABEND DD  SYSOUT=A
//SYSIN    DD  *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM087) REPLACE(YES)
  CF NAME(CF05)
     TYPE(009672)
     MFG(IBM)
     PLANT(02)
     SEQUENCE(000000050822)
     PARTITION(F)
     CPCID(00)
     SIDE(0)
     DUMPSPACE(2048)
  CF NAME(CF06)
     TYPE(002064)
     MFG(IBM)
     PLANT(02)
     SEQUENCE(000000010ECB)
     PARTITION(F)
     CPCID(00)
     SIDE(0)
     DUMPSPACE(2048)
```

Different types of policies and the sample jobs (which you can find in SYS1.SAMPLIB) to define and format the associated couple data sets are:

- CFRM policy, with sample jobs IXCCFRMF to define the couple data sets and IXCCFRMP to define the policy.

- Sysplex failure management (SFM) policy (sample jobs IXCSMFF and IXCSMFP).
- WLM policy. To define the couple data sets, you can use sample job IWMFTCDS. For the definition of policies, WLM provides an ISPF administration application.
- Automatic Restart Manager (ARM) policy (sample jobs IXCARMF and IXCARMP).
- System Logger (LOGR) policy. You can use sample job IFBLSJCL to define System Logger policies.
- UNIX System Services. You need a UNIX System Services couple data set when you implement shared HFS. To format the couple data set, you can use the sample job BPXISCDS. Since UNIX System Services use only XCF signaling services; you do not need any policy definitions.

2.1.5 Signaling

Signaling is the mechanism by which XCF group members communicate in a sysplex. In a multisystem sysplex, signaling can be achieved through:

- CTC communication connections
- A coupling facility (using coupling facility list structures)
- A combination of CTC connections and list structures

There must be an outbound and an inbound path between each pair of systems in a sysplex in order to achieve full signaling connectivity.

Signaling paths defined through CTC connections must be exclusively defined as either inbound or outbound. In a sysplex configuration using CTC communication connections with four members in the sysplex, each system needs at least two devices to connect to all other systems.

Figure 2-5 shows a sysplex with four systems, which are all using CTC connections with ESCON channels for signaling paths. Figure 2-6 shows the same scenario for a setup with FICON channels.

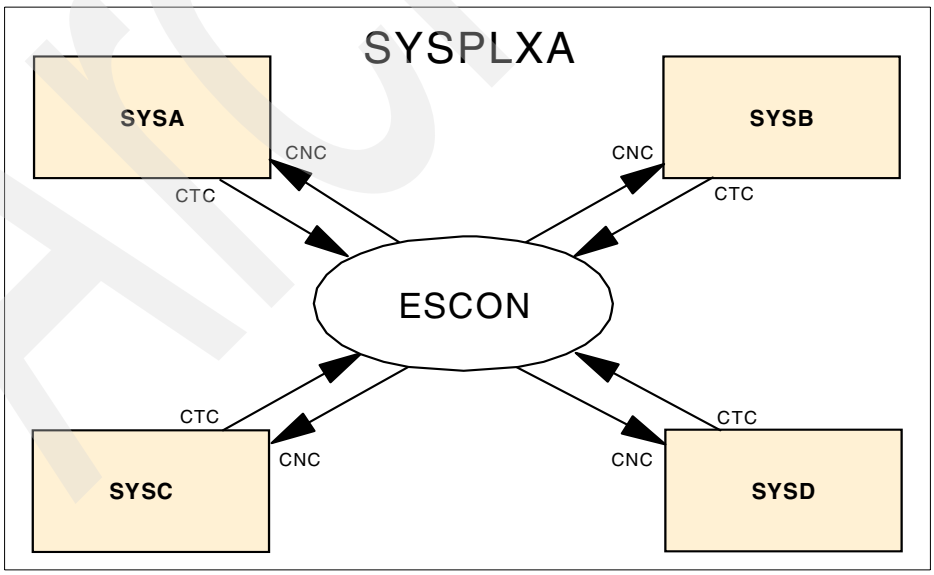


Figure 2-5 Signaling paths in a base sysplex with ESCON channel connections

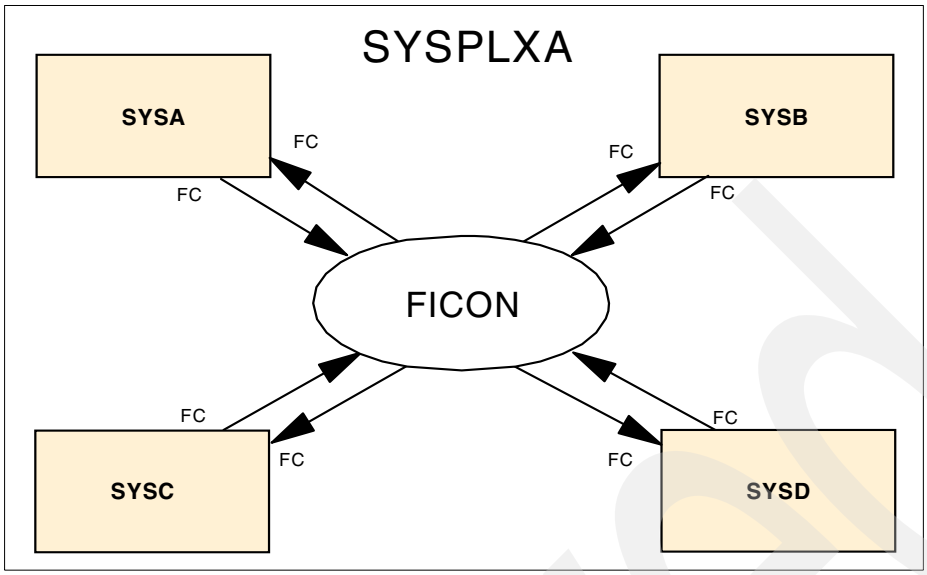


Figure 2-6 Signaling paths in a base sysplex with FICON channel connections

If you use ESCON channels, one channel has to be configured with type CNC, which means that the channel operates in native mode, and the associated channel at the other side has to be configured in channel-to-channel (CTC) mode.

If you use FICON channels, both channels are of type FC.

For both types, ESCON and FICON, the signaling paths are defined as equal and have to be in the parmlib COUPLExx member. Example 2-3 shows the PATHIN and PATHOUT statements required for this configuration.

Example 2-3 Signaling path definitions for a sysplex using CTC connections

```
/* * signaling paths for connection in sysplex SYSPLXA ************* */

   PATHIN  DEVICE(4013,4023,4033,4043)
   PATHOUT DEVICE(5013,5023,5033,5043)
```

In a Parallel Sysplex, signaling may be implemented through coupling facility list structures. Figure 2-7 shows a configuration using coupling facility list structures with the minimum required signaling paths.

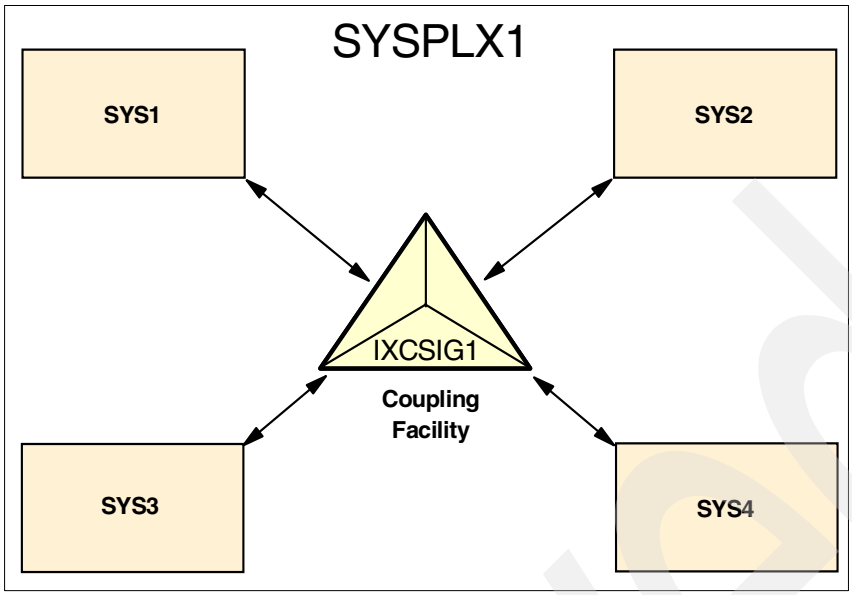


Figure 2-7 Signaling paths in a Parallel Sysplex using coupling facility list structures

With coupling facility list structures, you can use the same structure for inbound and outbound paths, and MVS automatically establishes the paths. For example, if you define a list structure for inbound traffic, MVS automatically establishes a signaling path with every other system that has the structure defined for outbound traffic. Example 2-4 shows the definitions of the necessary PATHIN and PATHOUT in COUPLExx parmlib member.

Example 2-4 Signaling path definitions using coupling facility list structure

```
/* * signaling paths to all systems in sysplex SYSPLX1 ************* */

   PATHIN  STRNAME(IXCSIG1)
   PATHOUT STRNAME(IXCSIG1)
```

2.1.6 Structures within the coupling facility

Storage within the coupling facility is divided into distinct objects called structures. The CFRM policy contains the structure definition. Structures are used by authorized programs to implement data sharing and serialization.

Figure 2-8 shows the three different structure types within a coupling facility.

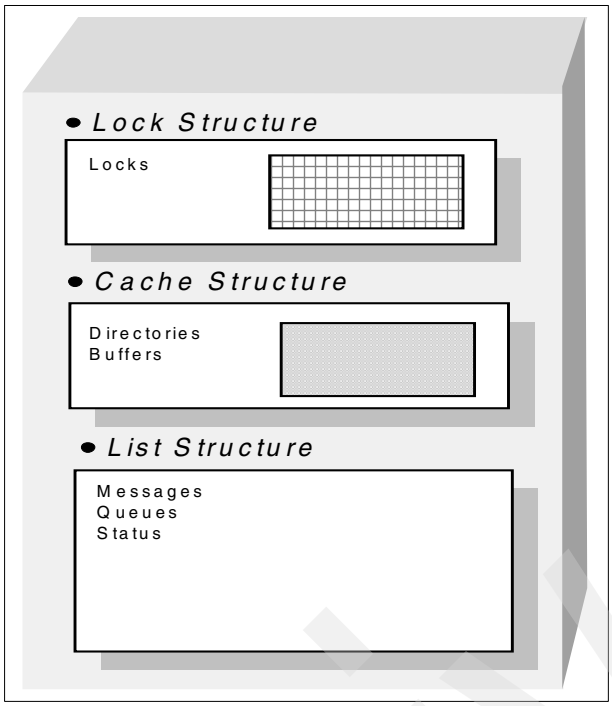


Figure 2-8 The three different structures in a coupling facility

Some storage within the coupling facility can also be allocated as dedicated dump storage for capturing structure information for diagnostic purposes. You can define only one dump space per coupling facility.

The characteristics and services associated with each structure support certain types of uses and offer certain unique functions.

Cache structure

Cache structures allow high-performance sharing of frequently referenced data. Cache structure services allow you to store and access data in the cache structure and to automatically notify affected users when you change shared data. You can also determine whether your copy of shared data is valid.

List structure

List structures enable users to share information organized as entries on a set of lists or queues. A list structure contains a set of lists. Each list contains a number of list entries. Each entry contains control information and data. List structure services allow you to read, write, move, and delete list entries.

Figure 2-9 shows the format of a list structure. MVS processes each IXLLIST request *atomically*, meaning the request cannot be interrupted. This ensures that the list structure cannot be viewed or accessed while it is being updated.

Lock structure

Lock structures allow users to serialize user-defined resources (including list or cache structure data) by creating a set of locks and locking protocols. Lock structure services allow you to associate user-specified data with each lock, implement locking protocols, resolve lock contention according to your own protocol, and recover locks as part of an overall recovery mechanism.

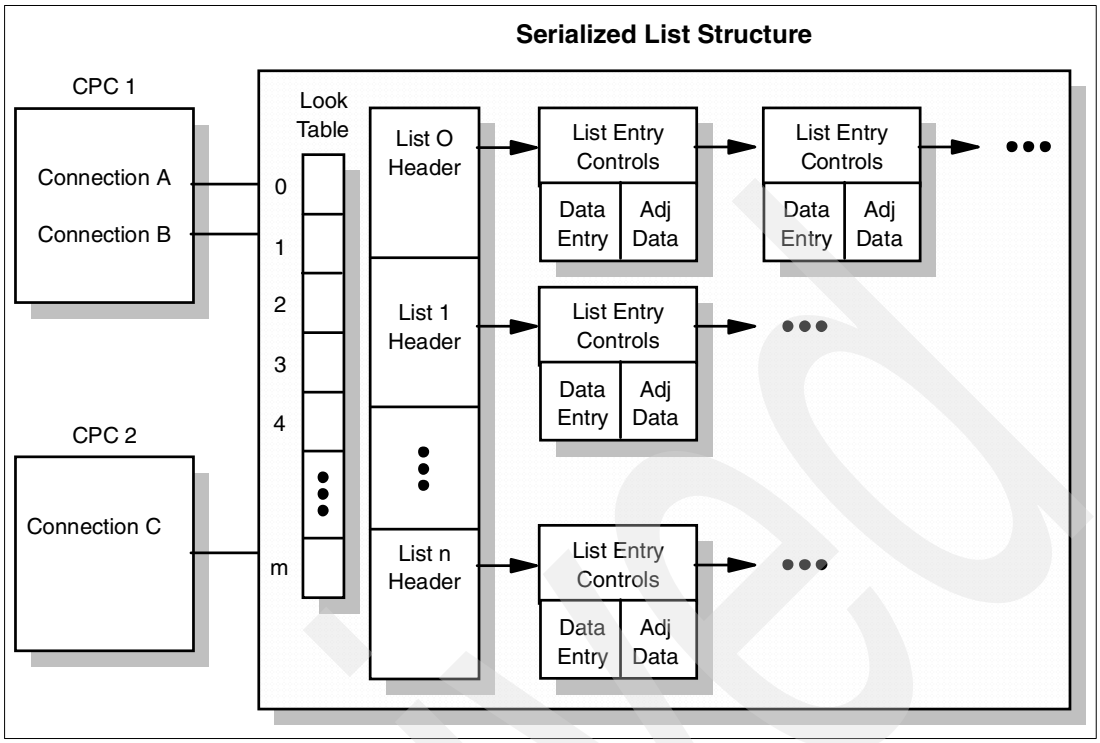


Figure 2-9 Serialized list structure

Each application specifies the structure types it requires. A coupling facility can support multiple coupling facility structures, but any one structure must reside entirely within a single coupling facility.

The size of each structure is specified in 1 KB blocks, and it is specified by the installation in its CFRM policy.

In order to size your structures appropriately, you can use the coupling facility structure sizer tool, which can be found at the following Web site:

`http://www.ibm.com/servers/eserver/zseries/cfsizer`

To get a list of structures defined in your coupling facility, you can use the command:

```
D XCF,STRUCTURE,STRNAME=strname
```

where *strname* represents a structure name, or `ALL`. Wildcard (*) suffixes are allowed.

2.1.7 Coupling Facility Resource Management (CFRM)

CFRM provides the services to manage coupling facility resources in a sysplex. This management includes the enforcement of CFRM policies to ensure that the coupling facility and structure requirements as defined in each policy are satisfied.

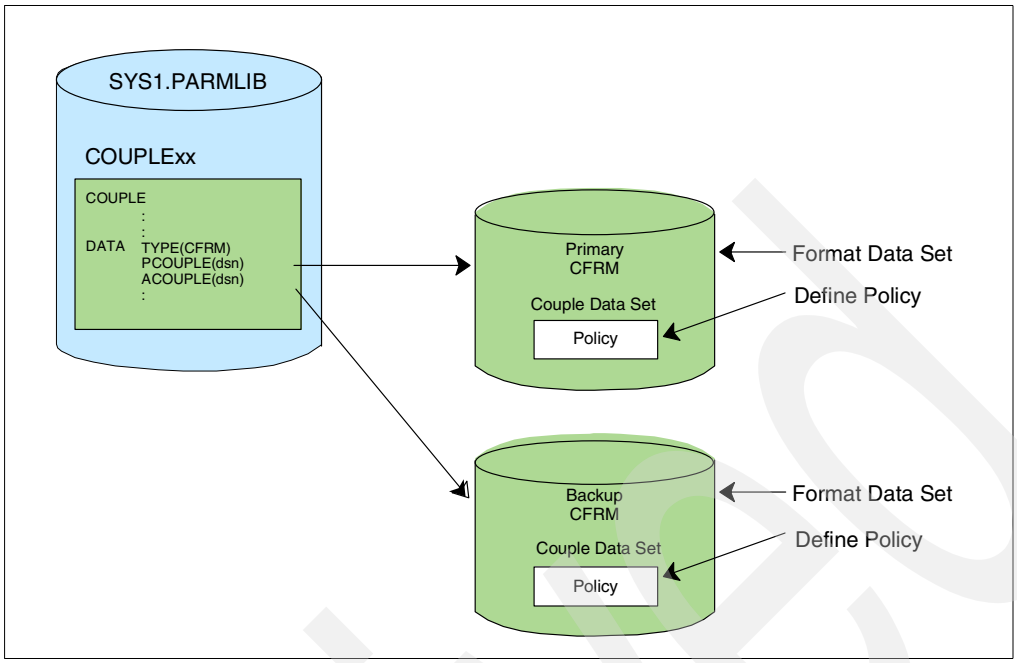


Figure 2-10 CFRM couple data set

CFRM uses the CFRM couple data set to maintain status data about the use of coupling facility resources in the sysplex and to hold its copy of the active CFRM policy. Whenever a CFRM couple data set is made available in the sysplex, either for the first time or when a switch is made to a different CFRM couple data set, CFRM becomes operational. Once operational, CFRM performs the following services for the sysplex:

- Cleans up status data
- Performs system-level initialization
- Reconciles active policy data with data in the coupling facility

Before you start to set up your CFRM policy, you need to know which subsystems and applications in your sysplex use the coupling facility and what the application's structure requirements are. The CFRM policy defines the amount of coupling facility storage used for each structure; it does not identify the type of the structure.

For more information on setting up the CFRM policy, see *z/OS MVS Setting Up a Sysplex,* SA22-7625.

2.1.8 Sysplex Failure Management (SFM)

Sysplex Failure Management allows you to define how system failures, signaling connectivity failures, and PR/SM (Processor Resource/Systems Manager™) reconfiguration actions are to be handled. The role of SFM is to isolate the failed component and reconfigure the sysplex so that units of work can continue. The goal of SFM is to allow these reconfiguration decisions to be made and carried out with little or no operator involvement.

The SFM data set contains all the information SFM will need to manage system and signaling connectivity failures. This includes:

- Policy statement
- Systems statements
- Reconfiguration statements

See Chapter 7, “Planning Sysplex Availability and Recovery,” in *z/OS MVS Setting Up a Sysplex,* SA22-7625, for more information on how to control system availability and recovery through the SFM policy.

2.1.9 Automatic Restart Manager (ARM)

Automatic Restart Manager is an MVS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention. The ARM policy specifies how batch jobs and started tasks that are registered with ARM should be restarted.

2.1.10 Workload Manager (WLM)

MVS Workload Manager provides dynamic sysplex-wide management of system resources. Each installation today processes different types of work with different response time requirements. With Workload Manager, you define a performance goal to each unit of work and assign a business importance to each goal. It is the job of WLM to attain these goals through the management and distribution of resources. Figure 2-11 shows a high-level overview of the workload management philosophy.

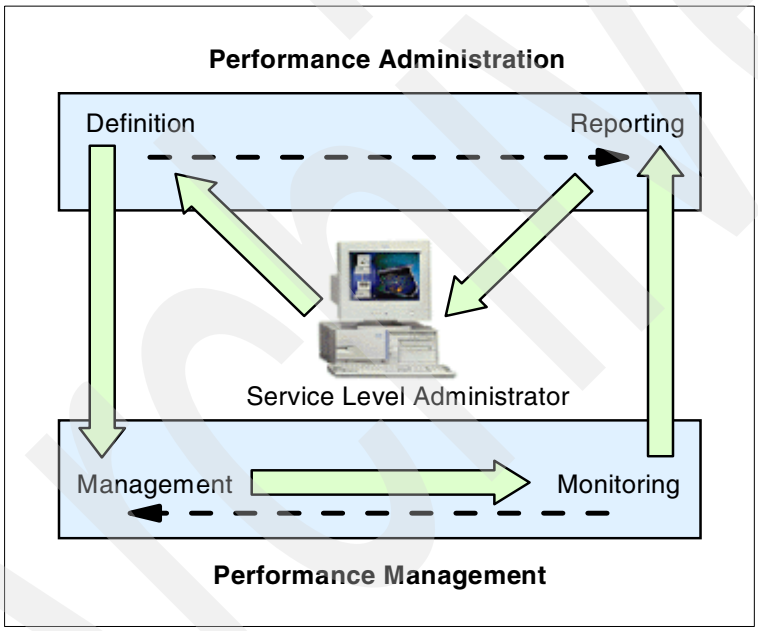


Figure 2-11 High-level overview of the workload management philosophy

For more information on WLM, see *z/OS MVS Planning: Workload Management,* SA22-7602.

2.1.11 MVS System Logger

The System Logger is a system component that allows applications to log data from a sysplex. WebSphere MQ does not use this facility to log its data. However, WebSphere MQ can participate in units of recovery managed by RRS, which uses the System Logger.

A System Logger application can write log data into a log stream, which is simply a collection of data. There are two types of log streams:

1. Coupling facility log stream, which is initially written to the coupling facility. When the structure becomes full, the data is moved to the DASD log data sets.

2. DASD-only log stream, which is first written to local storage buffers. When these buffers become full, log stream data is moved to the DASD log data sets. A DASD-only log stream can only be connected to one system at a time.

The System Logger component resides in its own address space, IXGLOGR, on each system in a sysplex. It maintains information about the current use of the log streams and coupling facility list structures in the LOGR policy. It is responsible for offloading the data from the coupling facility to DASD and automatically allocating new DASD logger data sets, when necessary.

The MVS System Logger manages the log streams based on the policy you defined in the LOGR couple data set. The LOGR policy includes log stream definitions.

For more information on the System Logger, refer to Chapter 9, “Planning for System Logger Applications,” in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

2.1.12 Global Resource Serialization (GRS)

In a multi-tasking, multi-processing environment, *resource serialization* is the technique used to coordinate access to resources that are used by more than one program. When multiple users share data, a way to control access to that data is needed.

GRS offers the control needed to ensure the integrity of resources in a multisystem environment, and provides the serialization mechanism for shared DASD across multiple MVS images.

Depending on the MVS operating system level and XCF environment the systems are running in, sysplex or non-sysplex, the complex consists of one or more systems:

- Connected to each other in a RING configuration through any of the following:
  - GRS-managed CTC adapters
  - XCF communication paths (CTCs)
  - Signaling paths through a coupling facility
- Connected to a coupling facility lock structure in a STAR configuration

The two types of GRS complexes are:

- Star

  In this type of GRS complex, all of the systems in the sysplex must match the systems in the GRS complex. This is sometimes called a GRS-plex.

- Ring

  In this type of complex, either:

  - The systems in the sysplex are the same as the systems in the complex (sysplex matches GRS complex).
  - At least one non-sysplex system in the GRS complex is outside of the sysplex (mixed complex).

GRS is required in a sysplex because components and products that use sysplex services need to access a sysplex-wide serialization mechanism. GRS must be active in each system within the Parallel Sysplex. Every system in a sysplex must be a member of the same GRS complex.

For more information on how to set up a GRS, see *MVS Planning: Global Resource Serialization,* SA22-7600.

2.1.13 Shared HFS

UNIX System Services provides mechanisms to make HFS data shareable for all sysplex users. With shared HFS, all file systems that are mounted by a system participating in shared HFS groups are available to all participating systems. In other words, once a file system is mounted by a participating system, that file system is accessible by any other participating system.

It is not possible to mount a file system so that it is restricted to just one of those systems. Consider a sysplex that consists of two systems, SYSA and SYSB:

- A user logged onto any system can make changes to file systems mounted on /u, and those changes are visible to all systems.
- The system programmer who manages maintenance for the sysplex can change entries in both /etc file systems from either system.

The UNIX System Services couple data set contains the sysplex-wide mount table and information about all participating systems, and all mounted file systems in the sysplex.

Shared HFS uses a sysplex root HFS to provide 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 version of HFS is the IBM-supplied root HFS data set. UNIX System Services creates a directory with the value *nnnn* specified on VERSION parameter in the BPXPRMxx member and uses this directory as the mount point for the specified version data set.

Figure 2-12 shows a sample setup for two systems using the same release level of the HFS data set.

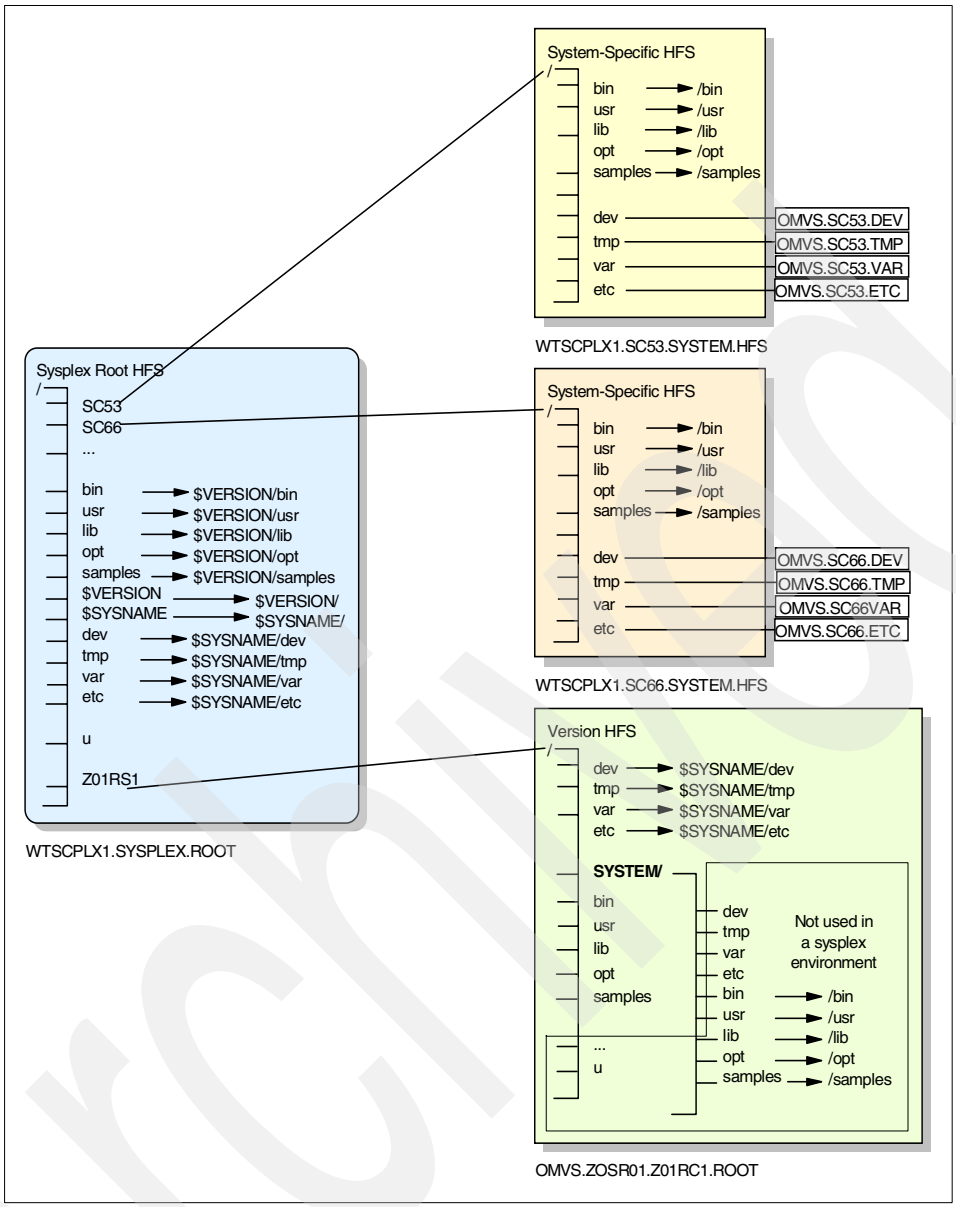


Figure 2-12 Shared HFS setup

Example 2-5 shows the changes to the BPXPRMxx member.

Example 2-5 Shared HFS setup in BPXPRMxx member with one version HFS for the 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')

.
.
```

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

2.2 Advantages of a Parallel Sysplex

Continuous availability of application

A single image, hardware or software, exposes you to system outages, whether planned or unplanned. Within a sysplex, all systems can have concurrent access to all critical applications and data. This enables systems to take over the work of a failing system by either restarting the applications of the failing system on other systems, or by redirecting work requests to other data-sharing instances of the subsystem on other systems.

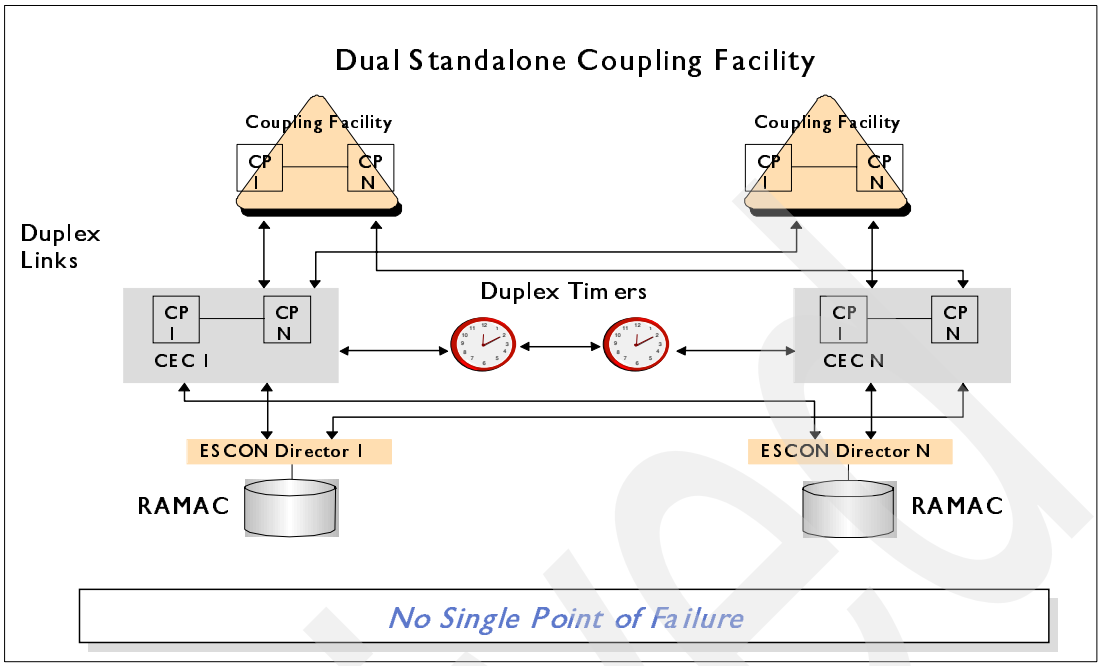


Figure 2-13 Parallel Sysplex technology: Continuous availability configuration

Reduction in planned outages

A Parallel Sysplex can make a planned outage less disruptive. One system can be removed and the work running on it will be routed to a remaining system *on the fly*. While the system is removed, new software or maintenance levels can be applied to this system. When reintroduced into the sysplex, the new software levels can coexist with any older levels on other systems within the sysplex.

Dynamic workload balancing

Dynamic workload balancing can be defined as the capacity to dynamically direct units of work requests to run on any or all of the systems within a sysplex. The entire Parallel Sysplex can be viewed as a single logical resource to end users and business applications.

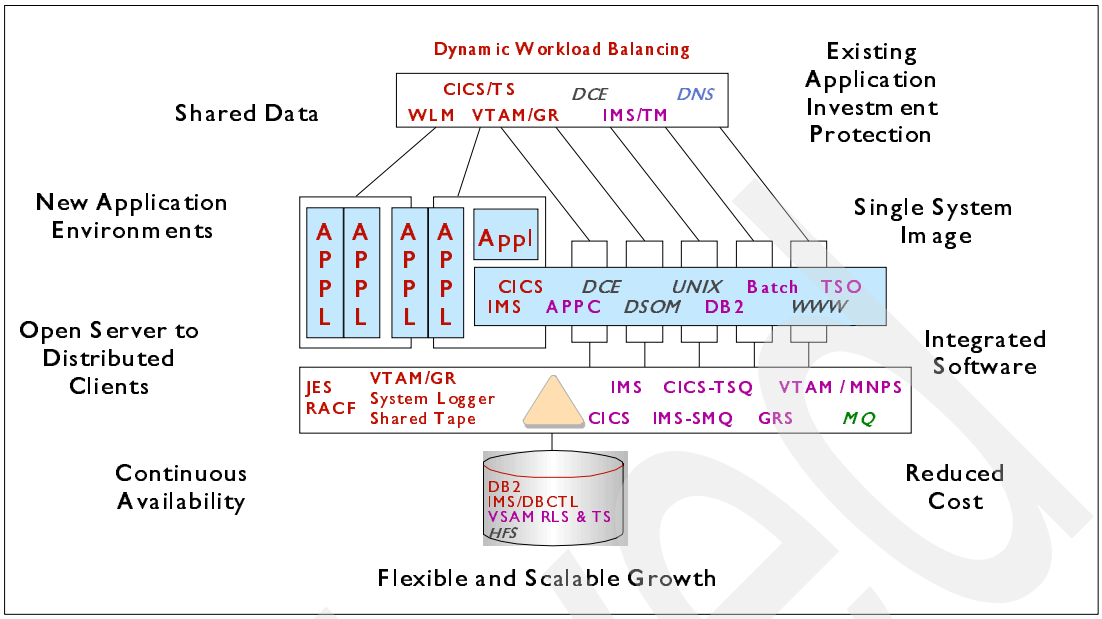


Figure 2-14 Parallel Sysplex technology: Workload balancing overview

Single system image

The collection of systems in a Parallel Sysplex appear as a single entity to operators, end users, and administrators. A single system image ensures reduced complexity from both operational and definition perspectives.

Scalability

You can increase the capacity of your system, and the throughput, by adding additional z/OS images to your existing configuration.

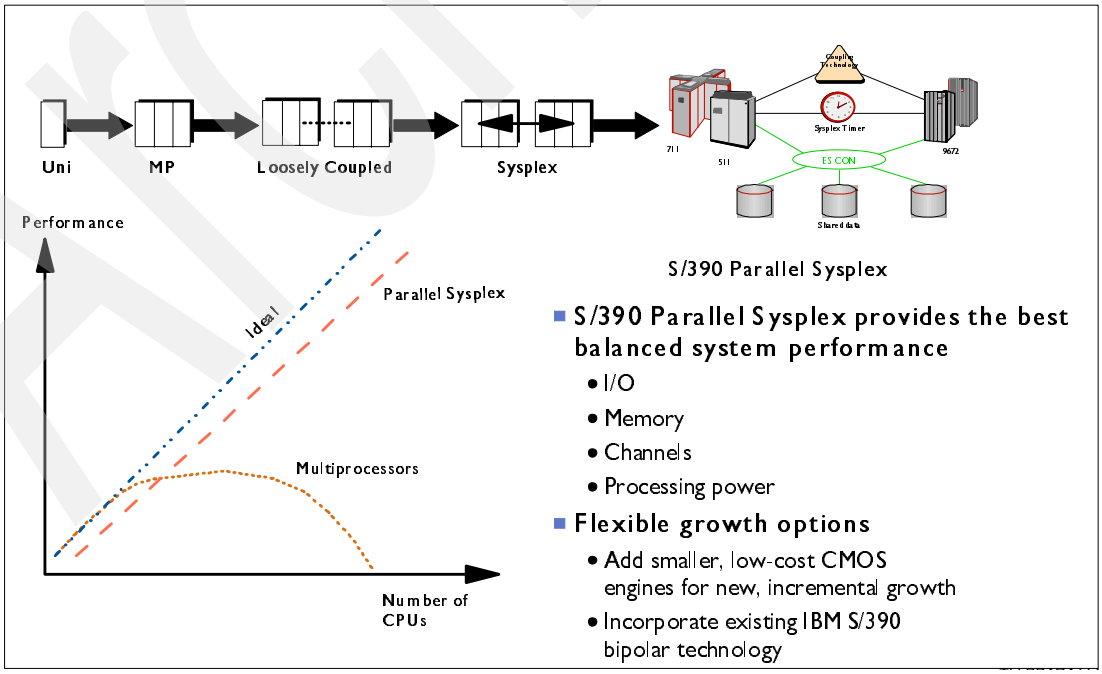


Figure 2-15 Parallel Sysplex technology: Scalability

2.2.1 Determining the appropriate number of Parallel Sysplexes

The number of Parallel Sysplexes needed in an installation will vary. For most installations, however, two Parallel Sysplexes is likely to be the norm. This should cover both the primary interface to the e-business application (a front end SYSPLEX) and the current users' production environments.

Among the issues to consider when deciding how many Parallel Sysplexes to implement are the following:

- Ease of operation and management

  It is easier to operate and manage a single large Parallel Sysplex than many smaller ones. If improved availability is one of your goals, this should not be overlooked; more outages are caused by human error than by hardware failures.

  Assuming that you have a test sysplex, which we recommend, the test sysplex should be used as a learning environment to get familiar with the aspects of managing a sysplex, and also as a place to test commands and procedures before they are used on the production system.

- Cost

  In nearly all cases, the cost will increase relative to the number of Parallel Sysplexes you create. Remember that CF partitions cannot be shared between different sysplexes, and that each Parallel Sysplex will need its own dedicated links between OS/390 images and CFs that are not in the same CPC. With G5 and later, it is possible to share an ICF CP between several CF LPs; however, remember that each CF LP must have its own dedicated CPC storage.

- Protecting your production systems

  There is a certain amount of risk associated with putting development systems into the same Parallel Sysplex as your production systems.

  The amount of this risk will vary from installation to installation. If your developers have a talent for bringing the development system down every other day, then it may be wiser to isolate them from the production systems. On the other hand, if you have a properly configured test Parallel Sysplex, and have thorough test suites, then the risk should be negligible. If you have not implemented a thorough test suite, then multi-system outages or problems may be encountered in the production environment rather than on the test systems. Some installations would therefore prefer to thoroughly test new software releases that have sysplex-wide effects (such as JES, XCF, and so on), in a less availability-critical environment.

  In this situation, you may prefer to keep the development environment in a sysplex of its own. The *system programmer test sysplex* should always be kept separate from the production sysplex. The use of VM is one way of providing a self-contained environment where the configuration can be changed with just a few commands, and mistakes cannot impact a system outside the VM environment.

- The scope of work for batch, CICS and IMS transactions, shared data, physical location and shared DASD

  If there are systems that share nothing with other systems, maybe for business or security reasons, it may make more sense to place those systems in separate Parallel Sysplexes.

3

Running ICSF in a Parallel Sysplex environment

In this chapter we discuss sharing ICSF PKDS and CKDS datasets in a Parallel Sysplex environment and provide an overview of the related constraints and recommendations. Further details on CKDS and PKDS sharing and related operating procedures are in *z/OS ICSF Administrator's Guide*, SA22-7521.

3.1 zSeries integrated cryptography review

IBM led the industry by offering the first *CMOS cryptographic coprocessor* as a standard feature on the S/390 G4, G5, G6, and zSeries servers in order to meet the increasing need for data security and integrity.

The S/390 Cryptographic Coprocessor Facility is implemented in CMOS technology on a single chip, providing more capability than any previous cryptographic offering. Included in the design is battery-backed non-volatile memory storage, laser delete chip personalization, integrated tamper detection and response, and high speed DES, Triple DES, DSS, RSA, Pseudo Random Number Generation, and hashing algorithms. Depending on the system model, there may be one or two coprocessor chips in operation.

The software that enables this solution and provides the application programming interface (API) is a follow-up release of the ICSF/MVS product. This release has been integrated into the base of OS/390 since V2R5.

Starting in June 2000, the PCI Cryptographic Coprocessor (PCICC) has been orderable feature that adds additional cryptographic function and cryptographic performance to G5/G6/zSeries servers. Up to 8 PCICC features may be ordered for a G5, G6, or zSeries server. Each PCICC feature comprises one 4758 Technology-based cryptographic coprocessor card embedded in an adapter package for installing within a G5 or G6 server; for zSeries, one PCICC feature comprises two 4758-based coprocessors. Support for PCICC is provided in OS/390 V2R9 by new ICSF functions.

The PCI Cryptographic Accelerator (PCICA) is the latest cryptographic coprocessor. This new addition to the mainframe cryptographic hardware is only available on IBM zSeries processors and is supported beginning with z/OS V1R2. There can be no more than 6 PCICA crypto features per server. PCICA is another adjunct crypto coprocessor designed for exploitation by SSL.

Having one or more PCICA features in addition to the CCFs and one or more PCICCs will ensure very high throughput for SSL-based transactions at the same time non-SSL crypto workloads are processed. Applications that call ICSF directly for “clear key” RSA operations will also transparently use the zSeries PCI Cryptographic Accelerator feature. The PCICA feature supports all public key sizes up to 2048 bits. The PCICA cryptographic hardware feature is designed to perform a very limited set of functions to support SSL cryptographic functions. No data privacy, financial, or key management operations are included in the PCICA design.

The PCI Cryptographic Coprocessor and PCI Cryptographic Accelerator features coexist with and augment CMOS CCF functions. ICSF transparently routes application requests for cryptographic services to one of the integrated cryptographic engines, either a CCF or a PCICC or PCICA, depending on performance or requested cryptographic function. For example, RSA signature generation and verification operations (with 1024-bit or shorter keys), such as those typically used by SSL, are routed to both CCF and PCICC engines. On the other hand, RSA Key Generation is performed only on PCICC engines, and the PCICA is dedicated to very computing-intensive public key operations that have to be performed during the SSL handshake.

3.1.1 zSeries integrated cryptography implementation

Figure 3-1 provides a high-level representation of how the cryptographic coprocessors work in zSeries.

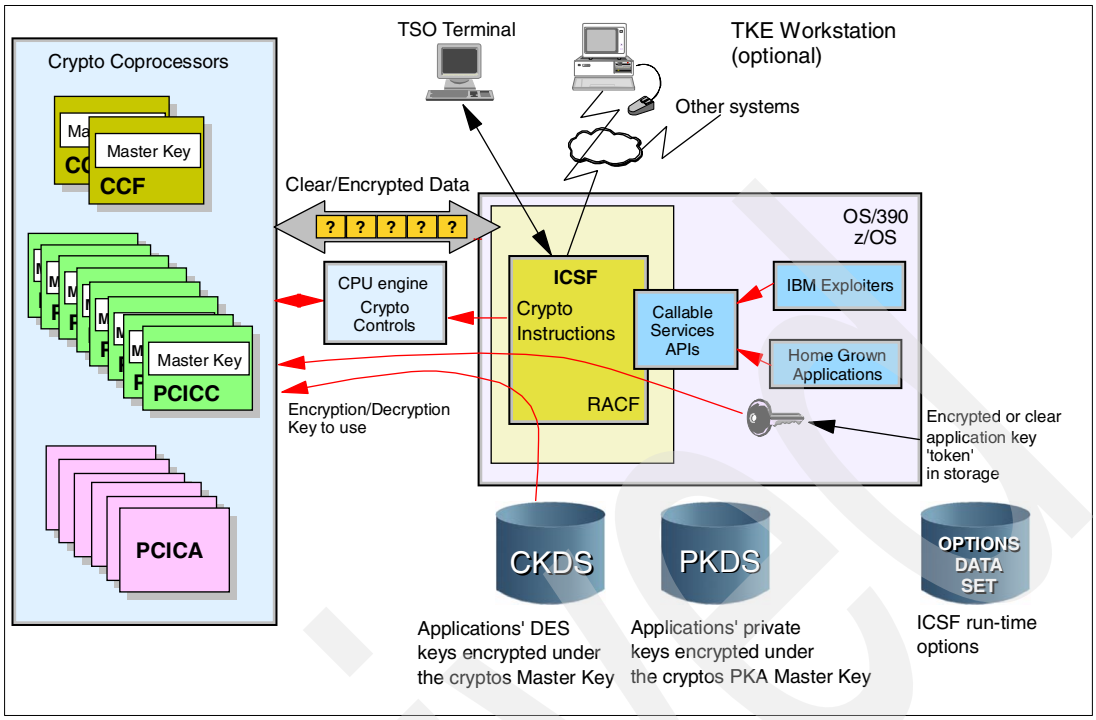


Figure 3-1 S/390 Cryptographic Coprocessors

The process flow is as follows:

- The exploiters of the cryptographic services call the ICSF API. Some functions will be performed by the ICSF software without invoking the cryptographic coprocessor; other functions will result in ICSF going into routines containing the proprietary S/390 crypto instructions.
- These instructions are executed by a CPU engine and result in a work request being generated for a cryptographic coprocessor.
- The crypto coprocessor is provided with the following:
  - Data to encrypt or decrypt from the system memory.
  - The key used to encrypt or decrypt provided by ICSF as per the exploiter's request.

    Physically, these keys can be stored in ICSF-managed VSAM data sets and pointed toby the application using the label they are stored under. The Cryptographic Key Data Set (CKDS) is used to store the symmetric keys in their encrypted form, and the Private Key Data Set (PKDS) is used to store the asymmetric keys. The application also has the capability of providing an encrypted encryption key or a clear encryption key directly in memory (that is, to use *as is*) to the coprocessor.

Note: These keys are represented as sealed envelopes here when provided to the CCF or PCICC cryptographic coprocessors. This graphic was selected to stress the fact that these encryption/decryption keys are themselves encrypted and, therefore, unusable when residing outside of the crypto coprocessor.

Important: The PCICA coprocessor is always provided with clear application keys; therefore, it does *not* implement the Master Key concept explained in the next section.

– For high-speed access to symmetric cryptographic keys, the keys in the CKDS are duplicated into an ICSF-owned data space, the “in-storage CKDS,” and, optionally, the key kept in the PKDS can also be stored in an in-storage cache beginning with z/OS V1R2.

– The Trusted Key Entry (TKE) Workstation is an optional, priced feature. This workstation provides a secure, remote, and flexible method of providing Master Key Part Entry and remotely managing the cryptographic coprocessors. The algorithm utilizes Digital Signature, Diffie-Hellman, and DES functions to provide a highly secure, auditable, and remote method of key entry, and it can be used by those customers requiring very high security for key entry.

3.1.2 The Master Key concept

The *Master Key* is installed by security officers in the cryptographic coprocessors of types CCF and PCICC; as already mentioned, the PCICA does not have any Master Key. This provides the ultimate security in a zSeries crypto installation in that it resides only in the coprocessors, with very strong built-in physical security, and it is used to protect other secrets, meaning the encryption/decryption keys used by applications. The application’s keys are intended to be kept encrypted under the Master Key in the two specialized VSAM data sets: CKDS and PKDS. Whenever an application wants the coprocessor to use one key, it provides to ICSF the label of the key stored in the CKDS or PDS; ICSF fetches this key encrypted with the Master Key and it is decrypted inside the coprocessor just before being used, as shown in Figure 3-2. The application issues a request to ICSF with the cryptographic function to be performed (a data encryption function in this illustration), the CKDS or PKDS label of the key to use, and the data to be encrypted, and it receives in return the encrypted data.

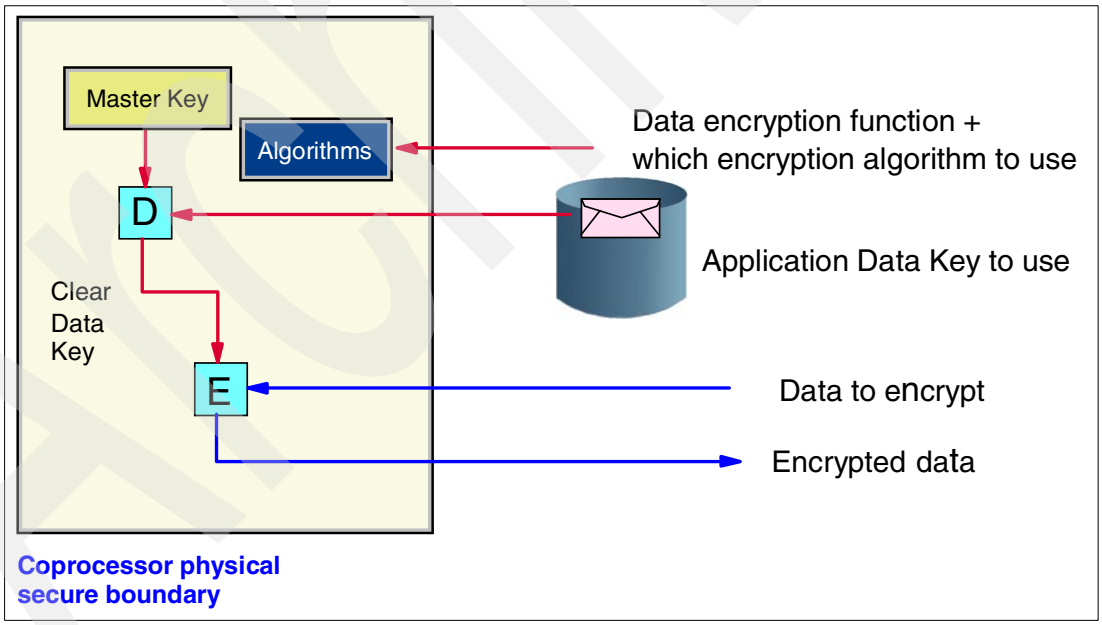


Figure 3-2 The Master Key concept

Note: The actual implementation of the Master Key concept in the zSeries cryptographic coprocessor ends up to be:

- Three Master Keys in the CCF:
  - One to protect the symmetric keys
  - Two, recommended to be of identical value, to protect the asymmetric keys (two keys because of implementation historical reasons).
- Two Master keys in the PCICC, one to protect the symmetric keys (set at the same value as the symmetric Master Key in the CCF), and one to protect the asymmetric keys, set at the same value as the two asymmetric keys in the CCF.

For the sake of simplicity in the rest of this chapter we will only refer to “a” Master key.

3.1.3 LPAR domains and TKE

The physical cryptographic coprocessors can be shared between logical partitions. Selecting one or two CCFs in the Processor page of the image profile allows further LPAR definitions pertaining to the CCF, the PCICC, and the PCICA. Refer to *zSeries Processor Resource/Systems Manager Planning Guide*, SB10-7033 for more details.

A cryptographic coprocessor of type CCF or PCICC actually has 16 physical sets of Master Key registers, each set belonging to a *domain*. A domain is allocated to a logical partition via the definition of the partition in its image profile; the same domain must also be allocated to the ICSF instance running in the logical partition via the ICSF Options Data Set. The PCICA does not use a domain.

Figure 3-3 illustrates how logical partitions, physical crypto coprocessors, and domains interact.

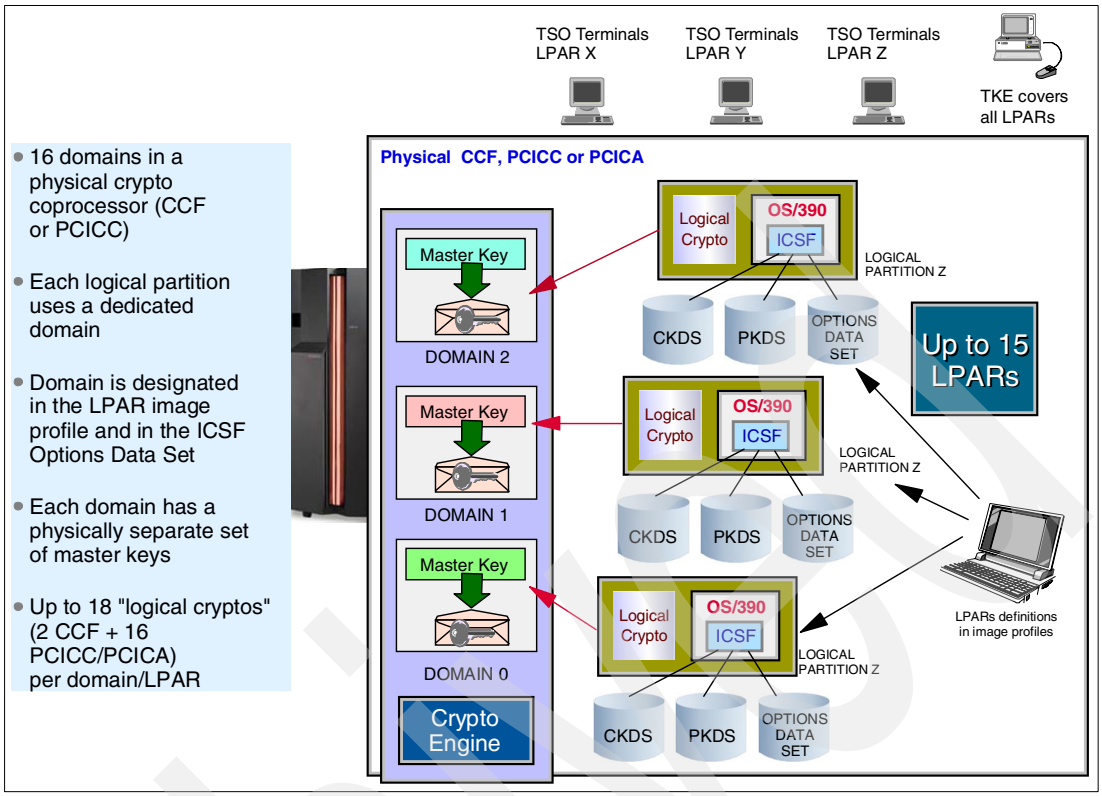


Figure 3-3 Physical coprocessor domains and logical coprocessors

- Each ICSF instance accesses only the Master Keys corresponding to the domain number specified in its image profile, at the system Service Element, and in its Option Data Set. Each ICSF sees a *logical crypto coprocessor* made up of the physical cryptographic engines shared among the logical partitions and the unique set of registers (the domain) dedicated to this partition.
- Each logical partition or each ICSF instance, therefore, has its own unique Master Key values and can encrypt its own set of users' keys with these unique Master Key values, thereby insuring perfect insulation between users of the shared crypto coprocessors from the standpoint of security.

Cryptographic services management through the TKE

The *Trusted Key Entry (TKE) workstation* provides a centralized control point for administering the multiple physical or logical crypto coprocessors (CCF and PCICC) in a single or multi-system configuration. The TKE workstation provides a way of entering key values into the coprocessors securely across a public network (be it LAN or WAN). The utilization of the TKE implies conversations between the TKE application running in the workstation and a dedicated piece of hardware logic inside the CCF and PCICC crypto co-processors. These conversations are made of commands and responses which constitute the *Public Key Secure Cable (PKSC)* functions.

> **Note:** Sensitive messages exchanged between the crypto coprocessor and the TKE application at the workstation are digitally signed for integrity and encrypted for confidentiality

The TKE communicates with one z/OS instance called the *TKE Host System*. If the TKE Host System is a logical partition, several domains, corresponding to other partitions in this

physical system, can be controlled from the Master Keys' administration standpoint by a single TKE. This is achieved securely by having the TKE supporting software in the TKE host issue proprietary calls to the PR/SM microcode for cross-domain Master Keys administration. This cross-domain capability is available only when using the TKE Workstation, as opposed to the ICSF ISPF panels. However, ICSF ISPF panels can still be used to administer the local Master Keys, whether a TKE is present in the configuration or not. The administration of multiple domains via the TKE is shown in Figure 3-4.

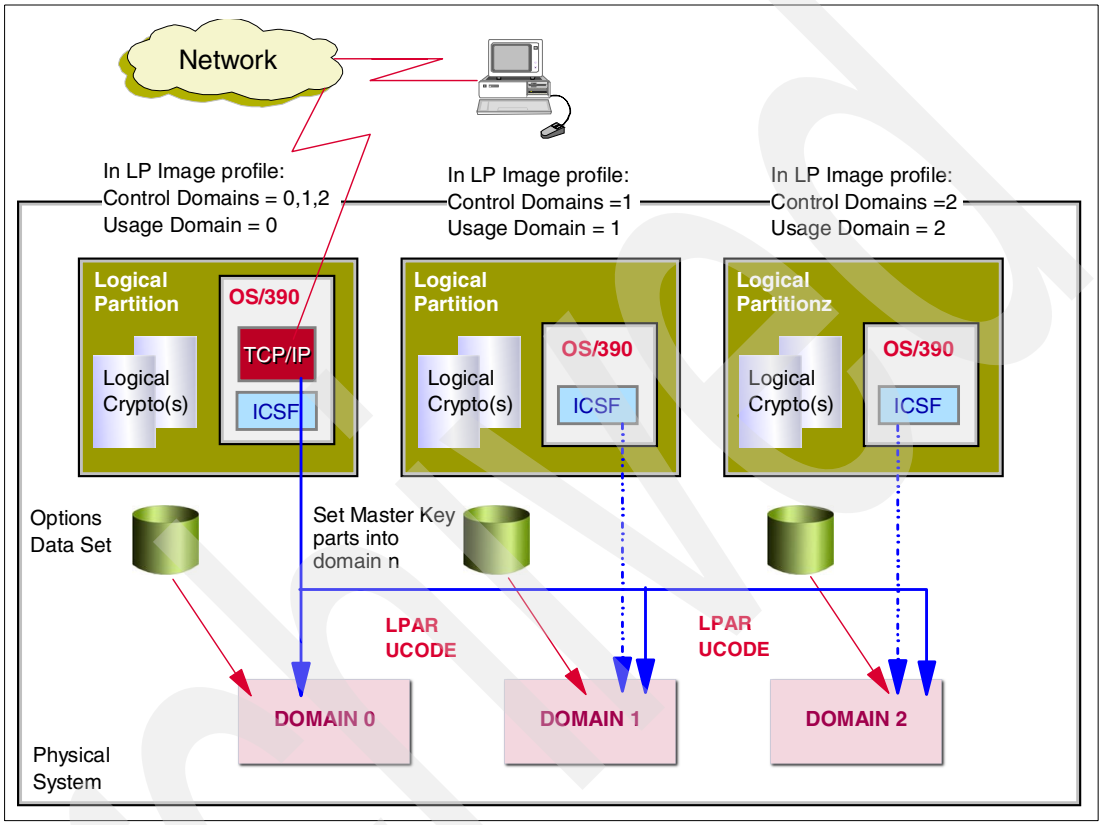


Figure 3-4 Multiple domains administration with the TKE

3.2 Sharing of CKDS and PKDS

The keys kept in the CKDS (symmetric keys) and the PKDS (asymmetric keys) are managed by ICSF and have the following characteristics:

1. They are kept in a format called the IBM CCA (Common Cryptography Architecture) *key token*.
2. The key kept in the key token is encrypted by the Master Key.
3. All key tokens residing in the same CKDS or PKDS are encrypted with the same value of the Master Key, and to be usable, this value is also the value of the Master Key currently set in the coprocessors.
4. For performance reasons the content of the CKDS is brought into a data space (*in-storage CKDS*) at ICSF startup; optionally, buffers can also be defined to keep in storage the PKDS keys (*PKDSCACHE*). ICSF is then looking for the application keys it needs in the in-storage copies of the CKDS and PKDS.

5. Everything we are referring to here does *not* apply to the PCICA coprocessors since they do not use any Master Key (they are provided always with the clear value of the application keys).

For all of these reasons, a single CKDS and PKDS instance can be shared between several ICSF instances provided—because of point 3—that all involved crypto coprocessors' domains are set with the same Master Key value, which is also the Master Key value used to encrypt the application keys in these very CKDS and the PKDS. This is illustrated in Figure 3-5 for the case of several logical partitions sharing the same CKDS and PKDS.

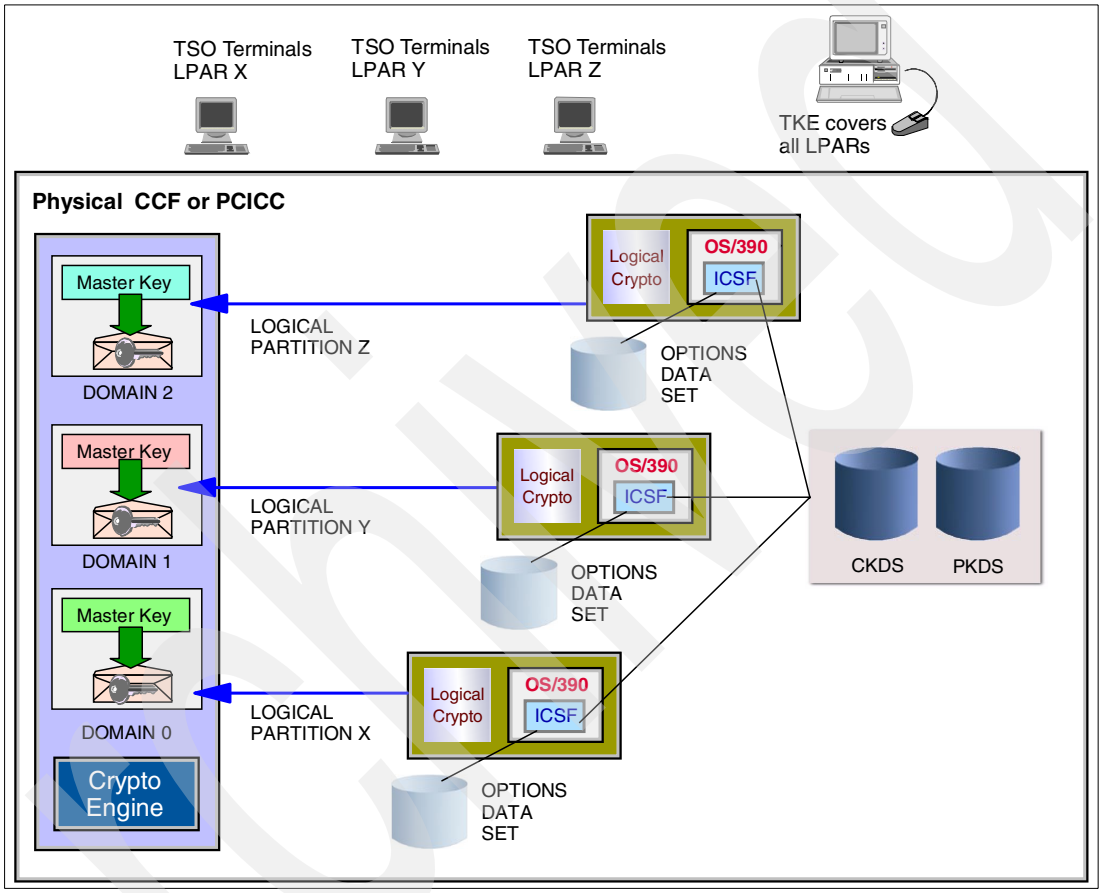


Figure 3-5 Sharing the CKDS and PKDS

Maintaining the in-storage copies of the CKDS and PKDS keys

Once the CKDS and PKDS contents are brought into storage, there is no automated mechanism that keeps the in-storage copies synchronized with the VSAM dataset contents; however, there are tools which assist in this process. An example is the KGUP utility program, which can be used to maintain symmetric keys in the CKDS dataset. KGUP updates the keys directly in the CKDS disk and the update is not reflected to the in-storage copy of the CKDS. It takes a manual CKDS refresh via the ICSF ISPF panel or a scheduled CKDS refresh via the CSFEUTIL utility program to synchronize the contents of the in-storage CKDS with the CKDS disk contents. The PKDS cache is also intended to be refreshed via the ISPF panels or the CSFPUTIL utility program.

When the CKDS and PKDS are shared between several ICSF instances this shows up as an exploitation problem: a local update, by an application, to the CKDS or PKDS will be reflected into the local in-storage copies and into the VSAM dataset. However, the other ICSF

instances will keep running using their own in-storage copies without any awareness of the change in the dataset.

In the case of the Parallel Sysplex, where one would expect sharing of the CKDS and PKDS between the sysplex ICSF instances, this leads to the constraints and recommendations discussed in the next section.

3.3 Sharing CKDS and PKDS in a sysplex

ICSF is currently not sysplex-enabled, but can run in a sysplex environment. The suggested method is to install the same Master Keys in all crypto-enabled members. The CKDS and PKDS can then be shared across the sysplex. This, however, is not mandatory. Each image may have its own Master Keys and its own CKDS, as shown in Figure 3-6.

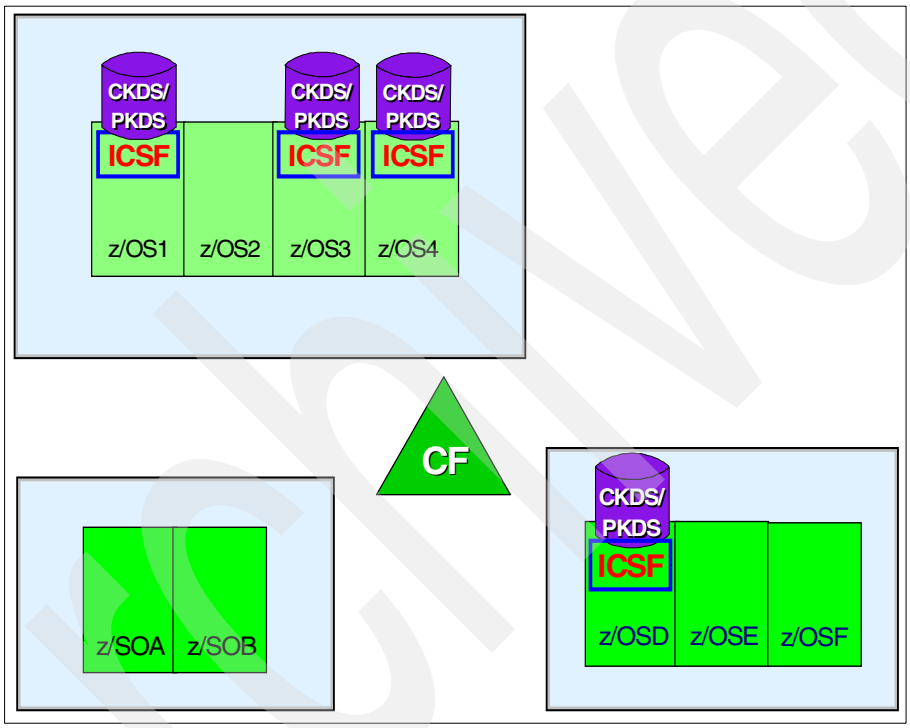


Figure 3-6 Non-sharing of CKDS and PKDS

If the user elects to share the CKDS and PKDS (Figure 3-7 on page 82), all coprocessors driven by the ICSF instances sharing the CKDS and PKDS must have the same Master Keys (so that any coprocessor in the configuration will be able to use any key token). A few additional precautions should be observed as well, specifically:

- Dynamic CKDS and PKDS services update the DASD copy of the CKDS and PKDS and the in-storage copies on the member where the update is performed. There is no SYSPLEX broadcast of the update. In order to update the in-storage copies of all members that share the CKDS and PKDS, you must perform a CKDS and PKDS REFRESH on each member.
- The PKDS may be shared between sysplex members that do not all have access to a PCICC. Some PKDS key tokens can only be handled by a PCICC coprocessor, as indicated in the ICSF Administrator's Guide and the ICSF Application Programmer's Guide. This will induce an operational affinity with particular members of the sysplex.

- Changing master keys should be carefully planned in a SYSPLEX environment as discussed in the next section. Keep in mind, however, that a Master Key change is a very rare event, probably done only once every one or two years.

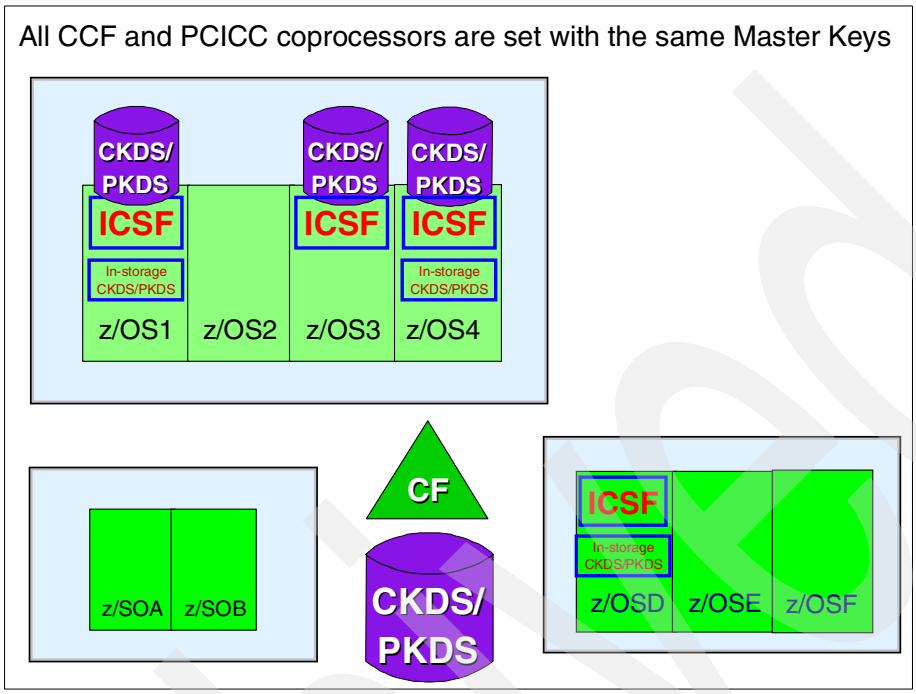


Figure 3-7 Sharing of CKDS and PKDS in the sysplex

3.3.1 Miscellaneous sysplex ICSF issues

Most simple case

No key updates, neither Master Keys nor application keys in key data sets. Applications which use the crypto-callable services can only run successfully on a member that has a crypto coprocessor active.

In a sysplex, this can be accommodated by building a router that associates the group of members which are crypto-capable with the group of resources which need to use the crypto functions. The router can be implemented, for instance, in CICS/TS; or, for batch jobs, via the Scheduled Environment facility of WLM.

Master Key change

The sequence of a Master Key change at the sysplex level is:

- To change the asymmetric (PKA) Master Key:
  a. Disable the PKA services in all the sysplex members. (First, stop your cryptographic applications if they cannot stand getting a return code 12 with reason code 11024 from ICSF).
  b. Change the PKA Master Key in all sysplex members.
  c. On one member, perform or schedule (with CSFPUTIL) a PKDS re-encipher and activate.
  d. On the other members, perform or schedule a PKDS cache refresh against the just re-enciphered PKDS.

e. Once all the members have performed the PKDS activate:

   i. Update the ICSF Options Dataset to point to the new PKDS.

   ii. Re-enable the PKA services in all members.

- To change the symmetric (DES) Master Key:

  Here there is no way to disable the symmetric encryption services, and you must be careful about a timing conflict between the Master Key change and cryptographic applications executing in the sysplex that would exchange key tokens. As you will see, a Master Key change in the sysplex is a rolling process among the crypto-enabled members and you do not want to have an application getting a key token for which the encrypting Master Key is not set yet. The safest way to proceed is during off-cryptographic workload hours.

  a. Load the new symmetric Master Key in all the domains in the sysplex.

  b. In one sysplex member, perform or schedule (with CSFEUTIL) a CKDS re-encipher.

  c. In all sysplex members, perform or schedule a Master Key change pointing at the just re-enciphered CKDS.

Application key changes with KGUP

This pertains to the symmetric keys kept in the CKDS, where KGUP maintains the keys in the VSAM dataset without propagating the updates to in-storage CKDS copy. In that case:

1. Assess any out-of-sync conditions that could occur with your applications, like asking for a key not yet generated or asking for a key already deleted, and put in place proper timing of the operations.
2. Run the KGUP updates on one member.
3. Perform or schedule (with CFSEUTIL) a CKDS refresh on all crypto-enabled members in the sysplex.

Application key changes by applications

This is a more tricky case than an update with KGUP since you may ignore, at a point in time, which applications, if any, have generated, updated, or deleted the shared key tokens in your sysplex. Remember that in this case the local in-storage copy of the CKDS or PKDS will be updated, along with the VSAM dataset, but this update will not be propagated to the other sysplex members' in-storage copies. There is not really a solution here except to individually manage applications maintaining keys and to schedule a CKDS or PKDS cache refresh whenever appropriate.

Some complex cases

- A symmetric Master Key change occurs while an application generates or imports a key token. When a symmetric Master Key is changed, the latest current key is kept as the "old Master Key" and is used by ICSF in case of an out-of-sync condition: if a key cannot be decrypted using the current symmetric Master Key, then automatically a try is made using the old Master Key. If that works the key token is used and re-encrypted under the current Master Key value and the application is warned about this condition with return code 0 and reason code 10000. Applications must, therefore, be coded to accept these codes.

  Another way of addressing this out-of-sync condition is to disable the CKDS dynamic accesses (in the ICSF administrative controls ISPF panel) while performing the symmetric Master Key change in the sysplex. All applications attempting to manage symmetric application keys will be denied access to the CKDS until the dynamic updates are again enabled.

Note: The asymmetric Master Key change implies the systematic disabling of the PKA services during the update.

- Key tokens are not kept in the CKDS or PKDS; they are therefore not handled by the ICSF re-encipher function and may be found to be no longer usable if transmitted to members which already changed their Master Key (with the exception of matching the previous symmetric Master Key for symmetric key tokens, as explained in the previous case). One classic approach to handle this off-board storing of key tokens is to encrypt the key token with an importer key, itself kept and maintained in the CKDS, as opposed to encrypting the key token with the Master Key. Such a key token is referred to as an *external key token* and is fully explained in the ICSF Application Programmer's Guide.

4

Exploitation and protection of the coupling mechanisms

In this chapter we discuss how to exploit and protect the coupling mechanisms.

We describe the potential users of the sysplex resources, namely:

- XCFAS address space
- IXGLOGR address space
- Sysplex resources administrators

The sysplex resources we discuss are:

- Coupling facility structures
- Sysplex timer
- Couple datasets
- iodf
- Sysplex operator commands
- System logger resources
- IXCMIAPU utility functions

4.1 Coupling facility structure

As documented by the subsystem or application using a coupling facility structure, the security administrator might have to define security profiles for certain structures. If the OS/390 Security Server, which includes RACF, or any other security product is installed, the administrator can define profiles that control the use of the structure in the coupling facility.

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

1. Define resource profile IXLSTR.structure-name in the FACILITY class.
2. Specify the users who have 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 in-storage profiles are maintained for the FACILITY class, refresh them.

Figure 4-1 shows an example of a sysplex with a coupling facility.

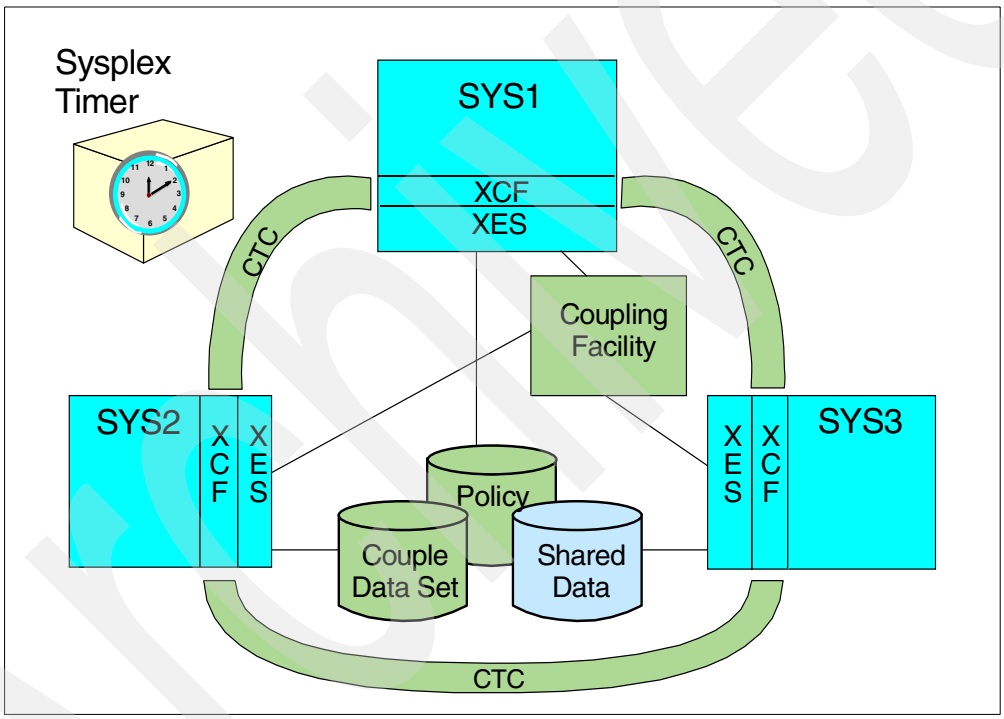


Figure 4-1 Sysplex with a coupling facility

You can specify RACF userids or RACF groupids on the ID keyword of the PERMIT command. If RACF profiles are not defined, the default allows any authorized user or program (supervisor state and PKM allowing key 0-7) to issue coupling facility macros for the structure.

The security administrator may protect the integrity of the data within the structure before the following coupling facility requests are issued:

- `IXLCONN`
- `IXLREBLD`
- `IXLFORCE`

You need to know the structures used in your configuration and the userid associated with the connectors. You can use the following command to collect information on your configuration:

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

Example 4-1 shows sample output from this command.

Example 4-1 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
```

We can distinguish the structures as belonging to a resource-sharing or data-sharing configuration environment.

4.1.1 Resource sharing

This environment is characterized as one in which the Coupling Facility technology is exploited by key OS/390 or z/OS components to significantly enhance overall Parallel Sysplex management and execution.

A number of base z/OS or OS/390 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 of z/OS or OS/390 software.

Typical Resource Sharing structures include XCF, GRS, ECS (Catalog), JES2 Checkpoint, RACF, and system logger.

Figure 4-2 on page 88 shows an overview of a typical Resource Sharing environment

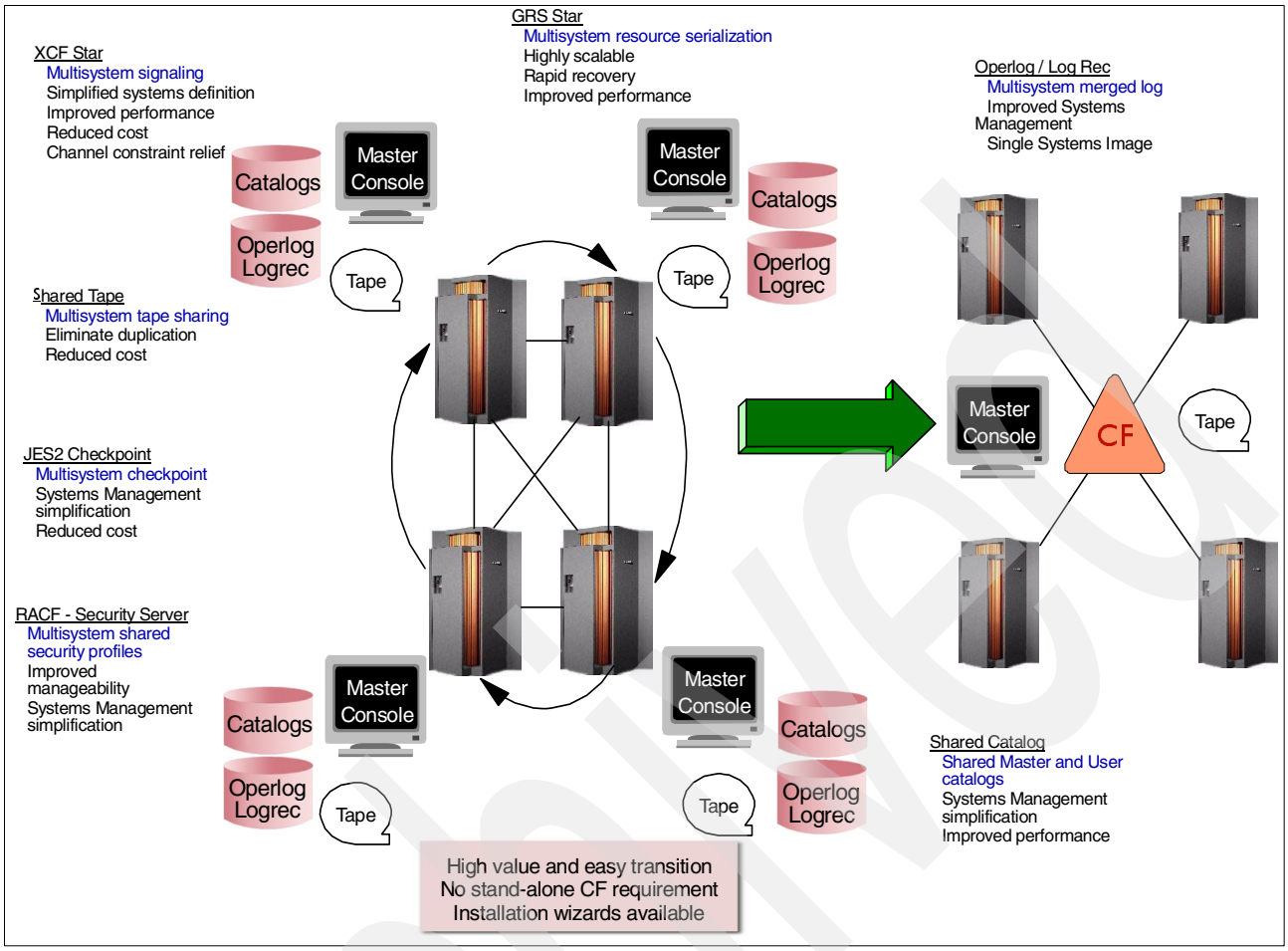


Figure 4-2 Resource Sharing functions and values

The connectors are most often system address spaces such as XCFAS or GRS.

Example 4-2 shows how to define XCFAS in the RACF STARTED class and associate a specific user-id with the XCFAS address space.

Example 4-2 Define XCFAS started class

```
AU XCFUSER NAME('XCF USER') DFLTGRP(SUPMVS) OWNER(SUPMVS) NOPASSWORD
RDEF STARTED XCFAS.* OWNER(SUPMVS) UACC(READ)   +
STDATA(USER(XCFUSER) GROUP(SUPMVS) TRUSTED(YES))
SETR RACLIST(STARTED) REFRESH
```

Example 4-3 shows how to protect XCF structure.

Example 4-3 Define XCF structure protection in FACILITY class

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

Figure 4-3 on page 89 shows an example of Resource Sharing mode.

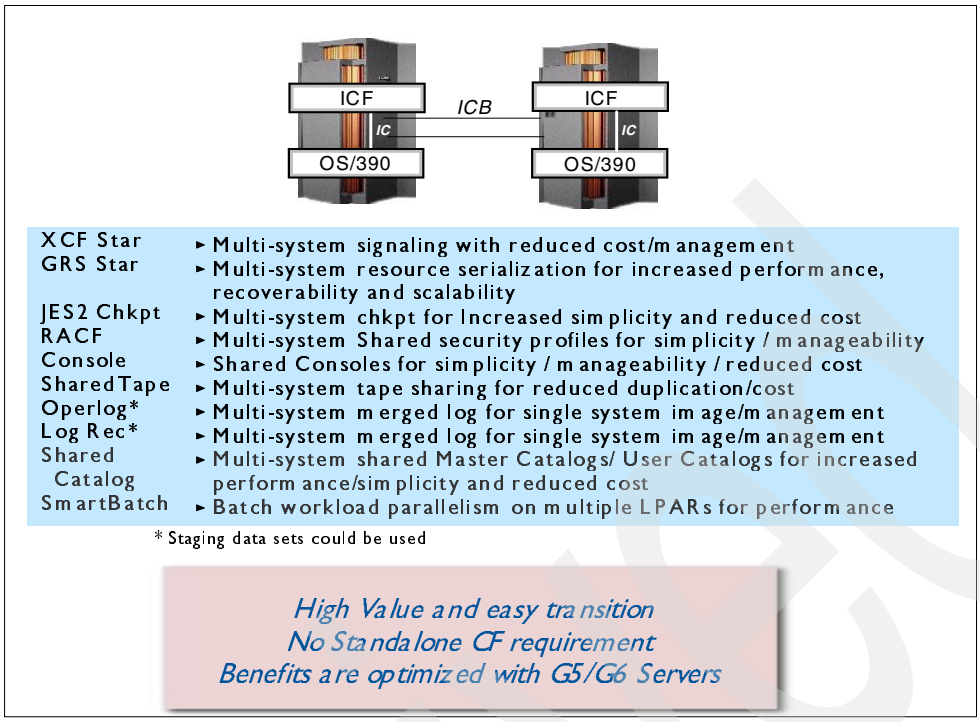


Figure 4-3 Resource Sharing mode

Note: It is recommended that your RACF support personnel define the XCFAS in the RACF STARTED class or the started procedures table with the trusted attribute.

4.1.2 RACF data sharing

When RACF is enabled for Sysplex communication, it uses XCF to join the RACF data-sharing group, IRRXCF00. The data-sharing group facilitates communication between systems in the sysplex enabled for sysplex communication. There is only one data-sharing group per sysplex.

See Figure 4-4 on page 90 for an overview of RACF data sharing.

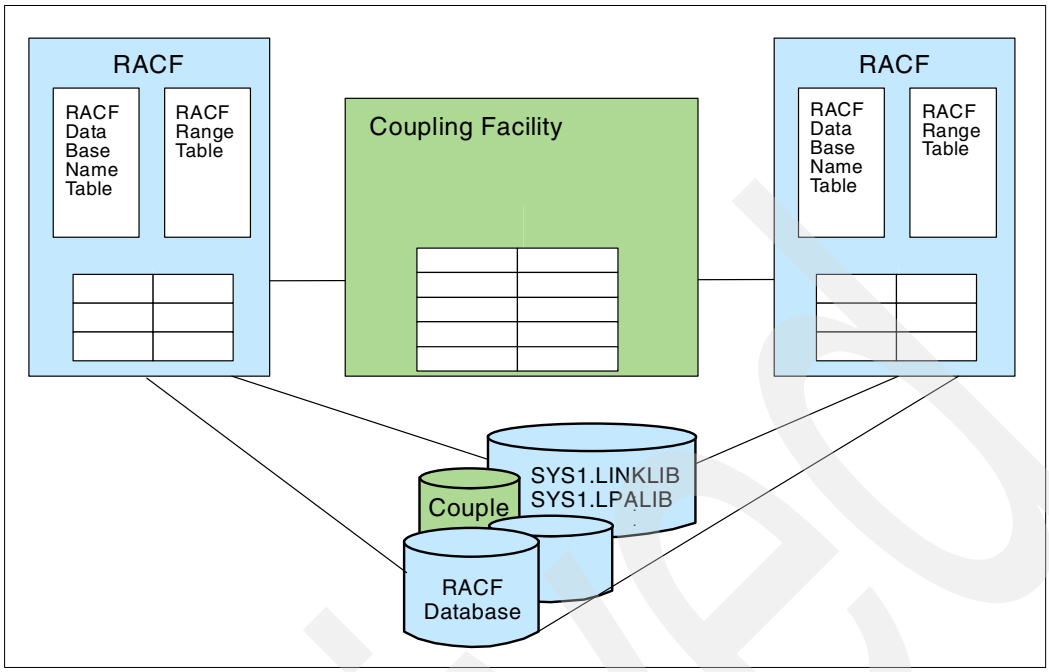


Figure 4-4 RACF Data sharing

When RACF is enabled for sysplex communication, you have the ability to enter certain commands that now affect the whole sysplex, thereby simplifying security management. The commands that can be entered once and then propagate through the rest of the sysplex are the following:

- RVARY SWITCH
- RVARY ACTIVE
- RVARY INACTIVE
- RVARY DATASHARE
- RVARY NODATASHARE
- 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 4-5 shows an example of how command propagation works.

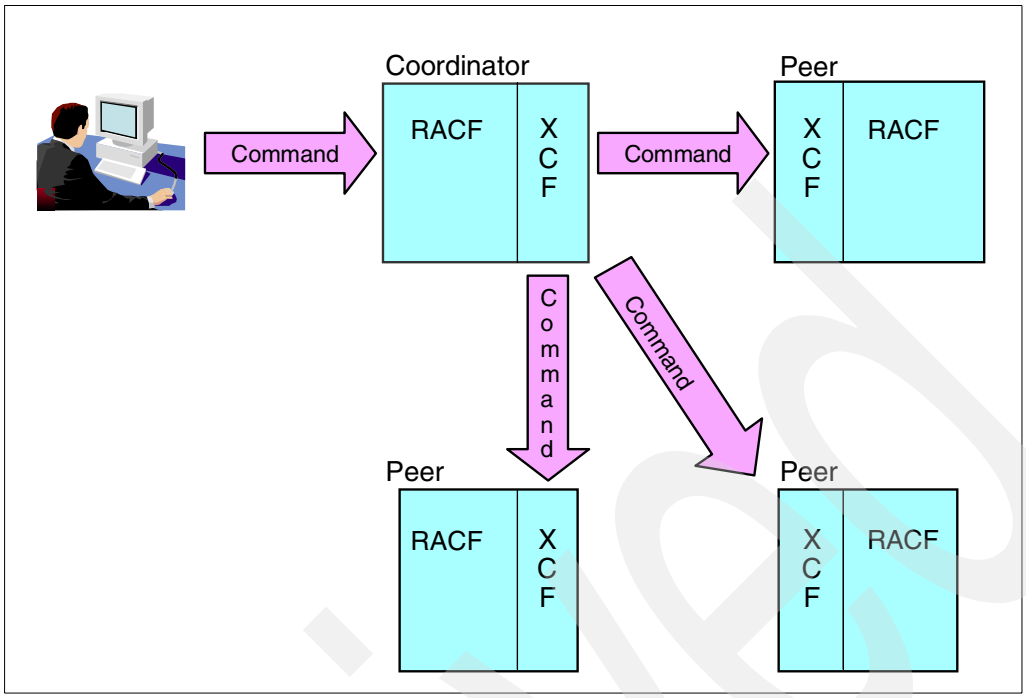


Figure 4-5 RVARY and SETROPTS command propagation

4.1.3 Data sharing

The application-enabled Parallel Sysplex environment represents 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 are sub-system address spaces, like DB2 for example.

Example 4-4 shows the command necessary to protect a DB2 structure.

Example 4-4 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)
SETR REFRESH RACLIST(FACILITY)
```

Figure 4-6 on page 92 shows an example of VSAM data sharing.

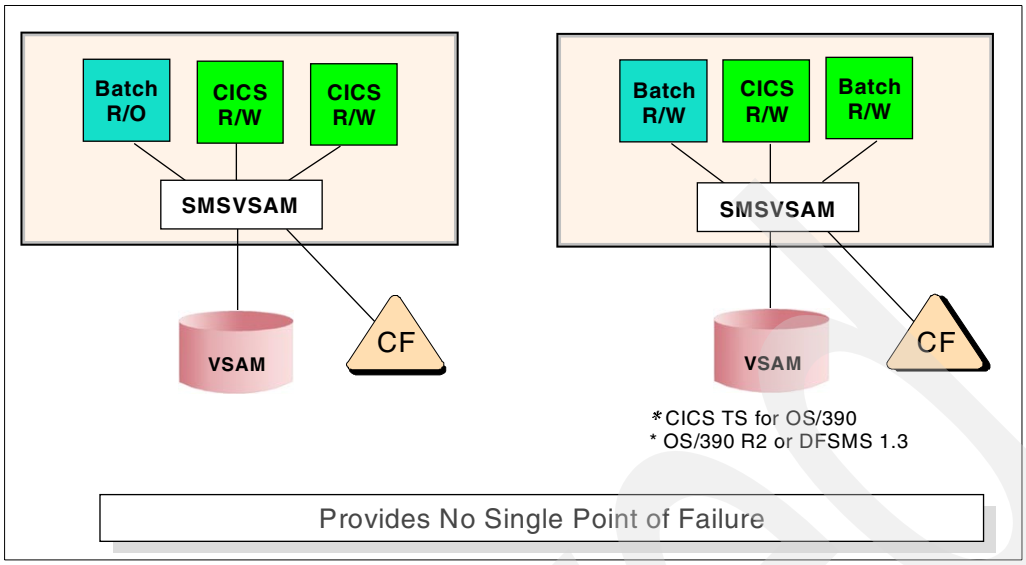


Figure 4-6 VSAM data sharing mode

4.2 Couple data sets

It is the responsibility of the installation to provide the security environment for the couple data sets. Consider protecting the couple data sets with the same level of security as the XCF address space (XCFAS). All these control files must be RACF-protected UACC(NONE) and authorized only to XCFAS.

Note: For z/OS and OS/390 systems, XCF does not need to be explicitly authorized to access couple data sets.

4.2.1 Sysplex files

Actual names can be found in SYS1.PARMLIB(COUPLExx) or through the D XCF,COUPLE command:

- SYSPLEX couple datasets (primary and alternate)
- ARM couple datasets (primary and alternate)
- BXMCDS couple datasets (primary and alternate)
- CFRM couple datasets (primary and alternate)
- LOGR couple datasets (primary and alternate)
- SFM couple datasets (primary and alternate)
- WLM couple datasets (primary and alternate)

System logger files

System logger allocates VSAM linear data sets for the DASD log data sets and DASD staging data sets.

Figure 4-7 on page 93 shows an example of how the System logger works.

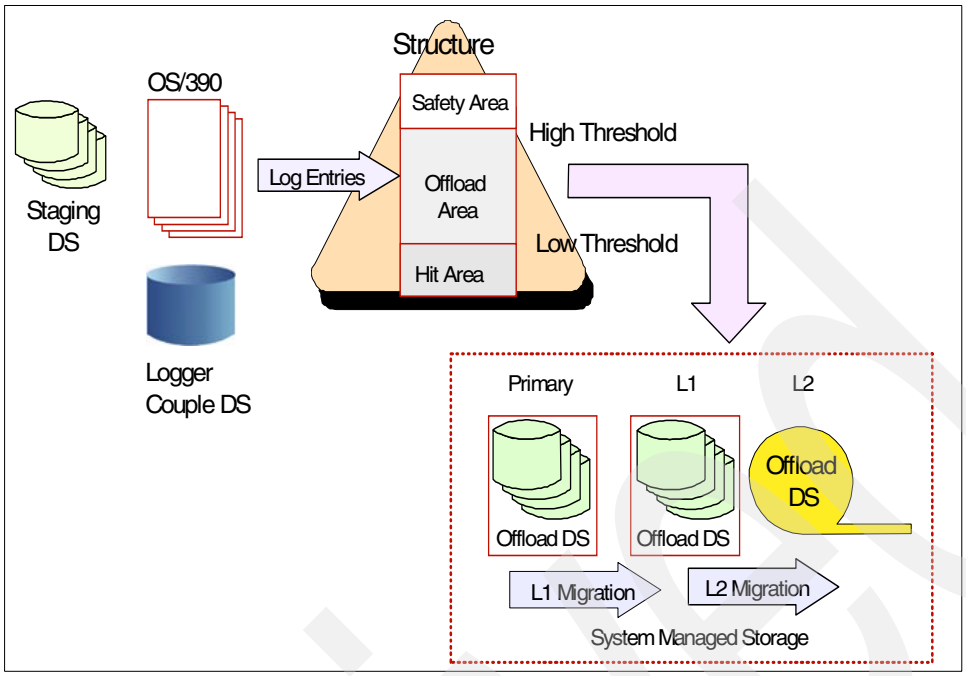


Figure 4-7 System Logger example

System logger does no 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 plan carefully 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! It is recommended 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 high-level qualifier is specified in DEFINE LOGSTREAM of the IXCMIAPU utility; this is illustrated in Example 4-5.

Example 4-5 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 following qualifiers are built according to logstream names, which can be listed with the D LOGGER,LOGSTREAM command.

Example 4-6 Output from D LOGGER,LOGSTREAM command

```
SYSPLEX.OPERLOG            SYSTEM_OPERLOG   000001 IN USE
   SYSNAME: P0S1
     DUPLEXING: LOCAL BUFFERS
```

The resulting logstream and staging dataset name are shown in Example 4-7 (where P0S1 is the system name).

Example 4-7 Resulting LOGSTREAM and STAGING dataset

```
MVSLOG.SYSPLEX.OPERLOG.A0000000
MVSLOG.SYSPLEX.OPERLOG.A0000000.DATA
MVSLOG.SYSPLEX.OPERLOG.P0S1
```

Note: It is recommended that system logger address space IXGLOGR be defined as a started task with the trusted attribute MVSLOG.** profile in class DATASET; or a more specific profile must be created with UACC(NONE). 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.

4.2.2 Authorizing use of IXCMIAPU utility

Each of the following IXCMIAPU functions should be protected in the FACILITY RACF class:

- ARM - Automatic restart management (See Appendix 4-8, “ARM support” on page 95)
- CFRM - Coupling facility resource manager
- LOGR - Logstream management
- SFM - Sysplex failure management

Example 4-8 shows how you can protect these IXCMIAPU functions.

Example 4-8 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.LOGR UACC(NONE) OWNER(SUPMVS)
RDEFINE FACILITY MVSADMIN.XCF.SFM UACC(NONE) OWNER(SUPMVS)
PE MVSADMIN.XCF.LOGR CLASS(FACILITY) ID(SYSPLXAD) ACC(UPDATE)
SETR RACLIST(FACILITY) REFRESH
```

In order to use Automatic Restart Management (ARM), the XCF address space must have adequate authorization in RACF. ARM issues commands from the XCF address space when performing restarts; to allow this to happen, XCFAS(XCF Address Space) must have enough authority to issue commands to restart any failed element. It is recommended that the XCFAS is in the RACF STARTED class or the started procedures table with the trusted attribute. See Figure 4-8 for overvierw of ARM support.

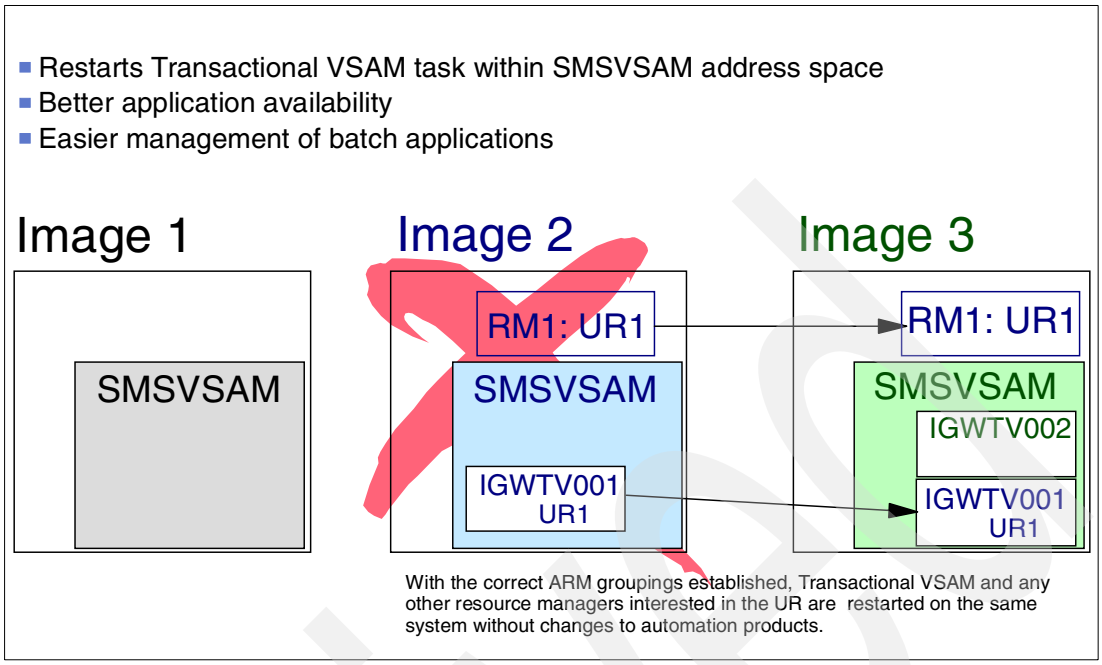


Figure 4-8 ARM support

UPDATE access must be given to users who must maintain the policy; READ access is for those who need only to list the policy.

In addition, for TYPE(LOGR), you must be authorized to update the Logstream resource in LOGSTRM class.

Example 4-9 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)
```

The security administrator can control the use of automatic restart management by unauthorized applications through the use of RACF or another security product. Unauthorized applications can use only the element names that are registered as ELEMBIND=CURJOB elements. Ensure that you are authorized to issue the IXCARM macro for such unauthorized applications.

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

1. Define 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)
```

4.2.3 Authorizations for system logger applications

It is recommended that you control which applications have access to the system logger resources, such as log streams or coupling facility structures associated with log streams.

For the application, define UPDATE access to SAF resource RESOURCE(log_stream_name) CLASS(LOGSTRM).

4.3 Sysplex Timer®

Sysplex Timer links are the links used to provide the clock synchronization between the mainframes in a Parallel Sysplex. There are two types of links used. The first is the link between each mainframe and the Sysplex Timer, known as the ETR (external throughput rate) links. The second is the link between redundant Sysplex Timers, referred to as the CLO (control link oscillator) links. In a high-availability GDPS environment, redundant Sysplex Timers are connected to each mainframe over ETR links, while the timers are connected to each other over the CLO links. This protocol operates at 16 Mbps.

Figure 4-9 shows an overview of Parallel Sysplex clock functions.

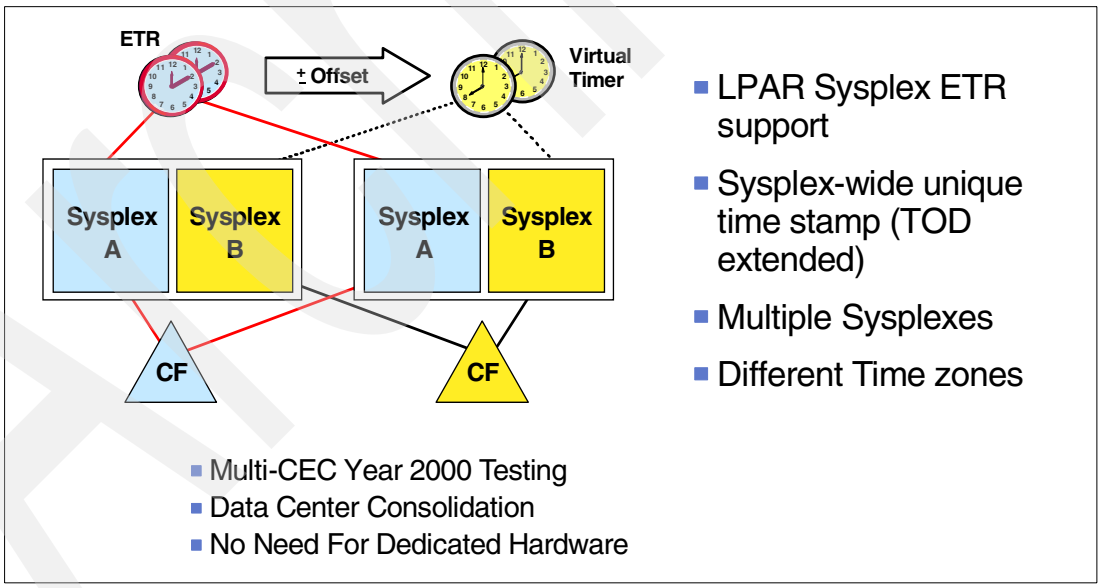


Figure 4-9 Enhanced Parallel Sysplex clock functions

The ETRMODE parameter in your CLOCKxx indicates whether you are using a Sysplex Timer to synchronize time; “yes” means the Sysplex Timer will be used. The time of day for each LPAR is the Sysplex Timer, time adjusted by the logical offset value. In a multisystem environment, XCF will allow only images with the same logical offset value to participate in the same Sysplex with one another. At IPL time, XCF detects any inconsistent logical offset values and prevents the IPLing system from joining the Sysplex.

4.4 Sysplex operator commands protection

To achieve a good security level, operator commands related to the sysplex should be protected. In a sysplex, because an operator on one system can enter commands that affect the processing on another system, your security measures become more complicated and you need to plan accordingly.

4.4.1 Console security

Console security means controlling which commands operators can enter on their consoles to monitor and control MVS.

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 userid 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:

- AUTH keyword on the CONSOLE statement of CONSOLxx
- LOGON keyword of the DEFAULT statement and RACF commands and profiles

Command authorization without RACF checking

When RACF is not active or when the OPERCMDS RACF class 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), and consoles with IO authority can issue in the IO command group.

Command authorization with RACF checking

RACF command authorization checking requires RACF 1.9 or later to be installed and active, and the OPERCMDS class to 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.

With RACF 1.9 or later installed, the installation can require or allow the operators to log on to any MCS console. The operator's RACF profile and group authority determines what commands can be issued from the console when the RACF OPERCMDS class is active. Operators can be required to log on and log off from MCS-managed consoles by specifying LOGON (REQUIRED) or LOGON (AUTO) options on the DEFAULT statement in the CONSOLxx parmlib member. When the RACF Console class 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 CONSOLE-NAME.

LOGON (REQUIRED)

LOGON (REQUIRED) specifies that an operator must log on to an MCS console before issuing commands from that console. If an operator is not logged on to the console, the system rejects commands issued from that console.

LOGON (AUTO)

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.

LOGON (OPTIONAL)

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.

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 dataspace 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's RACF profile
- The OPERPARM data area MCSOP
- System defaults

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 new TSO/E user TAPE1:

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

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

Example 4-10 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)
```

4.4.2 Command resource names

Command resource names, defined in your RACF database, are used by the CMDAUTH service to verify whether an operator is authorized to issue a given command. The command resource names (sometimes called entity names) enable you to logically group operator commands and 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 4-11 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 4-11 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
```

OPERCMDS class must first be activated and raclisted, and a profile may be defined for each of these commands. Table 4-1 lists all the sysplex-related MVS commands that should be protected.

Table 4-1 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
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 Commands	READ	MVS.UNKNOWN

5

TCP/IP security in a sysplex configuration

Sysplex Distributor was a new function for IBM Communications Server for OS/390 V2R10 IP. It takes the XCF dynamics and dynamic VIPA support to a whole new level in terms of availability and workload balancing in a sysplex. Workload can be distributed to multiple server instances within the sysplex without requiring changes to clients or networking hardware and without delays in connection setup. IBM Communications Server for OS/390 V2R10 enables you to implement a dynamic VIPA as a single network-visible IP address for a set of hosts that belong to the same sysplex cluster. Any client located anywhere in the IP network is able to see the sysplex cluster as one IP address regardless of the number of hosts that it includes.

With Sysplex Distributor, clients receive the benefits of workload distribution provided by both Workload Manager (WLM) and Quality of Service (QoS) Policy Agent. In addition, Sysplex Distributor ensures high availability of the IP applications running on the sysplex cluster, no matter whether one physical network interface fails or an entire IP stack or OS/390 is lost.

5.1 TCP/IP in Parallel Sysplex

5.1.1 Supported connectivity protocols and devices

Data Link Controls (DLCs) can be classified into two categories: TCP-exclusive DLCs and shared DLCs.

TCP-exclusive DLCs

TCP-exclusive DLCs are those only available for the Communications Server for OS/390 IP stack and cannot be shared between multiple instances of Communications Server for OS/390 IP. The TCP-exclusive DLCs supported by Communications Server for OS/390Communications Server for OS/390 IP include the following channel protocols:

- Channel Data Link Control (CDLC) - This protocol supports a native IP connection between Communications Server for OS/390 IP and an IP router coded within a 374x running Network Control Program (NCP) or within a 9x0 channel-attached router.
- Common Link Access to Workstation (CLAW) - This protocol is used to connect the Communications Server for OS/390 IP to a 3172 running ICCP, an RS/6000®, and Cisco routers supporting this interface.
- Channel-to-Channel (CTC) - This protocol is supported between two Communications Server for OS/390 IP systems and uses one read/write channel pair. Both parallel and ESCON channels are supported.
- Hyperchannel - This protocol is used to connect via the NSC A220 Hyperchannel Adapter and its descendants.
- LAN Channel Station (LCS) - This protocol is used by OSA, the 3172 running ICP, the 2216, and the 3746-9x0 MAE.

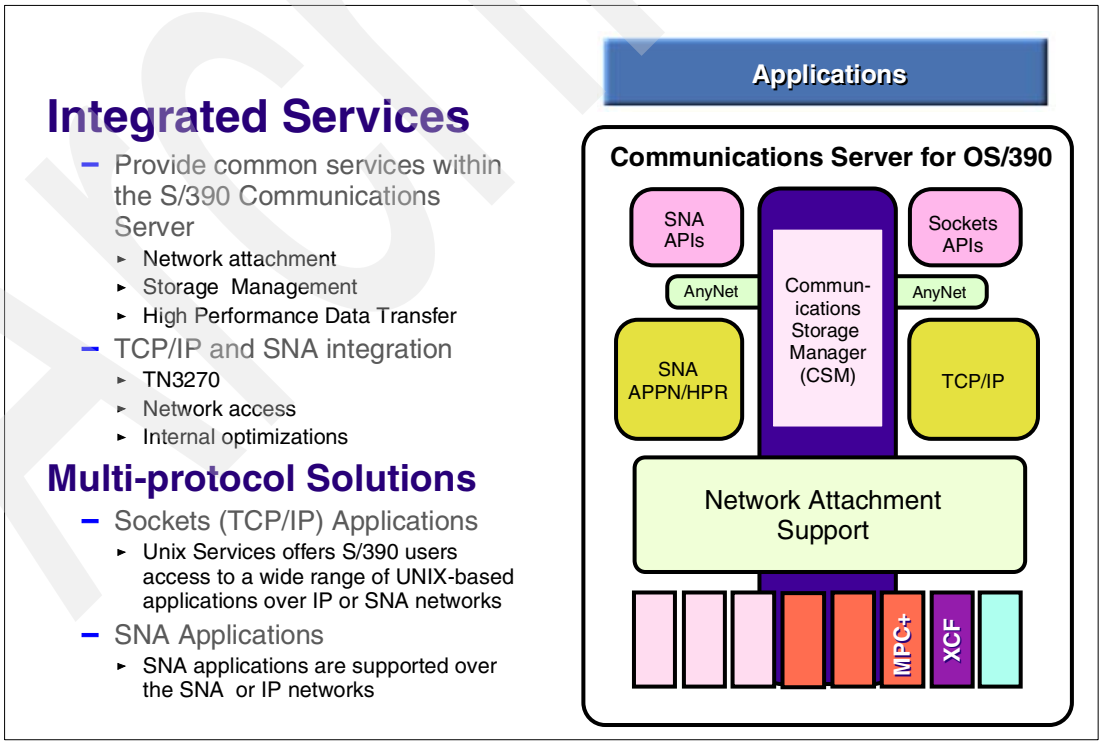


Figure 5-1 Communications Server for OS/390

In addition, one more TCP-exclusive DLC protocol exists, although it does not make use of S/390 channels. Not to be confused with PTP Samehost, the SAMEHOST DLC enables communication between Communications Server for OS/390 IP and other servers running on the same MVS image. In the past, this communication was provided by IUCV. Currently, three such servers exploit the SAMEHOST DLC:

- SNALINK LU0 - This server provides connectivity through SNA networks using LU0 traffic. It acts as an application to OS/390 VTAM.
- SNALINK LU6.2 - This server provides connectivity through SNA networks using LU6.2 traffic. It also acts as an application to OS/390 VTAM.
- X.25 - This server provides connectivity to X.25 networks by using the NCP packet switching interface (NPSI).

Shared DLCs

Shared DLCs are those that can be simultaneously used by multiple instances of multiple protocol stacks. For example, a shared DLC may be used by one or more instances of Communications Server for OS/390 IP and one or more instances of OS/390 VTAM. These shared DLCs include:

- Multipath Channel+ (MPC+) - MPC+ is an enhanced version of the Multipath Channel (MPC) protocol. It allows for the efficient use of multiple read and write channels. High Performance Data Transfer (HPDT) uses MPC+ together with Communication Storage Manager (CSM) to decrease the number of data copies required to transmit data. This type of connection can be used in two ways:
  - The first of these is MPCPTP, in which Communications Server for OS/390 IP is connected to a peer IP stack in a point-to-point fashion. In this way, Communications Server for OS/390 IP can be connected to each of the following:
    - Another Communications Server for OS/390 IP stack
    - 2216
    - RS/6000
    - 3746-9x0 MAE
    - Cisco routers via the Cisco Channel Interface Processor (CIP) or the Cisco Channel Port Adapter (CPA)
  - The second way to use MPC+ is to connect to an Open Systems Adapter (OSA). In this configuration, OSA acts as an extension of the Communications Server for OS/390 IP stack and not as a peer IP stack as in MPCPTP. The following are supported in this manner:
    - OSA-2 native ATM (RFC1577)
    - OSA-2 Fast Ethernet and FDDI (MPCOSA)
    - OSA-Express QDIO uses MPC+ for the exchange of control signals between Communications Server for OS/390 IP and the OSA-Express
- MPCIPA (QDIO) - The OSA-Express provides a new mechanism for communication called Queued Direct I/O (QDIO). Although it uses the MPC+ protocol for its control signals, the QDIO interface is quite different from channel protocols. It uses Direct Memory Access (DMA) to avoid the overhead associated with channel programs. Originally, the OSA-Express and Communications Server for OS/390 IP only supported the Gigabit Ethernet attachment. Communications Server for OS/390 V2R10 IP provides QDIO support for Fast Ethernet and ATM LAN emulation as well. A partnership between Communications Server for OS/390 IP and the OSA-Express Adapter provides offload of compute-intensive functions from the S/390 to the adapter. This interface is called IP Assist (IPA). Offloading reduces S/390 cycles required for network interfaces and provides

an overall improvement in the OS/390 OSA-Express environment compared to existing OSA-2 interfaces.

- XCF - The XCF DLC allows communication between multiple Communications Server for OS/390 IP stacks within a Parallel Sysplex via the Cross-System Coupling Facility (XCF). The XCF DLC can be defined, as with traditional DLCs, but it also supports XCF Dynamics, in which the XCF links are brought up automatically.
- PTP Samehost - Sometimes referred to as IUTSAMEH, this connection type is used to connect two or more Communications Server for OS/390 IP stacks running on the same MVS image. In addition, it can be used to connect these Communications Server for OS/390 IP stacks to OS/390 VTAM for the use of Enterprise Extender. See Figure 5-2 for and overview of TCP/IP and the Parallel Sysplex.

CS for OS/390 Coupled Value
- MPC+
- System Symbolics Support
  - TCP/IP System cloning
    - New in OS/390 R7
- TN3270 support
  - Cloning for TN3270
  - TN3270 visibility
  - Tivoli Application Monitor
    - New in OS/390 R8

TCP/IP XCF Support
- Horizontal growth for data center
  - XCF dynamics
    - New in OS/390 R7
- Helping applications exploit parallel Sysplex
  - Sysplex Sockets
    - New in OS/390 R7

Domain Name Service with WLM
- Targeted towards "long duration" applications like TN3270 and FTP

Interactive Network Dispatcher
- Targeted towards "short duration" applications like web traffic
  - Available on IBM's networking products (e.g. 2216)

Virtual IP Addressing (VIPA)
- Dynamic recovery from S/390 attachment failures
- Dynamic recovery from S/390 TCP/IP application failures
  - VIPA Takeover
    - New for OS/390 R8
  - Sysplex Distributor
    - New for OS/390 R10

SNA and TCP/IP Integration
- Enterprise Extender
  - Support on IBM Routers and Servers
  - Support on CS/390
    - New for OS/390 R6
- Multi-Node Persistent Sessions (MNPS)
  - MNPS With Enterprise Extender
    - New for OS/390 R6

Figure 5-2 TCP/IP and the Parallel Sysplex

Applications executing in a sysplex (or other cluster) can often take advantage of executing in the same sysplex—if the application instance knows the other end is in the same sysplex or the same MVS image. Possible benefits include avoiding encryption between MVS images when the sysplex is considered physically secure and the connection is via XCF or ESCON channels (CTC or MPCPTP) or use of shared memory to pass information between applications in the same MVS image.

IP applications know their partner only by IP address, which could be anywhere, even if the address is in the same subnet. Sysplex Sockets exploits MVS XCF Messaging to allow the sysplex TCP stacks to share their HOME lists of IP addresses with each other, without the need to communicate via IP or other networking protocol. In other words, a TCP stack can communicate with its partner stacks in the sysplex even before the first IP device and link are defined and activated. This is illustrated in Figure 5-3.

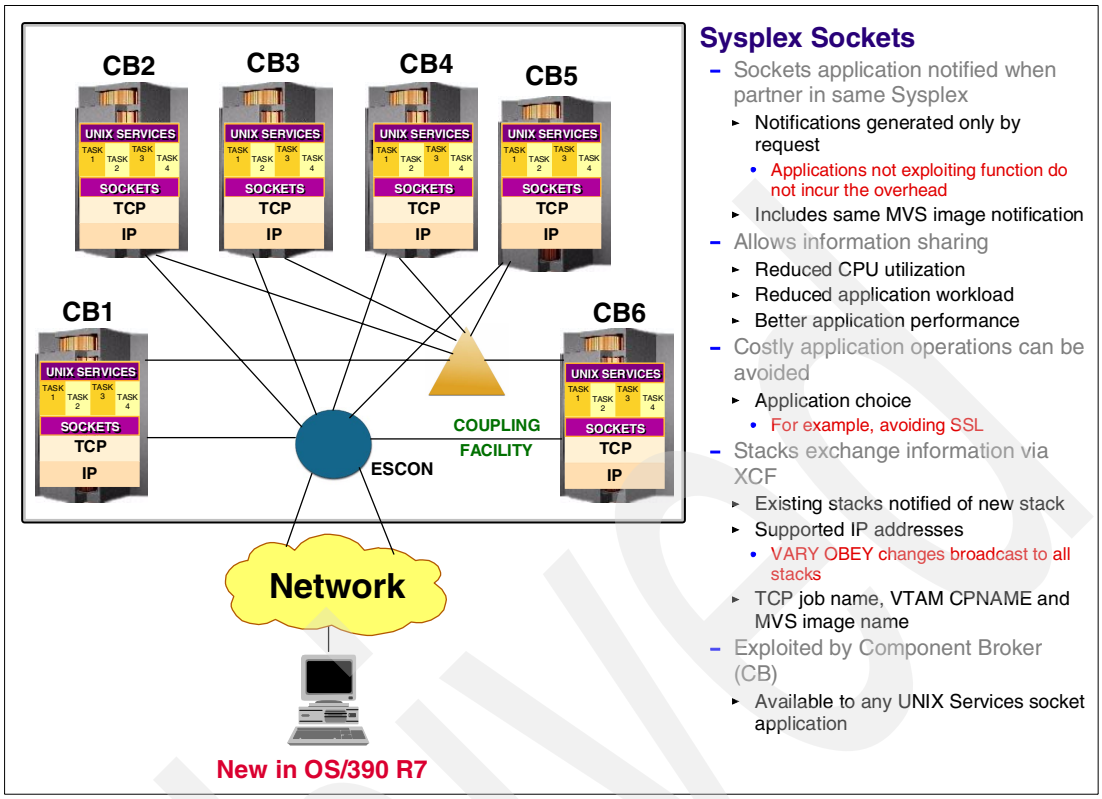


Figure 5-3 Application exploitation of sysplex

Existing applications' TCP connections are not affected, either in API or in performance, by this new function. A new getsockopt() option (CLUSTERCONNTYPE) was added. When specified by an application wishing to exploit sysplex sockets, the stack searches its table of known IP addresses in the sysplex and determines:

– Whether the IP address is known to be supported by one of the Sysplex TCP stacks and, if so:

- Whether the IP address is supported by the same stack or another stack in the same MVS image
- Whether the connection is Same Host (via IUTSAMEH device in VTAM) or via XCF or MPCPTP or CTC with a single-hop route to the other stack, in which case an indication of "Internal" is also returned.

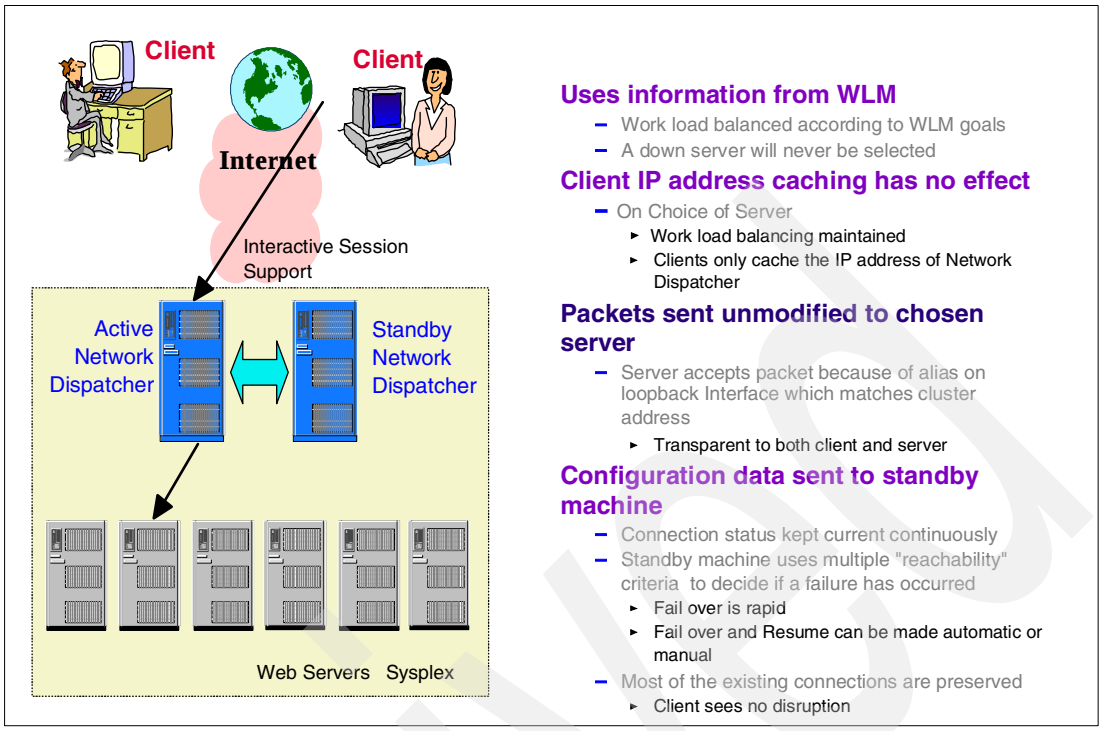


Figure 5-4 Interactive Network Dispatcher

The Interactive Network Dispatcher (illustrated in Figure 5-4) is provided to address the need to balance work requests sent to a single IP address across a number of servers on different MVS images (or at least TCP stacks) in the sysplex. In other words, IND provides a single-system image of the sysplex to the client population. The IND gets information from sysplex Work Load Manager instances on relative loads periodically, and uses this information to decide, for any particular request, to which server to send it. Since IND is doing balancing to multiple servers based on a single IP address, the fact that the client caches the IP address would defeat workload balancing via DNS/WLM.

IND also keeps a partner informed about the current status of existing connections. Should the active IND fail, the backup IND will quickly learn of the failure, and will have nearly all the information (except possibly for the very most recent connections) that the original IND had, thus allowing preservation of the existing connections with no visibility of the failure to the client or the server.

IND currently is available on 2216 and on AIX.

XCF Dynamics (OS/390 2.7)

This is built upon the capability of VTAM to discover other VTAMs in the sysplex, and to establish automatically XCF pipes by constructing dynamic Transport Resource List Entries (TRLEs) named after the discovered VTAM CPName. VTAM also supplies the TRLE called IUTSAMEH for communication (actually storage-to-storage transfer) with stacks in the same OS/390 image.

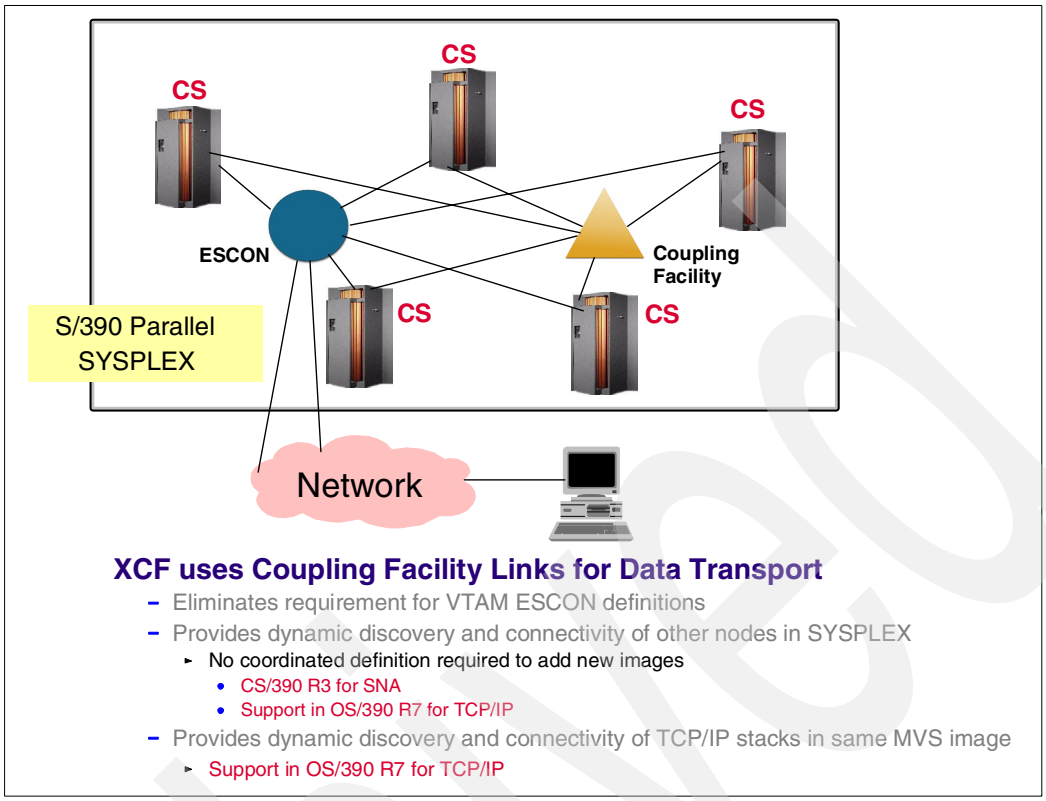


Figure 5-5 Sysplex Connectivity via the XCF facility

XCF Dynamics allow dynamic definition of TCP XCF and SameHost connectivity that builds on Sysplex Sockets (stacks communicate with each other) to exchange VTAM CPNames along with other information such as SYSCLONE value. It can:

- Define XCF dynamically when new TCPs are discovered on different VTAM:
  - DEVICE definition is always `DEVICE` *`newCPName`* `MPCPTP` (*newCPName* is the other VTAM)
  - LINK definition is always `LINK EZAXCF&SYSCLONE newCPName MPCPTP` (*SYSCLONE* values are unique within the sysplex)
  - Use same IP address for all dynamic definitions
  - START newCPName
- Define SameHost dynamically when a second stack is discovered on the same MVS image:
  - DEVICE definition is always `DEVICE IUTSAMEH MPCPTP`
  - LINK definition is always `LINK EZASAMEMVS IUTSAMEH MPCPTP`
  - Same IP address as XCF
  - START IUTSAMEH
- Handle routing automatically:
  - Subnet mask used to construct routing table entry
  - Routing daemon picks up new routing table entry and propagates it

The XCF and SameHost definition on each stack is achieved using a single configuration statement:

```
IPCONFIG DYNAMICXCF IPAdress subnetMask costMetric
```

IPAddress is used for all XCF and SameHost definitions for the stack; subnetMask and costMetric are used for routing.

It indicates that XCF dynamic support is enabled, Internet_addr is the IP address used with the dynamically generated HOME statement entries for XCF,IQDIO(Hypersockets), or same hostlinks. The subnet_mask and cost_metric is used on a BSDROUTINGPARMS entry for OROUTED. If dynamic routing is being provided by OMPROUTE, the subnet_mask and cost_metric values must be configured to OMPROUTE using the interface, OSPF_Interface, or RIP_Interface configuration statements in the OMPROUTE configuration file.

Note: Define the OMPROUTE statements before coding DYNAMICXCF in the profile. If you do not do this, the wrong subnet mask is used and an error message is issued. The DYNAMICXCF parameter is confirmed by the message:

```
DYNAMIC XCF DEFINITIONS ARE ENABLED
```

When encountered in initial profile processing or VARY OBEY, this causes dynamic XCF and SameHost definitions to be generated for all new VTAMs and same-host stacks discovered now and in the future, until VARY OBEY with a profile containing IPCONFIG NODYNAMICXCF. This halts all future dynamic XCF and SameHost definition activity. Existing connectivity is unaffected.

Note: The VTAM start option XCFINIT must be set to **yes** (default value) in order for TCP XCF Dynamics to work.

NETSTAT CONFIG displays `DynamicXCF : 00001` if dynamic XCF is specified in the IPCONFIG statement of the subject stack.

Security considerations include the following:

- All IP links definitions are automatically built when a new member joins the sysplex, with the result of providing information that might not be desirable to the already existing members.
- A running routing daemon will publicize the information.
- IPCONFIG NODYNAMICXCF prevents dynamic building of the links definitions, but stacks with NODYNAMICXCF will not be part of the Sysplex Distributor configuration.

5.2 VIPA and Dynamic VIPA

The concept of the *virtual IP address* (VIPA) was introduced by IBM to remove the dependencies of other hosts on particular network attachments to CS for OS/390 IP. Prior to VIPA, other hosts were bound to one of the home IP addresses and, therefore, to a particular network interface. If the *physical* network interface failed, the home IP address became unreachable and all the connections already established with this IP address also failed. VIPA provides a *virtual* network interface with a virtual IP address that other TCP/IP hosts can use to select an OS/390 IP stack without choosing a specific network interface on that stack. If a specific physical network interface fails, the VIPA address remains reachable by other physical network interfaces. Hosts that connect to OS/390 IP applications can send data to a

VIPA address via whatever path is selected by the dynamic routing protocol (such as RIP or OSPF).

A VIPA is configured the same as a normal IP address for a physical adapter, except that it is not associated with any particular interface. VIPA uses a virtual device and a virtual IP address. The virtual IP address is added to the home address list. The virtual device defined for the VIPA using DEVICE, LINK and HOME statements is always active and never fails. Moreover, the OS/390 IP stack advertises routes to the VIPA address as if it were one hop away and has reachability to it. Figure 5-6 presents an overview of how VIPA works.

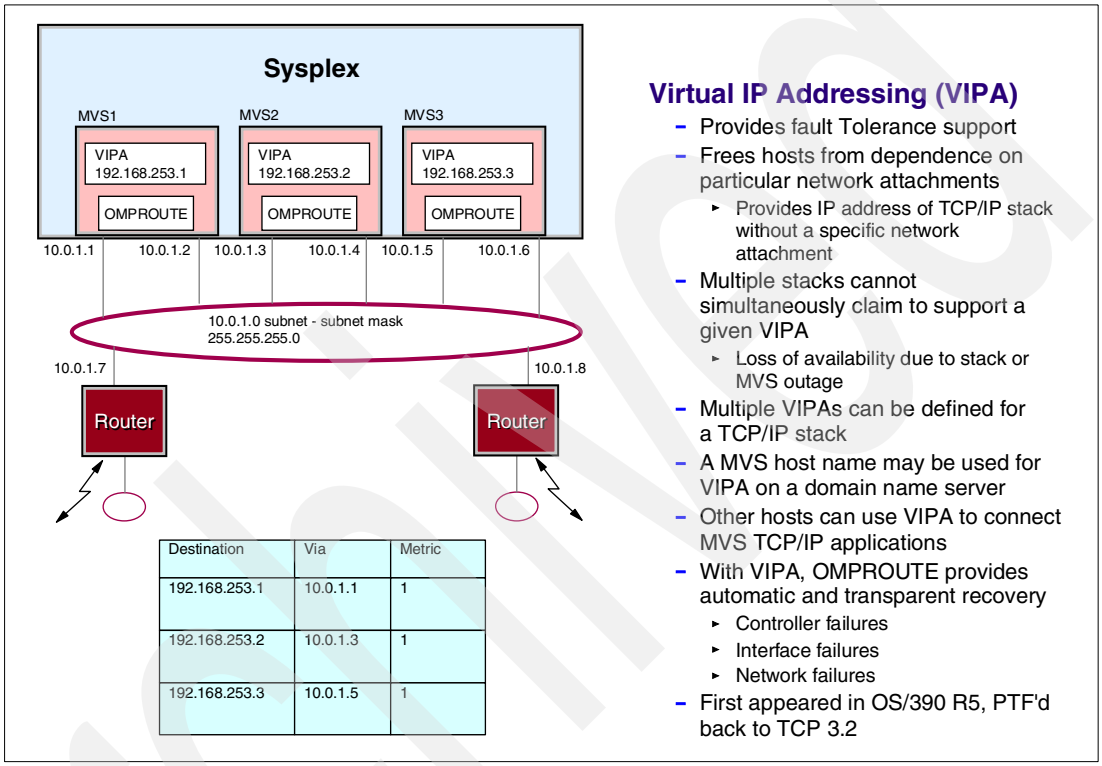


Figure 5-6 VIPA overview

To an attached router, the IP stack in OS/390 simply looks like another router. When the IP stack receives a packet destined for the VIPA, the inbound IP function of the stack notes that the IP address of the packet is in the stack's home list and forwards the packet up the stack. Assuming that the IP stack has more than one network interface, if a particular network interface fails, the downstream router will simply route VIPA-targeted packets to the stack via an alternate route. In other words, the destination IP stack on OS/390 is still reachable and it looks like another intermediate node. The VIPA may thus be thought of as an address of the stack and not of any particular network interface associated with the stack.

While VIPA certainly removes the dependency on any particular network interface as a single point of failure, the connectivity of a server can still be lost when a single stack or an OS/390 image fails. When this occurs, you could manually move a VIPA to another stack using the **obey** command (or semi-automatically using any system management automation of the manual process). This type of VIPA is not considered desirable since the process is inherently manual. Instead, automatic VIPA movement and activation mechanisms were added.

Dynamic VIPA was introduced by SecureWay Communications Server for OS/390 V2R8 IP to enable the dynamic activation of a VIPA as well as the automatic movement of a VIPA to

another surviving OS/390 image after an OS/390 stack failure. There are two forms of Dynamic VIPA, both of which can be used for takeover functionality:

- Automatic VIPA takeover allows a VIPA address to move automatically to a stack (called a backup stack) where an existing suitable application instance is already active, and allows the application to serve the client formerly going to the failed stack.
- Dynamic VIPA activation for an application server allows an application to create and activate VIPA so that the VIPA moves when the application moves.

Non-disruptive, immediate, automatic VIPA *takeback* was introduced by IBM Communications Server for OS/390 V2R10 to move the VIPA back to where it originally belongs once the failed stack has been restored. This takeback is non-disruptive to existing connections with the backup stack, and the takeback is not delayed until all connections with the backup stack have terminated (as was the case with CS for OS/390 V2R8 IP). New connections will be handled by the new (original) primary owner, thereby allowing the workload to move back to the original stack.

5.3 Sysplex Distributor

Sysplex Distributor was designed to address the requirement of having one single network-visible IP address for the sysplex cluster and to let the clients in the network receive the benefits of workload distribution and high availability within the sysplex cluster. With Sysplex Distributor, client connections seem to be connected to a single IP host even if the connections are established with different servers in the same sysplex cluster. See Figure 5-7 for an overview diagram of the Sysplex Distributor.

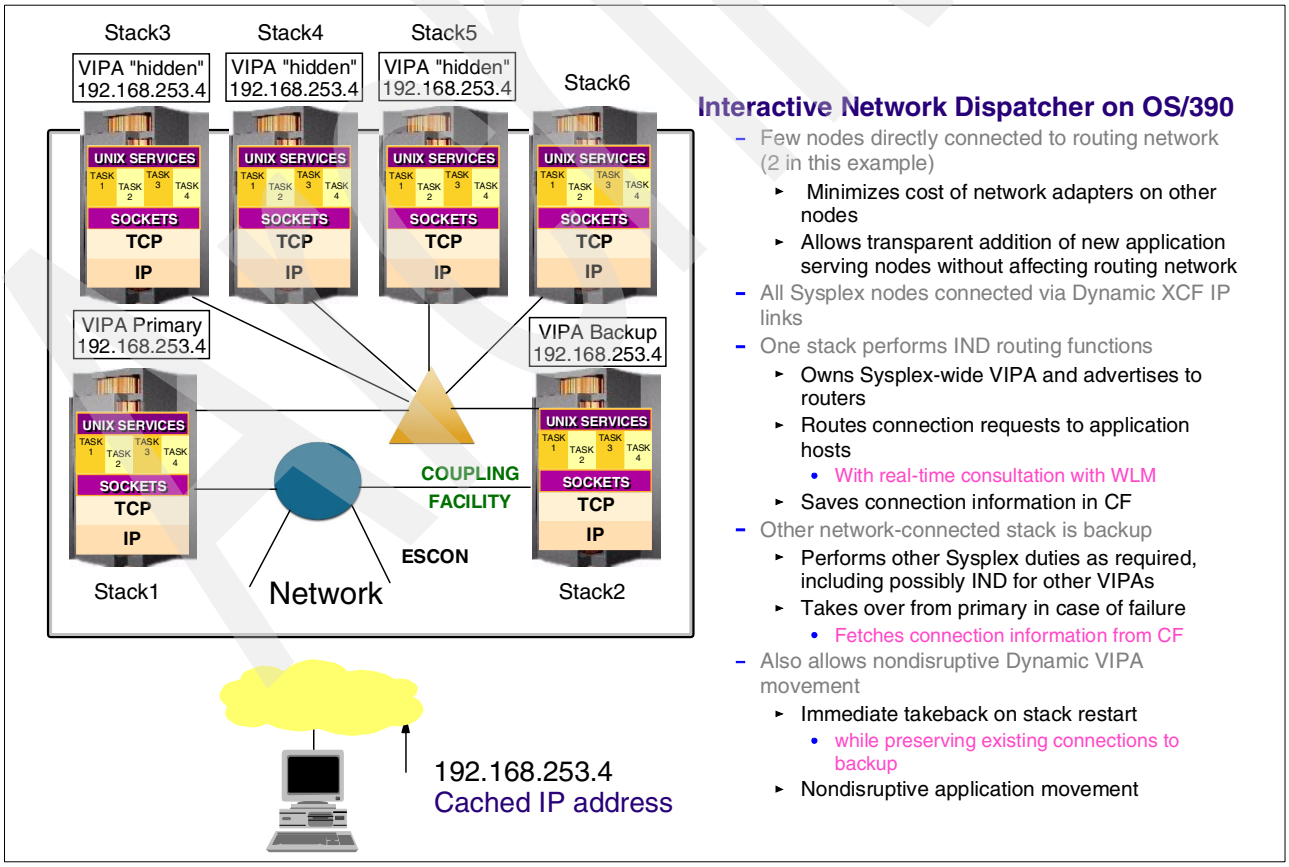


Figure 5-7 Sysplex Distributor overview

Because the Sysplex Distributor function resides on a system in the sysplex itself, it has the ability to factor “real-time” information concerning the multiple server instances, including server status as well as QoS and policy information provided by CS for OS/390 IP’s Service Policy Agent. By combining these real-time factors with the information from WLM, the Sysplex Distributor has the unique ability to ensure that the best destination server instance is chosen for a particular client connection. The Sysplex Distributor has more benefits than other load-balancing implementations, such as the Network Dispatcher or DNS/WLM, without their limitations.

The benefits of Sysplex Distributor include that it:

- Removes the configuration limitations of Network Dispatcher. Target servers can use XCF links between the distributing stack and target servers, as opposed to LAN connections such as an OSA.
- Removes the dependency of specific hardware in a WAN and provides a total CS for OS/390 IP solution for workload distribution.
- Provides real-time workload balancing for TCP/IP applications, even if clients cache the IP address of the server (a common problem for DNS/WLM).
- Enhances VIPA takeover and takeback support. Allows for non-disruptive takeback of VIPA original owner to get workload where it belongs. Distributing function can be backed up and taken over. Enhances Dynamic VIPA support of non-disruptive application server instance movement.

In summary, Sysplex Distributor provides:

1. A single, network-visible IP address for a sysplex cluster service. One IP address can be assigned to the entire sysplex cluster (usually for each service provided, such as Telnet). Sysplex Distributor will query the Policy Agent to find if there exists any policy defined for routing the incoming connection requests. WLM and QoS policy can be specified for workload balancing in real-time on every new connection request.
2. It raises the limit of 64 DVIPAs on a stack to 256.
3. Backup capability is enhanced. In case of failure of the distributing IP stack, the connections distributed to other IP stacks in the sysplex will not be disrupted.
4. Dynamic VIPA takeback without any disruption in the connections already established.
5. New commands are added to display both the connection routing table and destination port table information, showing details of configuration and current connection distribution.

The specific profile statements that should be used to configure Sysplex Distributor are discussed in 5.6, “Sysplex Distributor implementation” on page 120.

5.3.1 Sysplex Distributor functionality

Let us consider the scenario depicted in Figure 5-8 on page 115. This includes four CS for OS/390 V2R10 IP stacks running in the same sysplex cluster in GOAL mode (WLM goal mode). All of them have SYSPLEXROUTING, DATAGRAMFWD, and DYNAMICXCF configured. Let‘s assume that:

- H1 is configured as the distributing IP stack with V1 as the Dynamic VIPA (DVIPA) assigned to the sysplex cluster.
- H2 is configured as backup for V1.
- H3 and H4 are configured as secondary backups for V1.

- Let us suppose that APPL1 is running in all the hosts that are members of the same sysplex cluster. Note that the application could also be running in two or three of the hosts or in all of them at the same time.

With this in mind, we describe how Sysplex Distributor works:

1. When IP stack H1 is activated, the definitions for the local XCF1 link are created dynamically due to DYNAMICXCF being coded in the H1 profile. Through this new link, H1 recognizes the other IP stacks that belong to the same sysplex cluster and their XCF associated links: XCF2, XCF3, and XCF4.
2. The DVIPA assigned to the sysplex cluster and the application ports that this DVIPA serves are read from the VIPADISTRIBUTE statement in the profile data set. An entry in the home list is added with the distributed IP address in all the IP stacks. The home list entry on the target stacks is actually done with a message that H1 sends to all the stacks read from the VIPADISTRIBUTE statement. Only one stack advertises the DVIPA through the RIP or OSPF routing protocol. In this case it is the one that resides in H1, the host in charge of load distribution.
3. H1 monitors whether there is at least one application (APPL1 in Figure 5-8) with a listening socket for the designated port and DVIPA. Actually H2, H3, and H4 will send a message to H1 when a server (in our case APPL1) is bound to either INADDR_ANY or specifically to the DVIPA (and, of course, the designated port). With that information, H1 builds a table with the name of the application and the IP stacks that could serve any connection request for it. The table matches the application server listening port with the target XCF IP address.
4. When a client in the network requests a service from APPL1, the DNS resolves the IP address for the application with the DVIPA address. This DNS could be any DNS in the IP network and does not need to register with WLM.
5. As soon as H1 receives the connection request (TCP segment with the SYN flag), it queries WLM and/or QoS to select the best target stack for APPL1 and forwards the SYN segment to the chosen target stack. In our example, it is APPL1 in H4 that best fits the request.
6. One entry is created in the connection routing table (CRT) in H1 for this new connection with XCF4 as the target IP address. H4 also adds the connection to its connection routing table.

   Note: If a program binds to DVIPA on H4 and initiates a connection, H4 needs to send a message to H1, so H1 can update its connection routing table accordingly. As an example, this is used when the FTP server on H4 would initiate a data connection (port 20) to a client.

7. The H1 IP stack will forward subsequent incoming data for this connection to the correct target stack.
8. When the H4 IP stack decides that the connection no longer exists, it informs the H1 IP stack with a message so H1 can remove the connection from its connection routing table.

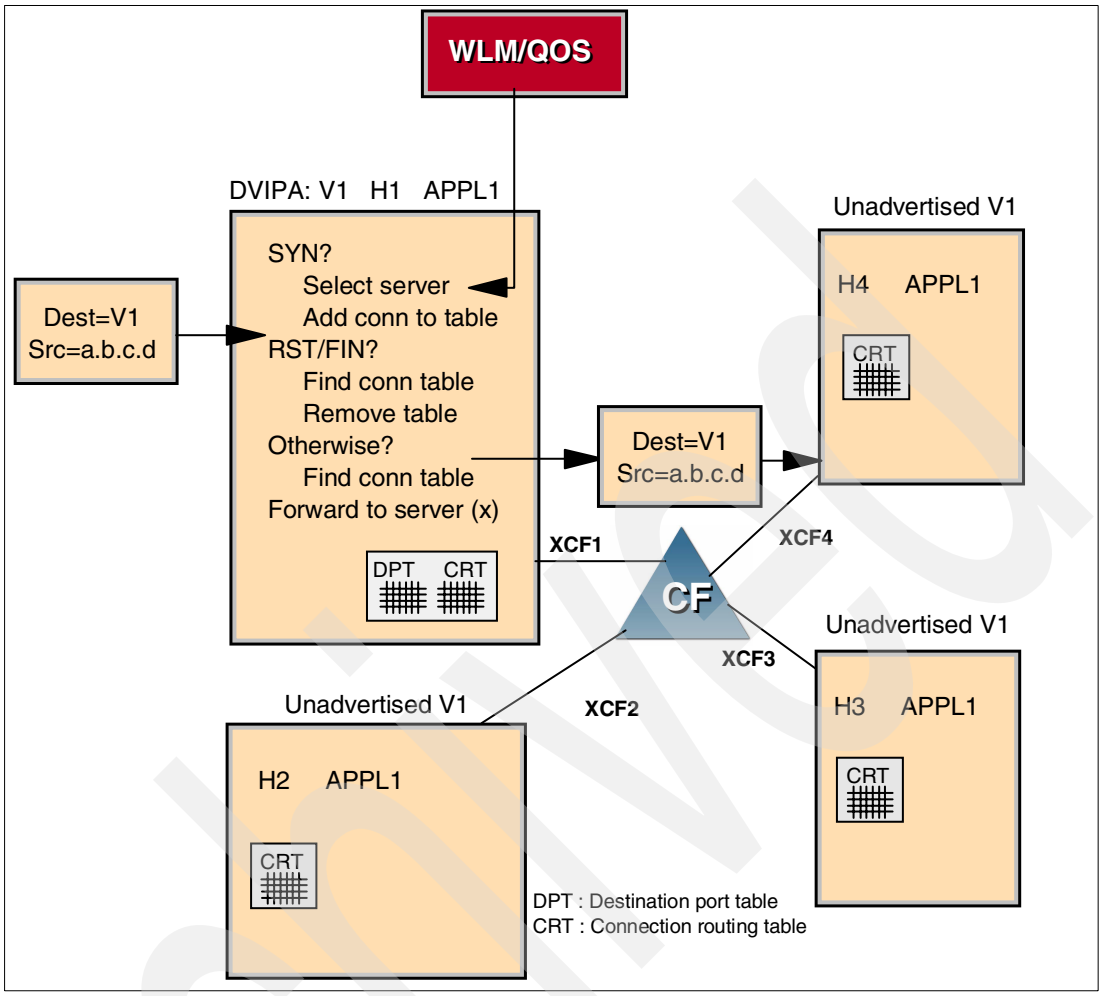


Figure 5-8 Sysplex Distributor functionality

5.3.2 Backup capability

Let us say that the scenario depicted has been running for some time without problems. The new APPL1 connections have been distributed according to WLM and QoS to H1, H2, H3, and H4. Suppose that a considerable number of connections are currently established between several APPL1 server images and clients in the IP network. What would happen if we had a major failure in our distributing IP stack, H1?

Automatic Dynamic VIPA takeover was introduced in SecureWay Communications Server for OS/390 V2R8 IP. This function allows a VIPA address to automatically move from one IP stack where it was defined to another one in the event of the failure of the first. The VIPA address remains active in the IP network, allowing clients to access the services associated with it.

In IBM Communication Server for OS/390 V2R10, this VIPA takeover functionality has been enhanced to support Sysplex Distributor. Consider the scenario illustrated in Figure 5-9.

H1 is the distributing IP stack and H2 is the primary VIPABACKUP IP stack. When H1 fails (Figure 5-9):

1. All the IP connections terminating at H1 are lost.
2. The Sysplex Distributor connection routing table (CRT) is also lost.

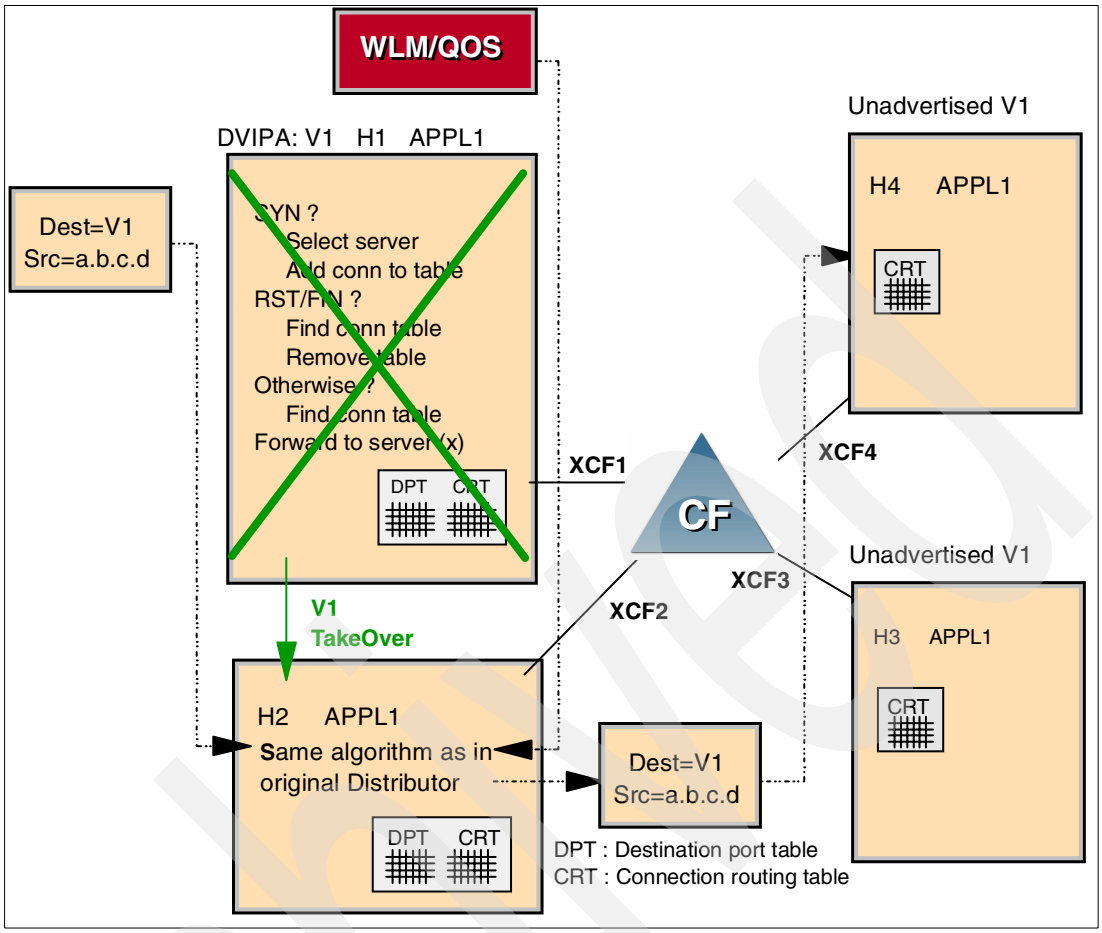


Figure 5-9 Sysplex Distributor and VIPA takeover

3. H2 detects that H1 is down and defines itself as the distributing IP stack for the VIPA.
4. Because H2 saved information about H1, it informs the other target stacks that it knows V1 is distributable.
5. H3 and H4 find out that H2 is the chosen backup for V1 and immediately send connection information regarding V1 to IP stack H2.
6. H2 advertises V1(DVIPA) through the dynamic routing protocol (RIP or OSPF). Retransmitted TCP segments for already existing connections or SYN segments for new connections are hereafter processed by IP stack H2 and routed by H2 to the appropriate target stacks.

Note: Only the IP connections with the failing IP stack were lost. All other connections remain allocated and function properly.

5.3.3 Recovery

Once the H1 IP stack is activated again, the process of taking back V1 to H1 is started. This process is non-disruptive for the IP connections already established with V1 regardless of which host is the owner at that time (in our example H2). In our example, connection information is maintained by H2. When H1 is re-activated, H2 sends its connection information to H1. This gives H1 the information it needs to once again distribute packets for existing connections to the correct stacks in the sysplex.

Connections with the backup host are not broken when the V1 address is taken back to H1, and takeback is not delayed until all connections with the backup host have terminated (Figure 5-10).

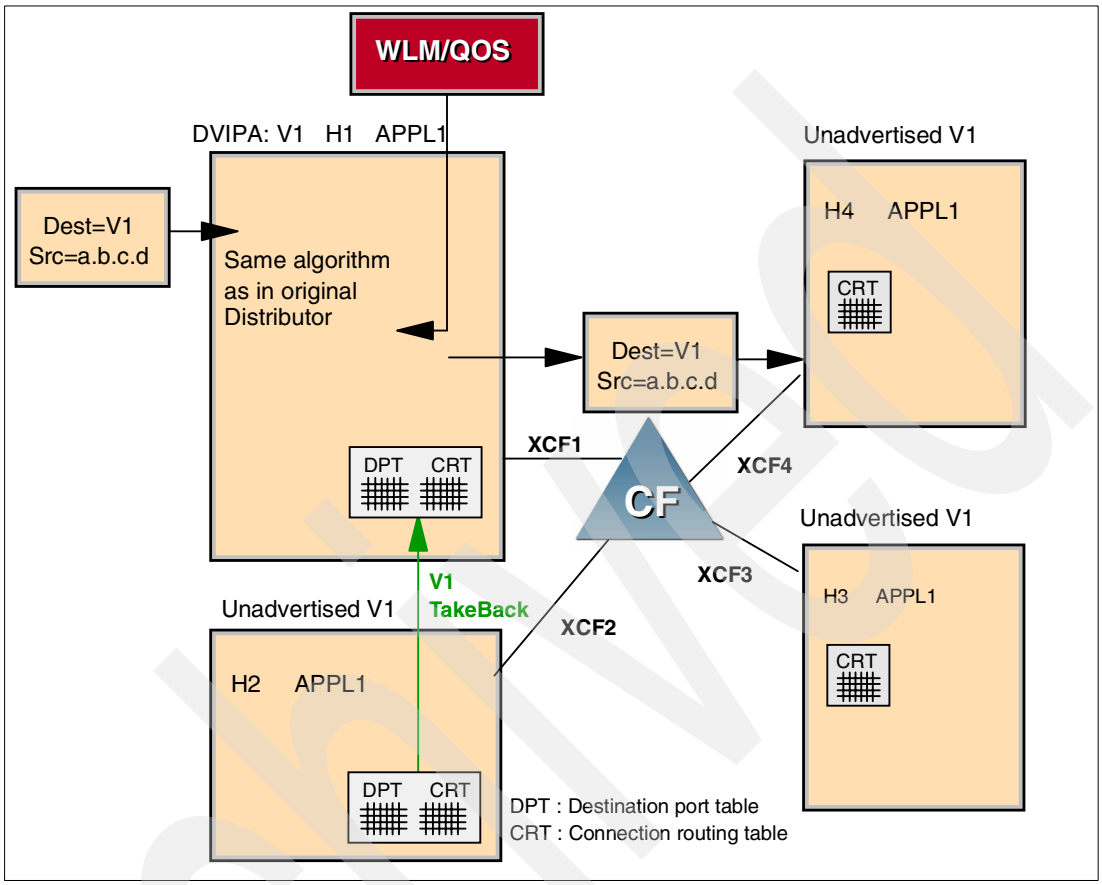


Figure 5-10 Sysplex Distributor and VIPA takeback

5.4 How dynamic routing works with the Sysplex Distributor

Routing IP packets for the Sysplex Distributor can be divided into two cases: routing inside the sysplex cluster and routing through the IP network.

Routing inside the sysplex cluster is accomplished by the distributing host. All incoming traffic (new connection requests and connection data) arrives first to the distributing stack. It forwards the traffic to the target applications, wherever they are in the sysplex cluster, through the XCF links. Here the routing process is done without considering any IP routing table. The WLM and QoS weights are the factors considered in target selection for new requests and the CRT is the consulted data structure for connection data. On the other hand, the outgoing traffic generated by the applications is routed considering the destination IP address and the routing table in each stack.

Routing outside the sysplex through the IP network is done by the downstream routers. Those routers learn about the DVIPA assigned to the sysplex dynamically using OSPF or RIP routing protocols. As a result, it is necessary to implement either one of these routing protocols in all the IP stacks of the sysplex cluster.

The distributing VIPA address is dynamically added to the home list of each IP stack participating in the Sysplex Distributor, but only one IP stack advertises the sysplex VIPA address to the routers: the one defined as the distributing IP stack. The other stacks do not advertise it and only the backup IP stack will do so if the distributing IP stack fails.

If ORouteD is being used, then the Dynamic VIPA support generates the appropriate BSDROUTINGPARMS statement.

If you are using OMPROUTE, you should consider the following:

- The names of Dynamic VIPA interfaces are assigned dynamically by the stack when they are created. Therefore, the name coded for the OSPF_Interface statement in the Name field will be ignored by OMPROUTE.
- It is recommended that each OMPROUTE server have an OSPF_Interface defined for each Dynamic VIPA address that the IP stack might own or, if the number of DVIPA addresses is large, a wildcard should be used.

It is also possible to define ranges of dynamic VIPA interfaces using the subnet mask and the IP address on the OSPF_Interface statement. The range defined will be all the IP addresses that fall within the subnet defined by the mask and the IP address. Example 5-1 defines a range of Dynamic VIPA addresses from 10.138.165.80 to 10.138.165.95.

Example 5-1 Dynamic VIPA OSPF definition

```
OSPF_Interface
   IP_address = 10.138.165.80
   Name = dummy_name
   Subnet_mask = 255.255.255.240
```

For consistency with the VIPARANGE statement in the TCPIP.PROFILE, any value that may fall within this range can be used with the mask to define a range of dynamic VIPAs.

5.5 Sysplex Distributor and policy

Policies are an administrative means to define controls for a network, in order to achieve the QoS levels promised by a given SLA or to implement security or resource balancing decisions. *Quality of Service Policy* allows classification of IP traffic by application, user group, time of day, and assignment of relative priority. The *Policy Agent* reads policy entries from a flat file called pagent.conf that can be located in any MVS or HFS file, or from an LDAP server, or both. The Sysplex Distributor uses these policies to limit the target stack to route its work to in conjunction with the WLM weights. Note that the WLM has to run in GOAL mode at the target stacks, or the QoS weights will have no effect and the distribution of the work is random.

The following types of policies are supported in IBM Communications Server for OS/390 V2R10 IP:

Integrated Services: Service that provides end-to-end QoS using resource reservations along a network path from sender to receiver. This service is provided by the RSVP Agent.

Differentiated Services: Services that provides aggregate QoS to broad classes of traffic (for example, all FTP traffic).

Sysplex Distribution: Policies that specify to which target stack the Sysplex Distributor may route incoming connection requests.

Traffic Regulation Mgmt: Policies that define the maximum number of connections to a TCP port and control the number from a single host to this port.

The Policy Agent performs two distinct functions to assist the Sysplex Distributor:

- Policies can be defined to control which stack the Sysplex Distributor routes traffic to. The definition of the outbound interface on the PolicyAction statement can limit the stacks to which work is distributed to a subset of those defined on the VIPADISTRIBUTE statement in the TCPIP.PROFILE. Using a policy, the stack to which work is distributed can vary, for example, based on time periods. Another possibility is to limit the number of SD target stacks for inbound traffic from a given subnet (Figure 5-11 on page 120).
- The PolicyPerfMonitorForSDR statement in the pagent.conf file will activate the Policy Agent QoS performance monitor function. When activated, the Policy Agent uses data about packet loss and timeouts that exceed defined thresholds and derives a QoS weight fraction for that target stack. This weight fraction is then used to reduce the WLM weight assigned to the target stacks, so that the Sysplex Distributor stack can use this information to better direct workload to where network traffic is best being handled. This policy is activated on SD target stacks (Figure 5-11 on page 120).

To exclude stale data from target stacks where the Policy Agent has terminated, the Policy Agent sends a “heartbeat” to the SD distributing stack at certain intervals. The SD distributing stack deletes QoS weight fraction data from a target stack when the heartbeat has not been received within a certain amount of time.

At ITSO Raleigh, we configured SD policies in two ways. In one scenario, the Policy Agent extracts the policy information from a static configured HFS file (PAGENT file), and in another case, an LDAP server running in OS/390 provides all policy information in the network.

The sysplex consists of three OS/390 systems: RA03, RA28, and RA39. On each system, there is a TCP/IP stack named TCPIPC, which participates in Sysplex Distributor. The TCPIPC stack on RA03 takes the role of the distributing stack, and the stacks on RA39 and RA28 act as the primary and secondary backup respectively. An FTP server has been configured on each SD stack as an application served by SD.

We defined an SD policy for FTP connections to install into the SD distributing stack, which is TCPIPC on RA03, and we defined an SD performance monitoring policy for the SD target stacks, which are TCPIPC on RA28 and RA39. Figure 5-11 illustrates the system environment at ITSO.

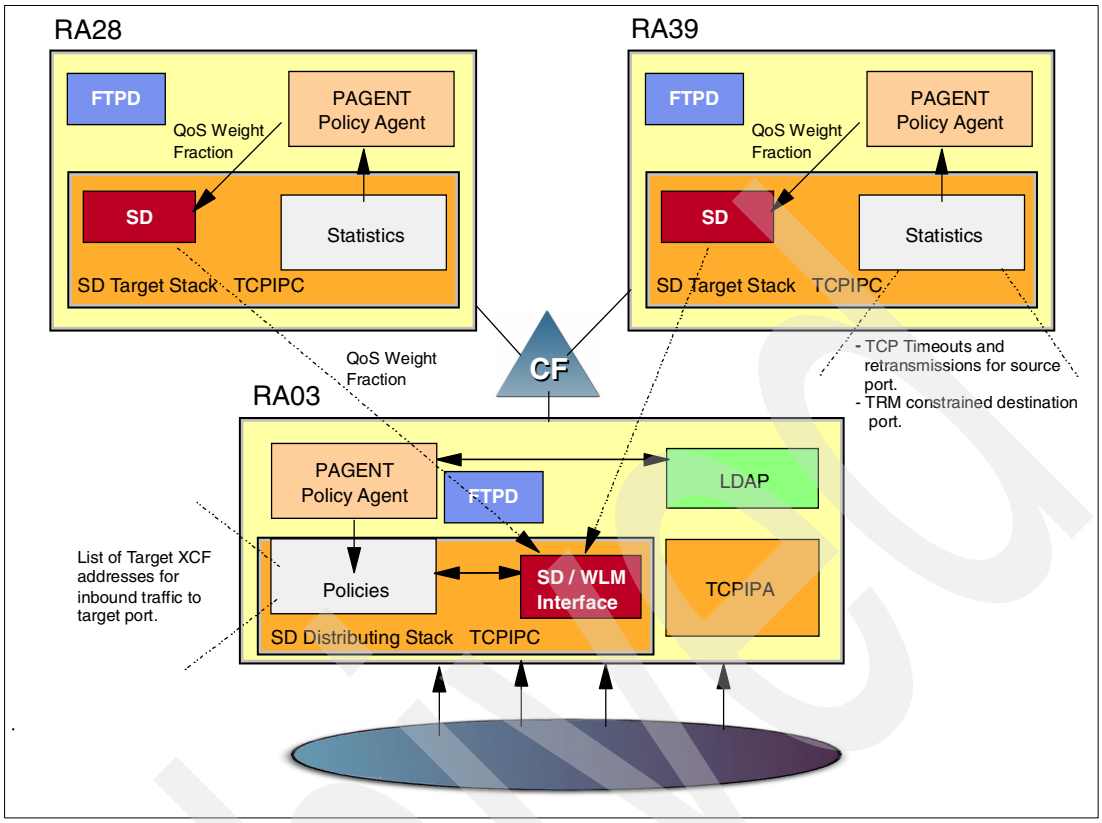


Figure 5-11 Sysplex Distributor policy implementation at ITSO Raleigh

5.6 Sysplex Distributor implementation

The implementation of Sysplex Distributor is very straightforward. The TCP/IP configuration that needs to take place is minimal compared to other connection dispatching technologies. For the most part, the sysplex environment enables the dynamic establishment of links and the dynamic coordination between stacks in the cluster.

5.6.1 Requirements

The IPCONFIG DATAGRAMFWD and DYNAMICXCF statement must be coded in the TCPIP.PROFILE data set in all the IP stacks of the sysplex cluster.

If you want to implement a WLM-based distribution, you have to register all IP stacks participating in the sysplex with WLM coding SYSPLEXROUTING in each IP stack. Also verify that all the participating CS for OS/390 V2R10 IP images are configured for WLM GOAL mode.

To enable the distributing IP stack to forward connections based upon a combination of workload information and network performance information, configure all the participating stacks for WLM GOAL mode. Specify SYSPLEXROUTING in the IPCONFIG statement in all the participating stacks and also define a Sysplex Distributor Performance Policy on the target stack with the Policy Agent. Otherwise, If SYSPLEXROUTING is not coded in any IP stack, the distribution for incoming connections to the target applications will be random.

For those OS/390 images that are running more than one IP stack, the recommended way to define XCF and IUTSAMEH links is to use the IPCONFIG DYNAMICXCF. In fact, IUTSAMEH links should not be specified if the IP stack is participating in Sysplex Distributor.

For any IP application that uses both control and data ports, both port numbers must be distributed by the same Dynamic VIPA address (for example FTP).

5.6.2 Incompatibilities

As specified in *OS/390 IBM Communications Server: IP Migration,* SC31-8512, IBM recommends that all the IP stacks that may be participants in a Sysplex Distributor environment be at V2R10 because of the following restrictions:

- Workload distribution through Sysplex Distributor is available only when both the distributing and target stacks are at V2R10.
- IP stacks at V2R7 and previous releases do not support automatic VIPA backup.
- IP stacks at V2R8 may back up a distributing Dynamic VIPA, but would be unable to distribute workload if the distributing stack were brought down. Moreover, VIPA takeback is disruptive in IP V2R8.

If any IP stack (different from the distributing one) is on the path from the client to the distributing IP stack, and if this intermediate IP stack is a target stack within the sysplex cluster, new connection requests passing through this intermediate IP stack might be routed to the local application and the service is not subject to distribution.

5.6.3 Limitations

FTP passive mode should not be used with Sysplex Distributor, as documented in *OS/390 IBM Communications Server: IP User's Guide,* GC31-8514. If the host for the secondary FTP server has the Sysplex Distributor function distributing FTP server workload, then the client should not use the proxy subcommand. The PASV command that is used by the proxy subcommand to allow the secondary FTP server to be the passive side of a data connection cannot be handled properly at the secondary FTP server host. Note that Sysplex Distributor cannot work with FTP passive mode because the FTP server uses ephemeral ports for the passive connection and ephemeral ports cannot be distributed by Sysplex Distributor function.

A short example is illustrated in Figure 5-12 on page 122 and described by the following steps:

1. The client that requests the passive mode connection to allow the FTP server to be the passive side of the data connection is using the DVIPA V1 of the Sysplex Distributor.
2. Sysplex Distributor in conjunction with WLM selects the target stack.
3. Sysplex Distributor forwards this control connection request to port 21 on the target stack. The FTP server recognizes the request for passive mode and selects an ephemeral port 1428 for the data connection.
4. This information about the ephemeral port is sent to the client over the control connection.
5. The client will use this ephemeral port (1428) to establish the data connection using the DVIPA V1 of the Sysplex Distributor.
6. The Sysplex Distributor will reject that request, since it is not aware of such a port number. Remember that we have to define, on the VIPADISTRibute statement, the ports for which we are to distribute workload.

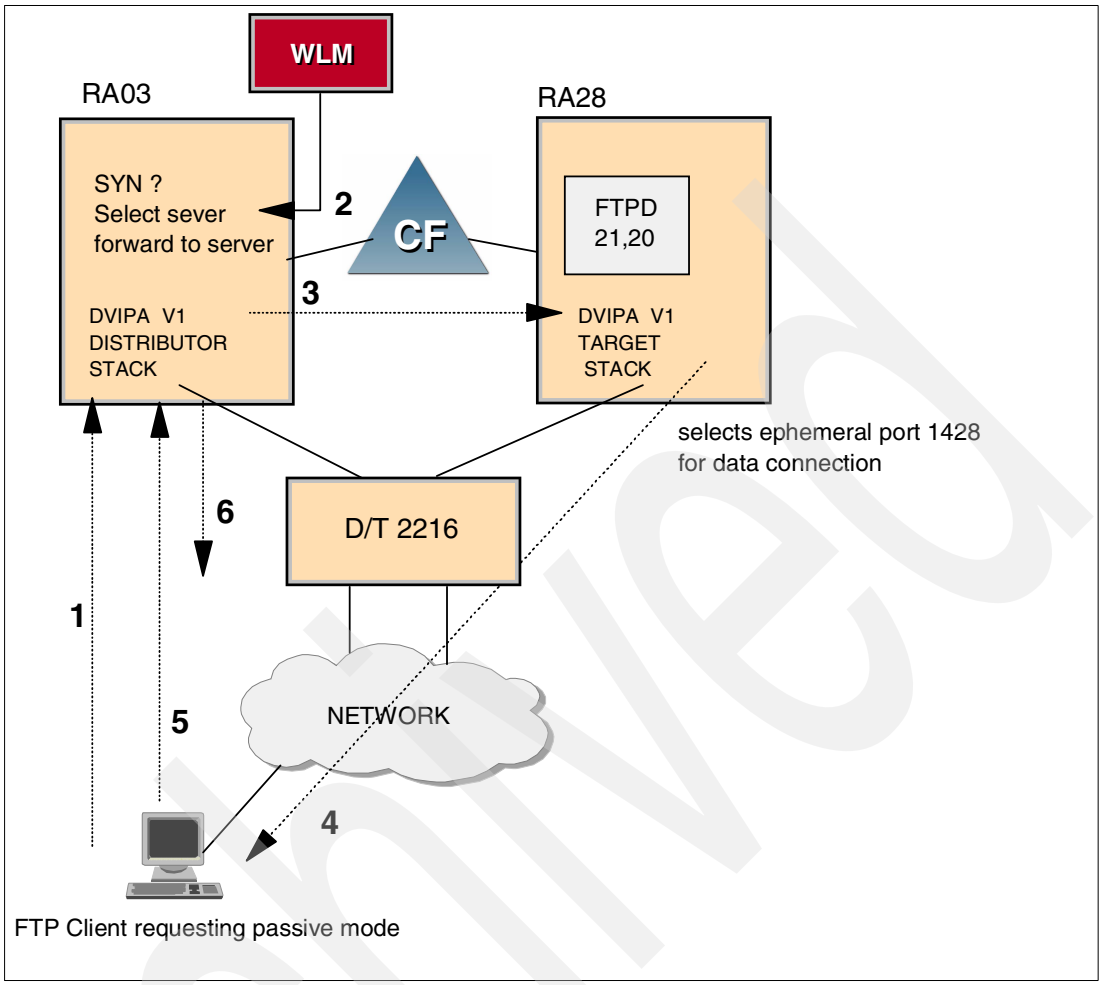


Figure 5-12 FTP passive mode limitations for Sysplex Distributor

During our tests we found another limitation that should be mentioned. If the Sysplex Distributor distributing stack is distributing workload for one port and not for some other (no VIPADISTribute statement for it is defined), the Sysplex Distributor stack will not allow the connection to that port even if a local application is listening on it.

Consider the example shown in Figure 5-13 on page 123:

- Assume that we want to distribute only TN3270E services to the participating stacks RA03, RA28, and RA39. For that we code the statements in Example 5-2.

Example 5-2 Coding for distributing TN3270E only

```
VIPADYNAMIC
   VIPADEFINE MOVE IMMED 255.255.255.0 172.16.251.3
   VIPADISTRIBUTE 172.16.251.3 PORT 23 DESTIP ALL
  ENDVIPADYNAMIC
```

- We have only defined one DVIPA to distribute the workload.
- If we now try to connect to the FTP or the Web server on the distributing stack RA03 using the same DVIPA, the Sysplex Distributor stack disallows access to these ports even though the distributing stack has these applications currently active. The connection requests for these applications time out.

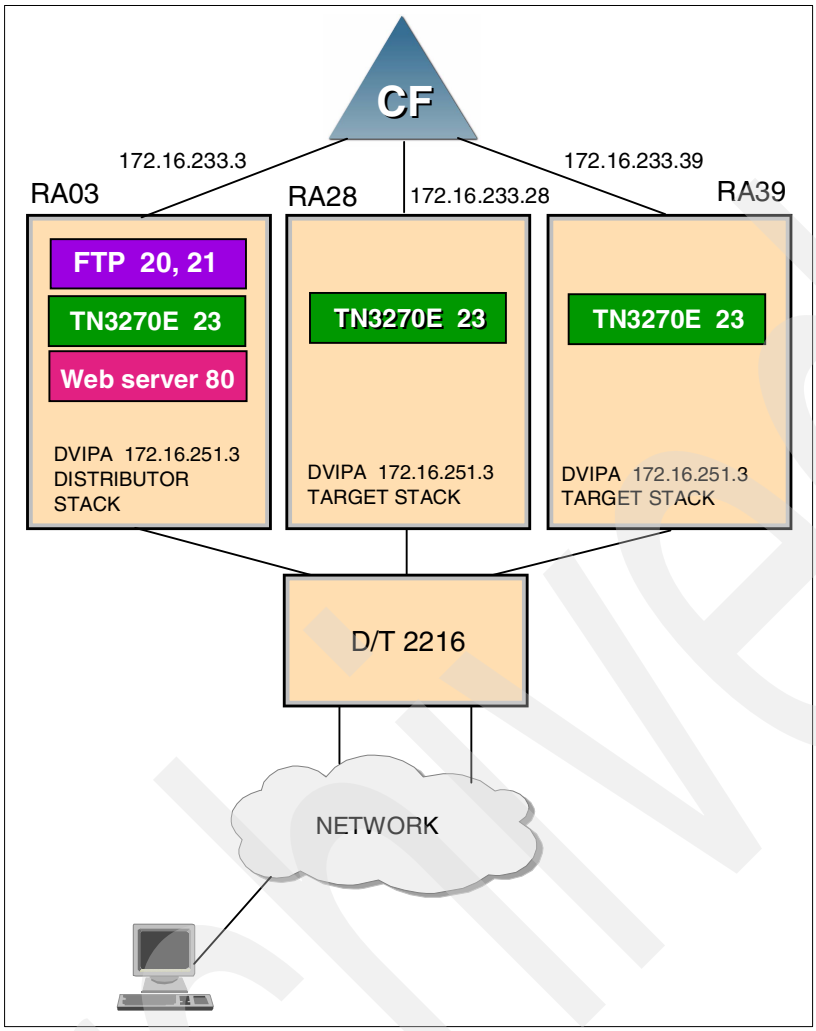


Figure 5-13 Sysplex Distributor limitation

There are actually two techniques to solve this limitation:

1. We could code another VIPADISTribute statement for the IP service (port) in question (FTP in our case) and explicitly define on the DESTIP keyword the <dynxcfip> address to which target stack it should be distributed (the local, distributing stack). However, note that you can only define four distributed ports per DVIPA. In our case, we would define this for the Web server using the VIPADISTribute statement in Example 5-3.

Example 5-3 VIPADISTribute statement

```
VIPADYNAMIC
   VIPADEFINE MOVE IMMED 255.255.255.0 172.16.251.3
   VIPADISTRIBUTE 172.16.251.3 PORT 23 DESTIP ALL
  ENDVIPADYNAMIC
```

   The configuration for the FTP server would be similar.

2. An alternative is to code one DVIPA for every different IP service (port). The limit of DVIPAs per stack has been raised to 256, thereby effectively eliminating this concern. However, you would have to use different hostnames/IP addresses to connect to the applications, which may pose a problem in already implemented environments in which the same DVIPA is already being used for multiple services.

5.6.4 Implementation

The following list shows what is needed to implement Sysplex Distributor:

1. Choose which IP stack is going to execute the Sysplex Distributor distributing function.
2. Select which IP stacks are going to be the backup stack for the Sysplex Distributor stack, and in what order.
3. Ensure that WLM GOAL mode is enabled in all the LPARs participating in the Sysplex Distributor.
4. Enable sysplex routing in all the IP stack participating in the Sysplex Distributor with the SYSPLEXROUTING statement.
5. For those IP stack that are active under a multi-stack environment, the samehost links have to be created dynamically. In general, code DYNAMICXCF in all the IP stacks participating in the Sysplex Distributor.

   Note: For Sysplex Distributor you cannot specify the XCF address using the IUTSAMEH DEVICE, LINK, and HOME statements. XCF addresses have to be defined through IPCONFIG DYNAMICXCF.

6. Code DATAGRAMFWD in all IP stacks participating in the Sysplex Distributor.
7. Select, by port numbers, the applications that are going to be distributed using the Sysplex Distributor function. Note that if the application chosen requires data and control ports, both ports have to be considered.
8. Code the VIPADYNAMIC/ENDVIPADYNAMIC block for the distributing IP stack:
   a. Define the dynamic VIPA associated to the distributing IP stack with VIPADEFINE statement.
   b. Associate the sysplex Dynamic VIPA to the application's port number with the VIPADISTRIBUTE statement.
9. Code VIPADYNAMIC/ENDVIPADYNAMIC block for the distributor's backup IP stack.
10. Define the IP stack as backup for the sysplex DVIPA address with the VIPABACKUP statement.

The way that Sysplex Distributor distributes the workload can be modified using the Policy Agent, as detailed in to 5.5, "Sysplex Distributor and policy" on page 118. For a detailed discussion about policy-based network management, refer to *IBM Communications Server for OS/390 V2R10 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228.

5.7 Monitoring Sysplex Distributor

Three new parameters have been added to the `NETSTAT` command (or the `onestat` command) and output for three already existing parameters have been modified in order to monitor the SYSPLEX DISTRIBUTOR activity. Table 5-1 on page 125 briefly describes those new and changed parameters.

Table 5-1 New and modified NETSTAT/onetstat parameter to monitor Sysplex Distributor

Command	Parameter	Type	Description
NETSTAT	VIPADCFG	New	Display Dynamic VIPA configuration data
onetstat	-F	New	Display Dynamic VIPA configuration data
NETSTAT	VDPT	New	Display DVIPA port distribution table
onetstat	-O	New	Display DVIPA port distribution table
NETSTAT	VCRT	New	Display DVIPA connection routing table
onetstat	-V	New	Display DVIPA connection routing table
NETSTAT	CONFIG	Changed	Dynamic VIPA information removed
onetstat	-f	Changed	Dynamic VIPA information removed
NETSTAT	VIPADYN	Changed	Expanded OUTPUT
onetstat	-v	Changed	Expanded OUTPUT
SYSPLEX	VIPADyn	Changed	Expanded OUTPUT

Refer to *OS/390 IBM Communications Server: IP User's Guide,* GC31-8514 for more information on those commands.

6

Securing the connection to the Internet

In this chapter we describe how to implement both network level and platform security in a sysplex member connected to a non-secure network. We first present an overview of our configuration, then we discuss in detail the different levels of security and how to implement them in a sysplex.

6.1 Our configuration

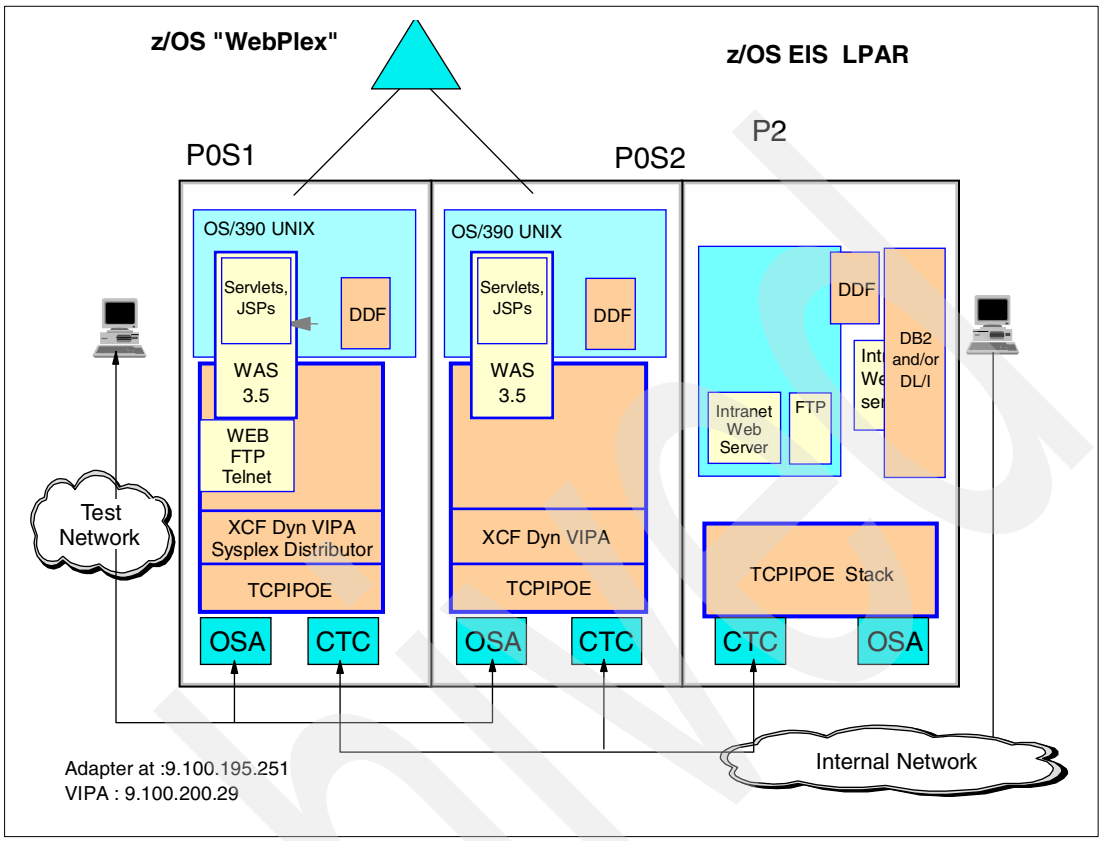


Figure 6-1 Our configuration

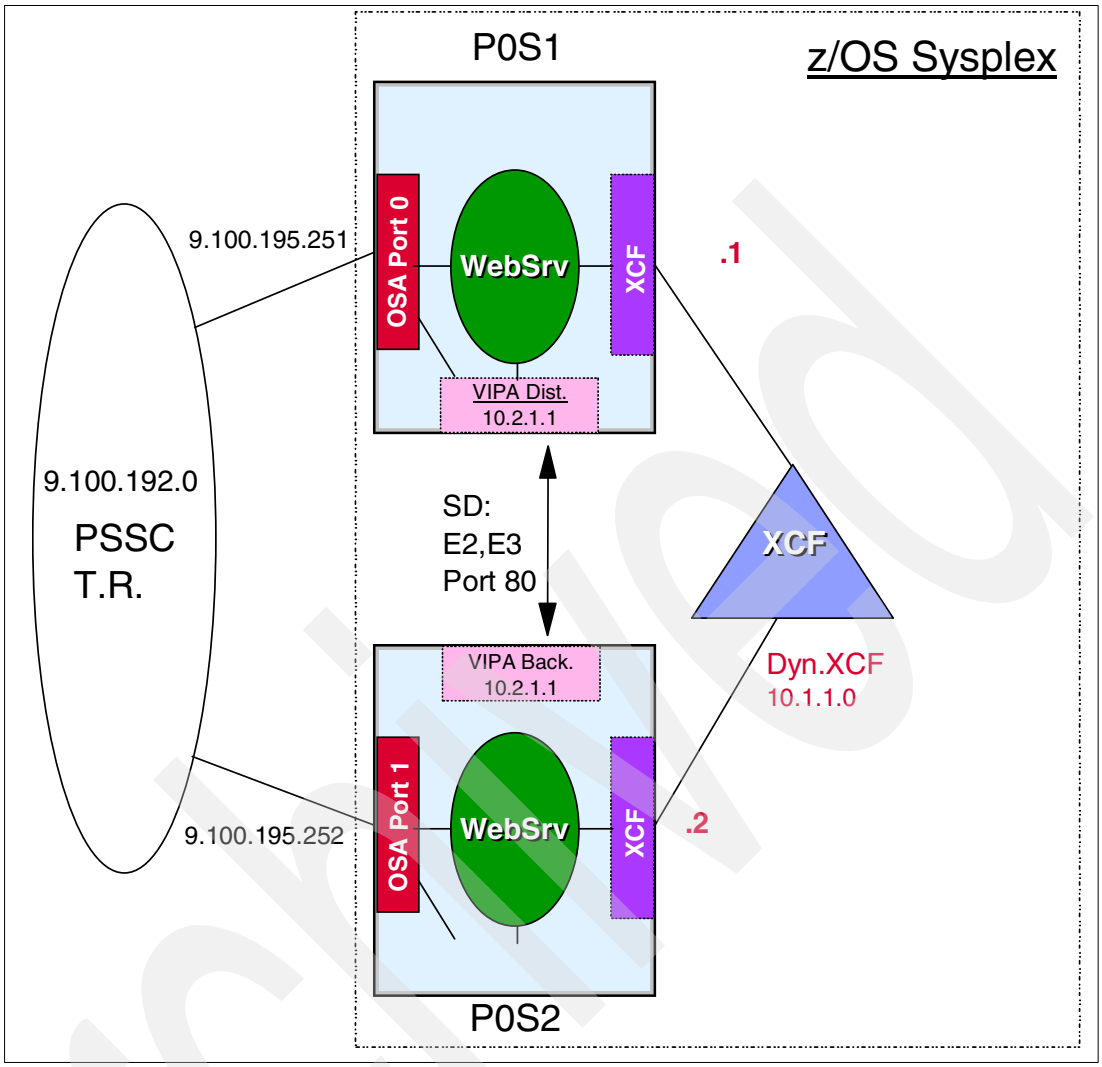


Figure 6-2 Our Sysplex Distributor

```
IPCONFIG
          IGNOREREDIRECT                ;
          DATAGRAMFWD                   ;
          FIREWALL                      ;
          VARSUBNETTING                 ;
          SYSPLEXROUTING                ;
          DYNAMICXCF 10.1.1.1 255.255.255.0 1
DEVICE  NAETH1    MPCIPA  NONROUTER AUTORESTART
 LINK    ENETLINK  IPAQGNET  NAETH1
START NAETH1
VIPADYNAMIC
     VIPADEFINE MOVE IMMED 255.255.255.252 9.100.200.29
     VIPADISTRIBUTE DEFINE 9.100.200.29 PORT 21 23 80
           DESTIP 10.1.1.1 10.1.1.2
ENDVIPADYNAMIC
HOME
   9.100.195.251  ENETLINK
```

Figure 6-3 System POS1 TCP/IP profile

```
IPCONFIG
          IGNOREREDIRECT              ;
          DATAGRAMFWD                 ;
          VARSUBNETTING               ;
          SYSPLEXROUTING              ;
          FIREWALL                    ;
          DYNAMICXCF 10.1.1.2 255.255.255.0 1
DEVICE  NAETH1     MPCIPA  NONROUTER AUTORESTART
 LINK    ENETLINK  IPAQGNET  NAETH1
START NAETH1HOME
VIPADYNAMIC
VIPABACKUP 10 9.100.200.29
ENDVIPADYNAMIC
HOME
9.100.195.252  ENETLINK
```

Figure 6-4 System POS2 TCP/IP profile

```
D TCPIP,,N,HOME
MR0000000 POS1 01324 21:16:31.76 STC01336 00000090  EZZ2500I NETSTAT CS V1R2 TCPIPOE 703
 LR                          703 00000090  HOME ADDRESS LIST:
 LR                          703 00000090  ADDRESS         LINK         FLG
 DR                          703 00000090  9.100.203.109   TRLC5        P
 DR                          703 00000090  9.100.195.251   ENETLINK
 DR                          703 00000090  10.1.1.1        EZASAMEMVS
 DR                          703 00000090  10.1.1.1        EZAXCFS2
 DR                          703 00000090  9.100.200.29    VIPL0964C81D
 DR                          703 00000090  127.0.0.1       LOOPBACK
```

Figure 6-5 Netstat HOME on POS1

```
NC0000000 POS2 01324 21:28:52.68 BRUNO    00000290  D TCPIP,,N,HOME
MR0000000 POS2 01324 21:28:52.75 STC01343 00000090 EZZ2500I NETSTAT CS V1R2 TCPIPOE 686
 LR                686 00000090  HOME ADDRESS LIST:
 LR                686 00000090  ADDRESS          LINK             FLG
 DR                686 00000090  9.100.203.122    TRLC5            P
 DR                686 00000090  9.100.195.252    ENETLINK
 DR                686 00000090  10.1.1.2         EZASAMEMVS
 DR                686 00000090  10.1.1.2         EZAXCFS1
 DR                686 00000090  9.100.200.29     VIPL0964C81D     I
 DR                686 00000090  127.0.0.1        LOOPBACK
```

Figure 6-6 Netstat HOME on POS2

```
D TCPIP,,N,VDPT
MR0000000 POS1 01324 21:31:54.54 STC01336 00000090  EZZ2500I NETSTAT CS V1R2 TCPIPOE 783
 LR         783 00000090  DYNAMIC VIPA DISTRIBUTION PORT TABLE:
 LR         783 00000090  DEST IPADDR     DPORT DESTXCF ADDR    RDY TOTALCONN  WLM
 DR         783 00000090  9.100.200.29    00080 10.1.1.1        000 0000000000 01
 DR         783 00000090  9.100.200.29    00080 10.1.1.2        000 0000000000 00
```

Figure 6-7 Netstat VDPT (Dynamic VIPA Distribution Port) on POS1

6.2 Implementing security at the network level

We have described in the previous chapters how to implement security at the operating system level, in our case by using RACF, delivered in the z/OS Security Server, to achieve the objectives of identification, authentication, access control, and accountability. When it comes to implementing security at the network level, although less obvious, we can achieve the security objectives described in ISO 7498-2:

- Access control: We want to control access to our secure network in the context of communications established between a secure network (intranet) TCP/IP host and another TCP/IP host located on the non-secure network (Internet). Because such a function is not built into the TCP/IP communication itself, one has to use additional devices, such as a “firewall,” to restrict this access on the basis of communication characteristics like the IP addresses of the parties involved in the communication, the direction of the communication, and so forth.
- Identification: Remaining strictly at the TCP/IP network level, the basic identification factor is the source host IP address, which, as everybody knows, can be easily faked in an IP datagram issued by attackers.
- Authentication: There is no built-in authentication mechanism in the basic TCP/IP communication itself, so we need to use the IPSec VPN technology with one of its two protocols (AH or ESP) to insure strong mutual authentication by having the involved two parties sharing a secret encryption key.

> **Note:** Other well known exploitations of cryptography, as in the SSL protocol, do not occur at the TCP/IP communication level but at the application level; therefore, they are not addressed in this chapter.

- Data integrity and data confidentiality: Here again, one would use the IPsec protocols (and shared secret keys between TCP/IP hosts) to achieve these objectives at an IP protocol level: AH (Authentication Header, IP protocol number 51) for data integrity and ESP (Encapsulated Security Payload, IP protocol number 50) for data confidentiality.
- Accountability: Since this is not built into the IP communication, one has to rely on the tracing done by devices such as firewalls. Needless to say, this would be too weak an evidence to be used in the context of non-repudiation disputes; where strong accountability is needed, we again need to go to the application level.

The technologies that we use to achieve these objectives are generically termed *firewall technologies*, which encompass the IETF-ratified IPSec technology and also vendor-specific code in the host TCP/IP stack. In z/OS, the firewall technologies comprise:

- IP filtering
- Network Address Translation
- FTP proxy
- SOCKS server
- IPsec VPNs

These technologies are explained in details in several redbooks: *Stay Cool on OS/390: Installing Firewall Technologies,* SG24-2046 and *Implementing VPNs in a z/OS Environment,* SG24-6530. The present discussion includes only those pieces of the firewall technologies that are appropriate for securing a sysplex member’s Internet connection.

6.3 General discussion on Internet threats

A lot has already been written on the threats one is exposed to when connecting to the Internet. All 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 is 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. Sensitivity to these configurations is therefore vendor, and even code-level dependant, and is information 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 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 r 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 are dealing 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.
- 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.

6.4 What z/OS can do for you

In this section we emphasize some aspects of z/OS security that are of interest to people new to the e-business environment and its risks and constraints. Some of these issues are discussed in more detail in other chapters of the book.

6.4.1 Platform-level security - RACF

An external security manager such as RACF provides for operating system, or platform-level security through the SAF interface. It provides services for identification, authentication, access control and, to some extent, accountability, which have been exploited for decades and are still the basis for platform security when connecting to the Internet.

However, the very special context of communicating with a non-secure network, and of using UNIX applications (such as z/OS TCP/IP, Web server, and so forth), leads us to emphasize specific RACF features here:

- The UNIX user dual identity in z/OS

  A z/OS UNIX user is first an MVS user, in that the user is given an MVS user profile in RACF with an OMVS segment that contains a UNIX UID. As soon as a UNIX UID is specified, any task run on behalf of this user will be given 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 6-8. A very important point to remember here is that a UNIX program authorized to switch UID on z/OS will also switch the MVS userid, in order to always conform to the dual identity as specified in the RACF profile. Great care must be exercised when dealing with these programs since they can also access MVS resources (they are running on z/OS) with the switch to the MVS userid.

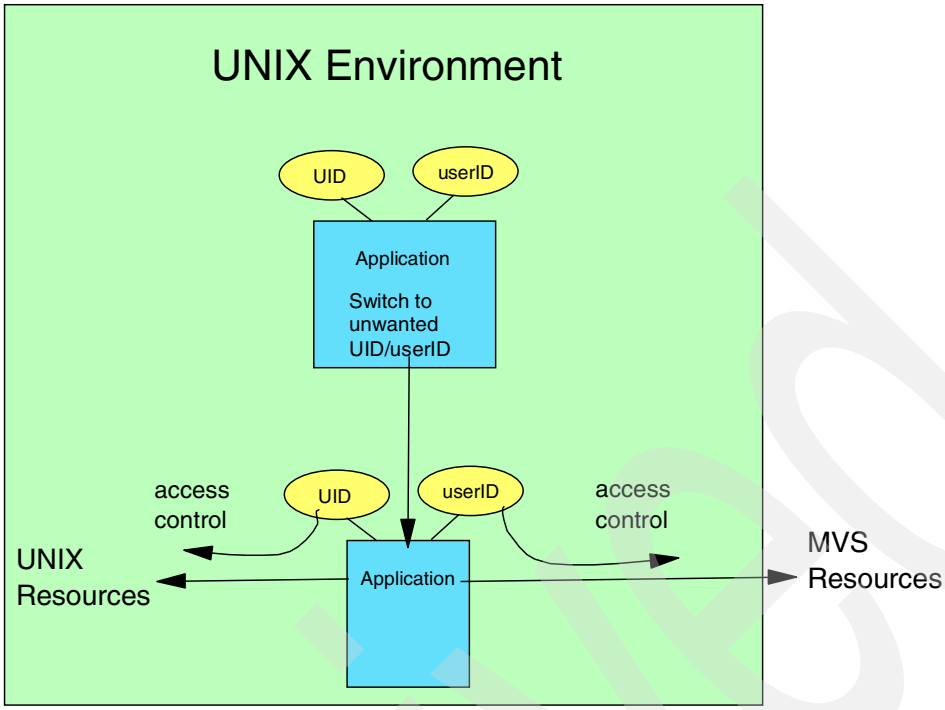


Figure 6-8 The z/OS UNIX user dual identity

The issue gets even more critical since you might have in your installation several UNIX users sharing the same UID. Although IBM does not recommend such a setup, there is nothing to prevent several users from having the same UID in UNIX, and when one z/OS UNIX application switches to one of these shared UIDs, it is usually not predictable what the corresponding MVS userid picked up by RACF will be. Actually, we can say that RACF is going to pick up the first user profile with the desired switched-to UID and will give, accordingly, the corresponding MVS userid to the task. Think of what could happen if you are sharing the same UID between MVS users, of which one is SPECIAL. Therefore it is most important to understand the additional protection facilities that RACF offers, as described in the following items, to strictly control who can eventually switch UID using the C run-time functions:

- setuid(), seteuid(), setruid()
- pthread_security_np
- _login()
- _passwd (verify/change password)

- UNIX Security versus z/OS UNIX security

  Defining the BPX.DAEMON profile in the FACILITY class is a mandatory setup to achieve control of UID switching. Without the BPX.DAEMON defined, UNIX superusers can easily exploit the critical functions listed previously. With BPX.DAEMON defined, z/OS provides the *z/OS UNIX Security,* where:

  - It is not sufficient to have UID(0) to perform:
    - setuid(), seteuid(), setruid()
    - _login()
    - _passwd (verify/change password)

    The MVS userid of the task requesting the function must also be permitted (READ) to the BPX.DAEMON profile. The idea here is that only applications which have been thoroughly checked for proper integrity of operations will be permitted to the profile and hence to switch UIDs (and by the same token, to switch MVS userids).

– BPX.DAEMON being defined also allows the exploitation of the RACF Program Control facility (explained in the next item) to check for the integrity of the address space the function is requested from.

Another FACILITY profile, BPX.SERVER controls access to the pthread_security_np function.

- Program Control is a facility initially put in place in MVS and RACF to control who can execute programs. *Profiles* can be created in the RACF PROGRAM class for MVS load modules or MVS program libraries. These profiles have a UACC and an access list field where users and groups can be permitted to execute these programs. The program control class is enabled in RACF with a SETR WHEN(PROGRAM) command.

  This facility also controls the environment of the executing address space: as programs are loaded into storage, the MVS load or exec functions check that they are getting the loaded programs from the sources specified in the PROGRAM class profiles. Any load module loaded from a source not declared in the PROGRAM profiles will turn on the so-called “dirty bit” in the address space TCB. The dirty bit is eventually checked by the system when further action is needed for which address space integrity is required. This mechanism has been extended to support source integrity checking for UNIX executables as well. These executables reside in a HFS file that can be flagged as “program control” using the “p“extended attribute. (Remember, there are no RACF profiles for the HFS files).

  So, when an address space contains a program (MVS load module or UNIX executable) fetched from a non-controlled location, the address space is marked as dirty. No program in a dirty address space will be able to do any of the controlled functions, which are:

  – setuid(), seteuid(), setruid()
  – pthread_security_np
  – auth_check_resource_np
  – _login()
  – _spawn() (with UserID switch)
  – _passwd

  All BPX.SERVER and BPX.DAEMON privileges are revoked, including the right to check passwords. Assuming also that there is a very strict control on who can install programs in program controlled libraries or HFS files, this facility provides a built-in protection against rogue programs (trojan horses, virus, and so forth) or simple code mistakes invoking critical system functions, which makes it mandatory for a system connected to a non-secure network.

  To illustrate this, Table 6-1 shows the C run-time functions that involve switching UID, and therefore MVS userid on z/OS. (The information in this table assumes that BPX.DAEMON is defined.)

Table 6-1 C run-time functions involving switching UID

USS function	Change effective UID	Change MVS UserID	Additional controls (RACF)
setuid flag	Yes	No	
su in shell	Y = 0	No	userID access to PBX.SUPERUSER
su with userid and password	Y	Y	
su with userid without password	Y	Y	userID surrogate to target userID
setuid() or _login with authentication of target ID	Y	Y	clean address space

USS function	Change effective UID	Change MVS UserID	Additional controls (RACF)
setuid() or _login without authentication of target ID	Y	Y	userID surrogate of target or (userID has UID(0) + access to BPX.DAEMON) + clean address space[1]
pthread_security_np with authentication of target ID	Y	Y	userID permitted to BPX.SERVER + clean address space
pthread_security_np without authentication of target ID	Y	Y	userID 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	userID surrogate of target or (userID has UID(0) + access to BPX.DAEMON) + clean address space

1. BPXROOT if target UID(0)

Following are some “good practice” tips:

- If possible, give to RACF SPECIAL users an OMVS segment without a UID field. The SPECIAL userid will never be associated to a UNIX UID (and will therefore never be involved in a UID switch). The OMVS segment has to be present as an additional security measure to prevent the use of the BPX.DEFAULT profile to provide a default UID.
- The default superuser userid specified in SYS1.PARMLIB(BPXPRMxx) member with the SUPERUSER keyword must not be permitted to BPX.DAEMON.
- Whenever possible, give the attribute NOPASSWORD to surrogated userids.
- When running the OS/390 and z/OS IHS:
  - Carefully implement separation of HFS subtrees between servers via the pass or exec directives in the IHS configuration file.
  - Be aware that, contrary to what the documentation says, IHS does not need access to BPX.DAEMON
  - Implement APAR PQ41777 to be able to give to the IHS stack a UID>0.

6.4.2 z/OS TCP/IP stack security

The z/OS TCP/IP stack security is provided through:

- Code robustness

  Beginning with OS/390 V2R6, the OS/390 and z/OS testing process in the laboratory comprises the execution of hacking and penetration test cases gathered by GSAL (Global Security Analysis Lab, at IBM's T.J. Watson Research Laboratories) to assess the robustness of the TCP/IP stack. Code is then added or fixed to withstand the attacks.

  Note: IBM does not publish a list of attacks tested against.

- z/OS UNIX security

  z/OS TCP/IP and the TCP/IP services are UNIX applications in that they are using z/OS UNIX System Services. As such, the userid given to the involved tasks has also a UID and must be permitted, whenever necessary, to the BPX.DAEMON and BPX.SERVER profiles (refer to the installation directives). In addition, program control security must be applied so that the status of their execution address space, “clean” or “dirty,” can be continuously checked by the system.

- RACF protection of TCP/IP resources

  TCP/IP resources such as stacks, ports, and networks can be secured with RACF profiles in the SERVAUTH. class.

- Built-in protections

  The TCP/IP stack has built-in protections, such as the automated discard of non-completed connection requests (SYN flood attack), and offers the capability to enable the firewall technologies and the Intrusion Detection Services (IDS, beginning with z/OS V1R2).

- Quality of Service (QoS)

  Although service policies are often thought of in terms of performance, they may also be used to enforce certain types of security, specifically to enforce traffic filtering. Security filtering is an often overlooked feature of the policy agent; one frequently assumes that filtering technologies that limit or block connections reside only in firewall code, when, in fact, the policy agent can play a role here as well.

Note: Policy agent versus firewalls

Firewall filtering can be used to block or limit traffic that enters, exits, or traverses the node on which it is defined. Policy agent filtering affects only traffic whose endpoint is on the CS for z/OS IP node. That is, forwarded traffic is generally not affected by these filters. Bear in mind that the Sysplex Distribution function views all participating z/OS nodes as one logical node. Therefore, the Sysplex Distributor (SD) function, in which the SD may choose to route traffic to specific distribution targets based upon an OutboundInterface parameter, can route packets whose endpoint is not the CS for OS/390 IP.

Furthermore, the packet blocking in policy agent should not be viewed as a replacement for the robust filtering available in the z/OS firewall technologies. Policy agent filtering is fast and easy to implement. It could easily represent one of the first phases of a security implementation, to be augmented later by firewall filtering and IPSec.

Note: QoS rules versus blocking rules

Rules that intend to set QoS, RSVP, or SD policies, including connection limits, apply only to traffic flowing outbound. Policy agent does not set QoS or RSVP values for inbound traffic. The one exception to this statement is when the SD node views itself together with the target stacks logically as one entity and will examine inbound traffic that is *in transit* to a distributed stack under the aegis of the sysplex distribution function.

Rules that intend to *block* traffic apply to both inbound and outbound traffic, whether or not the traffic originates at CS for OS/390 IP.

Suppose you want to limit, but not block, concurrent FTP connections from a specific network to a mere fifty (see Example 6-1). In other words, you want to define a QoS policy for TCP traffic that does not completely filter out (“block”) FTP traffic, but just restricts it.

Example 6-1 Limiting inbound FTP connections: Policy agent Version 2 definition

```
PolicyAction ftpaction
  {
  MaxConnections 50                # Limit FTP concurrent connections to 50.
  MaxRate 400                      # Limit FTP connection throughput to 400 Kbps.
  OutgoingTOS 01000000             # the TOS value of outgoing FTP packets.
  }

PolicyRule ftprule
  {
  ProtocolNumberRange   6
  SourcePortRange       20 21
  DestinationAddressRange    172.16.232.1 172.16.252.254
  policyactionreference ftpaction
  }
```

The PolicyAction named *ftpaction* allows a maximum of fifty connections across *both* FTP ports while regulating throughput to 400 Kbps. The PolicyRule named *ftprule* references this PolicyAction and is invoked when packets identified as FTP (TCP protocol #6) need to flow from port 21 (the well-known FTP control port) or from port 20 (the well-known FTP data port) to remote addresses ranging from 172.16.232.1 through 172.16.252.254.

Note: Since QoS policies are enforced only for outbound traffic and not for inbound traffic, a good rule of thumb is to indicate the SourcePortRange and the DestinationAddressRange in the PolicyRule.

Can you completely stop certain types of traffic using policy agent? The following set of definitions (Example 6-2) illustrates how to accomplish this for inbound telnet traffic from network 192.168.5.0.

Example 6-2 Blocking inbound Telnet traffic: Policy agent Version 2 definition

```
PolicyAction telnet-block
  {
  Permission Blocked               # Do not permit inbound telnet
  }

PolicyRule telnetin-block23
  {
  DestinationPortRange             23 23
  SourceAddressRange               192.168.5.1     192.168.5.254
  policyactionreference            telnet-block
  }
```

This definition causes the stack to discard all packets from subnet 192.168.5.1/254 for Telnet at Port 23. Yet there might be other telnet servers at other ports available on the IP stack, including a UNIX telnet server. Perhaps it is not important to block access to those telnet servers because you have other security mechanisms in place for them. However, if it is critical to inhibit all attempts to reach any telnet server at z/OS, you need to include a PolicyRule for each of them.

Note: Rule of thumb: Blocking traffic

Recall from earlier notes that QoS policies are applied to traffic flowing outbound from CS for z/OS IP. However, blocking policies are applied to both inbound and outbound traffic flow. Generally speaking, if we want to block inbound traffic to CS for z/OS IP, we would name a CS for z/OS IP “destination port” with a “source address” of the remote host or (sub)network. Likewise, if we want to block outbound traffic to a remote destination, we would define a “destination port” at the remote site and a CS for z/OS IP “source address.” However, a distinction is made as to how policy agent applies this general rule:

- For UDP or RAW socket flows, the origination of the request is irrelevant. The direction of the traffic flow determines whether a particular blocking rule applies or not. No other factors are considered.
- For actions specifying a permission of *blocked*, TCP traffic is handled differently depending on whether the connection request is inbound or outbound:
  - When an inbound request is received, an inbound rule is checked to see if the SYN should be accepted. If it is, then the outbound rule is checked to see if the connection is allowed. If both the inbound and outbound rules indicate that it is all right to accept the connection, then a tcb (TCP connection control block) is built. Any other QoS rules, for example, TOS settings or similar, may now be applied to the outbound connection. If the inbound rule permits the connection but the outbound rule does not, the tcb is not built and the connection request is rejected.
  - If an outbound TCP SYN request is generated and there is no outbound blocking rule in effect, the tcb is created and any outbound QoS rules are applied. Possible inbound blocking rules are ignored. When the SYN/ACK arrives back at the CS for z/OS IP server, a tcb with an assigned outbound QoS already exists and there is no further checking to see if an inbound blocking rule is in effect.

Note: Policy agent as security

We cannot emphasize this enough: To put this description of policy agent as a security mechanism in perspective, PAGENT should be considered as just one more “line of defense.” You may choose to use it as such behind an external firewall, or local firewall filtering technology, or even CS for OS/390 V2R10 IP access control capabilities described elsewhere in this book.

- WLM classification

  This item addresses the capability to limit the consumption of system resources by the TCP/IP stack when it goes under a denial of service attack, as described in WLM classification, when running in goal mode with proper service classification.

6.5 Exploiting the z/OS firewall technologies in a sysplex

The firewall technologies available beginning with OS/390 V2R5 are functions embedded in the TCP/IP stack or running in additional address spaces that provide protection mechanisms which were formerly delivered by dedicated firewall boxes. The protection mechanisms of interest here are the IP Filtering and the IPSec Virtual Private Network because of their usefulness and specific features that need to be understood when they run in a sysplex.

6.6 IP filtering

The IP filtering mechanism runs in the z/OS TCP/IP stack and analyzes on the fly all IP packets traversing the stack or being sent to or issued by the stack itself. A high level layout is given in Figure 6-9. The IP Filtering function is controlled by “filtering rules” that tell it what to look at in the IP packet and what decision (accept or deny the packet) has to be made accordingly. The filtering criteria are:

- Source and/or destination IP address
- Source and/or destination port numbers
- Underlying protocol (tcp, udp, icmp, tcp with ack)
- Interface the packet is at (secure/non-secure)
- Routing status (local to the TCP/IP stack or to be routed)
- Direction (inbound or outbound)

The IP filtering code in the z/OS TCP/IP stack, when receiving a packet, examines the rules from top to bottom and takes the indicated action on first match. All rule sets include a “deny all” entry at the bottom

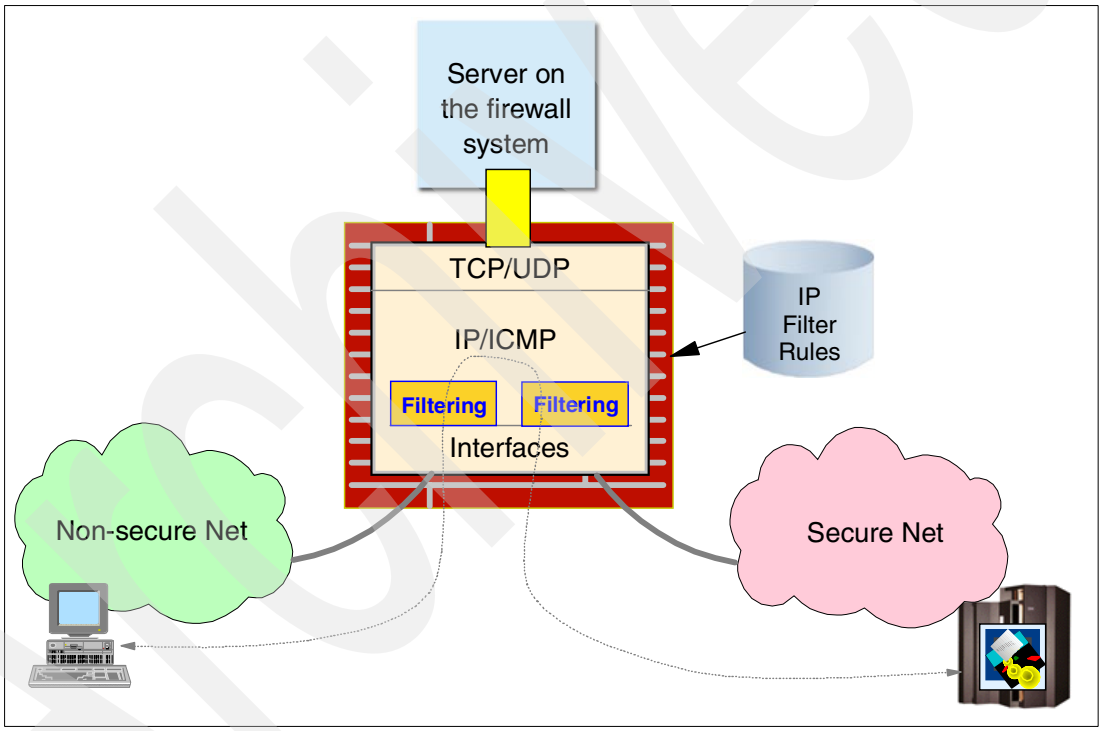


Figure 6-9 z/OS IP filtering

6.6.1 z/OS IP Filtering and sysplex

The IP Filtering operates for communications received via a network adapter such as an OSA card, but also works on IP packets transmitted by TCP/IP over XCF or transmitted via hipersockets.

IP Filtering also operates for VIPA addresses, as shown in Figure 6-10. In this figure we give an example, at a very high level, of IP filtering directives that would allow the HTTP traffic to occur between the outside world and Member 1, where a sysplex distributor stack operates, and a target stack in Member 2. On the other hand we want to prevent communications between Member 3 and the outside world.

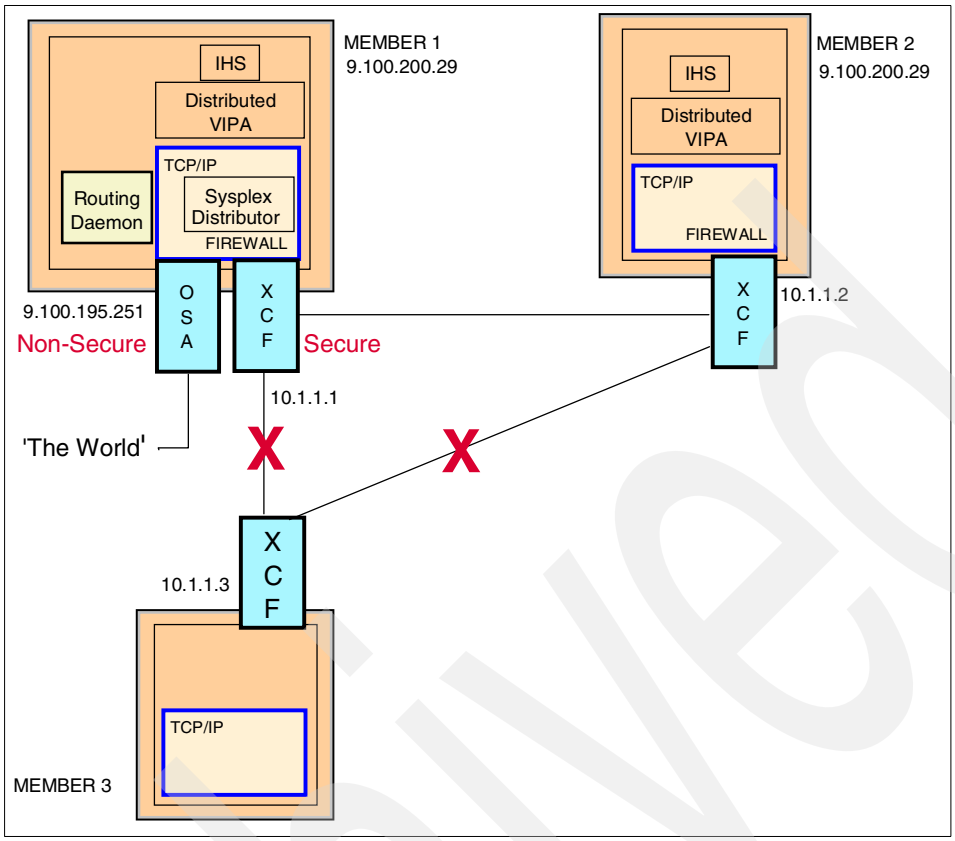


Figure 6-10 IP Filtering and Sysplex Distributor

The distributed VIPA is 9.100.200.29; the IP addresses of the physical adapters are 9.100.195.251 (to the outside world) and 10.1.1.1, 10.1.1.2, and 10.1.1.3 for the IP over XCF connections. Roughly, the rules here will be:

- In Member 1, on the non-secure interface (9.100.195.251):
  - Permit TCP from The World to port 80 at 9.100.200.29, inbound and outbound, routed and local. This takes care of HTTP communications, both ways and distributed to another stack by the stack in member 1.
  - Deny everything else that must not go through. That takes care of blocking any communication between Member 1 and any member other than Member 2.
- In Member 1, on the secure interface (10.1.1.1):

  Permit TCP from the world to port 80 at 9.100.200.29 inbound and outbound, routed only.
- In Member 2, deny all communications with 10.1.1.3.

Note: If policies are used for distribution by Member 1, then the filtering rules should allow communications between 10.1.1.1 and 10.1.1.2 for proper policy agent communications.

Important: As soon as the firewall function is invoked, (this is done by starting a TCP/IP stack with the `IPCONFIG FIREWALL` statement), a set of rules is activated, by default, until the customer's filters can be set up. This default set of rules does not allow routing between the non-secure and secure interfaces of the stack, and this translates into the Sysplex Distributor mechanism not providing the desired routing to the target stack. A quick way to avoid this problem is to activate the z/OS firewall "rule services 25," which allows all communications be it local or routed. But, again, this is intended to be a temporary measure until the installation implements its own filtering rules (and of course these rules should allow routing towards the Sysplex Distributor target VIPA).

6.6.2 IPSec Virtual Private Network

IPSec (IP Security) is a protocol aimed at protecting the communications between two TCP/IP stacks by ensuring data integrity and data privacy by cryptographic means. The sharing of the secret cryptographic key between the two ends of the communications can also be considered as a strong authentication of the other party since this key only exists at both ends of the VPN tunnel, in a set of resident information called a "Security Association."

The z/OS IPSec VPN Technology is implemented as shown in Figure 6-11, and comprises an IKE (Internet Key Exchange) daemon, also called a "key server," which is in charge of automatically installing and refreshing the encryption keys.

Note: These are the IP filtering rules which are calling for the creation of a VPN tunnel.

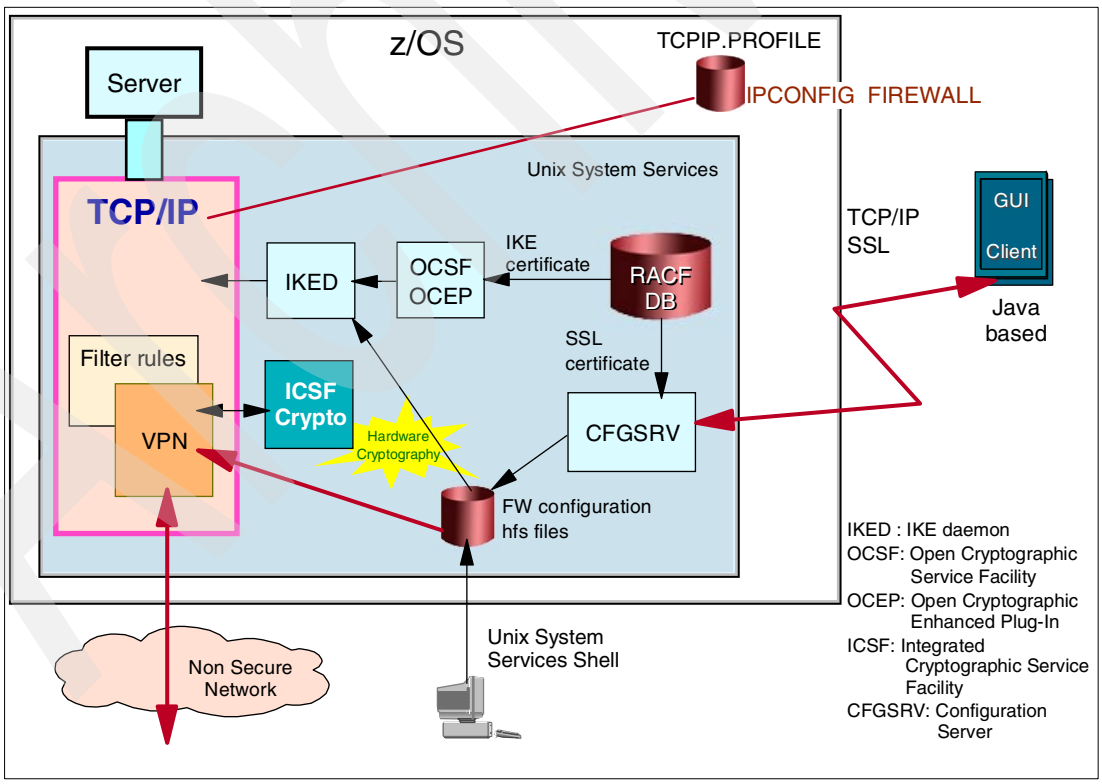


Figure 6-11 z/OS Implementation of IPSec VPN Technology

To automatically install the encryption keys, the IKE daemon communicates with a peer daemon at the other end of the VPN in two phases, as shown in Figure 6-12 on page 143.

Phase 1 is intended to install a random key to protect the communications between the two IKE daemons, and during Phase 2 the actual data encryption keys, also random ones, are installed in the two communicating stacks, securely transmitted encrypted by the key generated during phase 1. At the completion of phase 2, data flowing between the two stacks is protected by the IPSec protocol according to the active filtering rules.

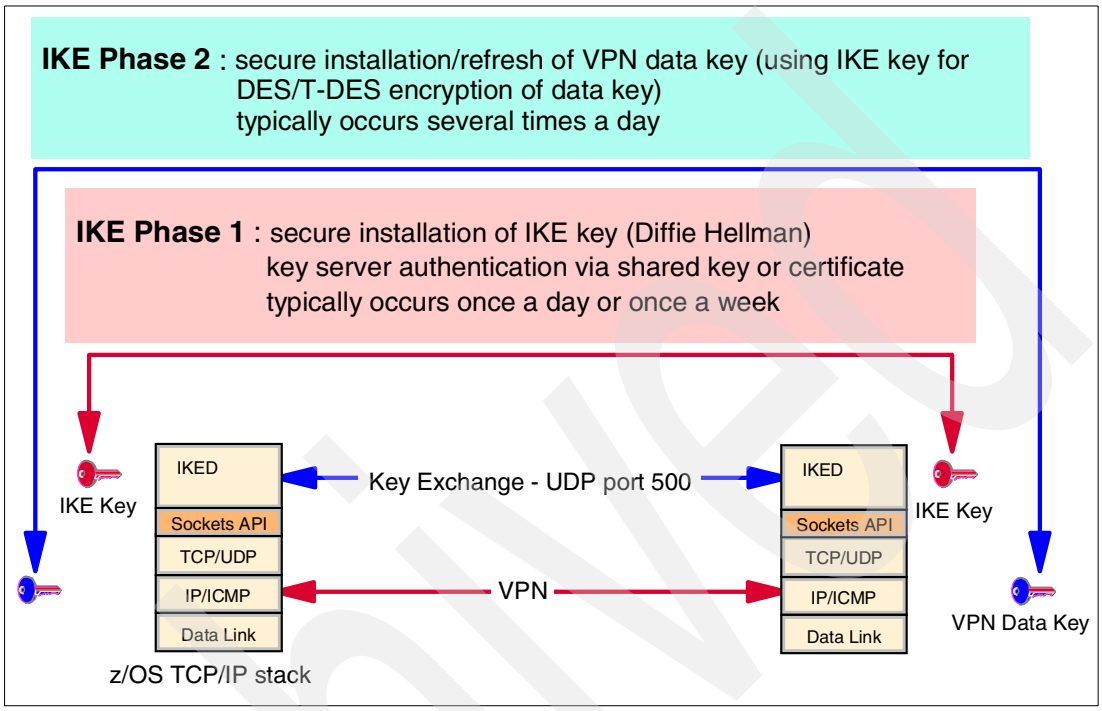


Figure 6-12 The two phases of the IKE protocol

6.6.3 IPSec VPNs and Parallel Sysplex

The implementation of VPNs in a sysplex can be considered in two cases:

- Between sysplex members

  The use of a VPN between sysplex members is generally not recommended because of the overhead of cryptographic operations. This recommendation is acceptable from the security standpoint if the intra-sysplex TCP/IP communications occur over XCF links or hypersockets, or even on a private network.

- Between a sysplex member and the outside world

  VPN is typically implemented in this case to protect the data flowing over the non-secure network between the sysplex and another system, and to provide strong authentication of the two involved parties. This configuration is quite feasible when the protected communications are not intended to be distributed within the sysplex, that is, the end point of the communication is the sysplex member connected to the non-secure network, as shown in Figure 6-13. If the front-end sysplex member acts as a sysplex distributor, then the use of IPSec is constrained.

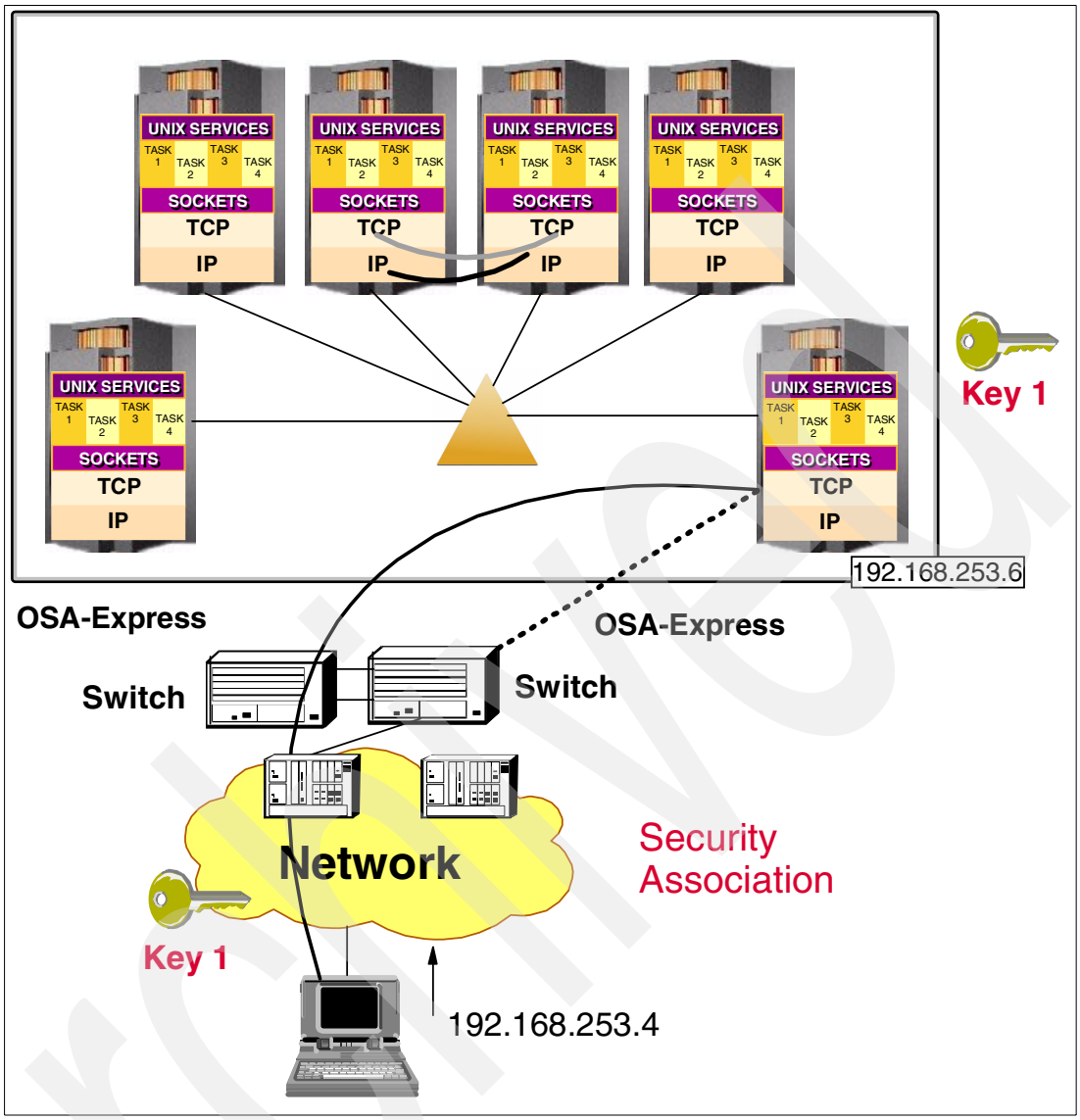


Figure 6-13 IPsec VPN with the Internet Sysplex Member

IPSec VPNs and sysplex distributor

Transposing this mechanism to a sysplex distributor configuration raises the problem of sharing the VPN encryption keys between the distributor's target servers, as shown in Figure 6-14. This was not possible until z/OS V1R4 and its "Sysplex Wide Security Associations" feature.

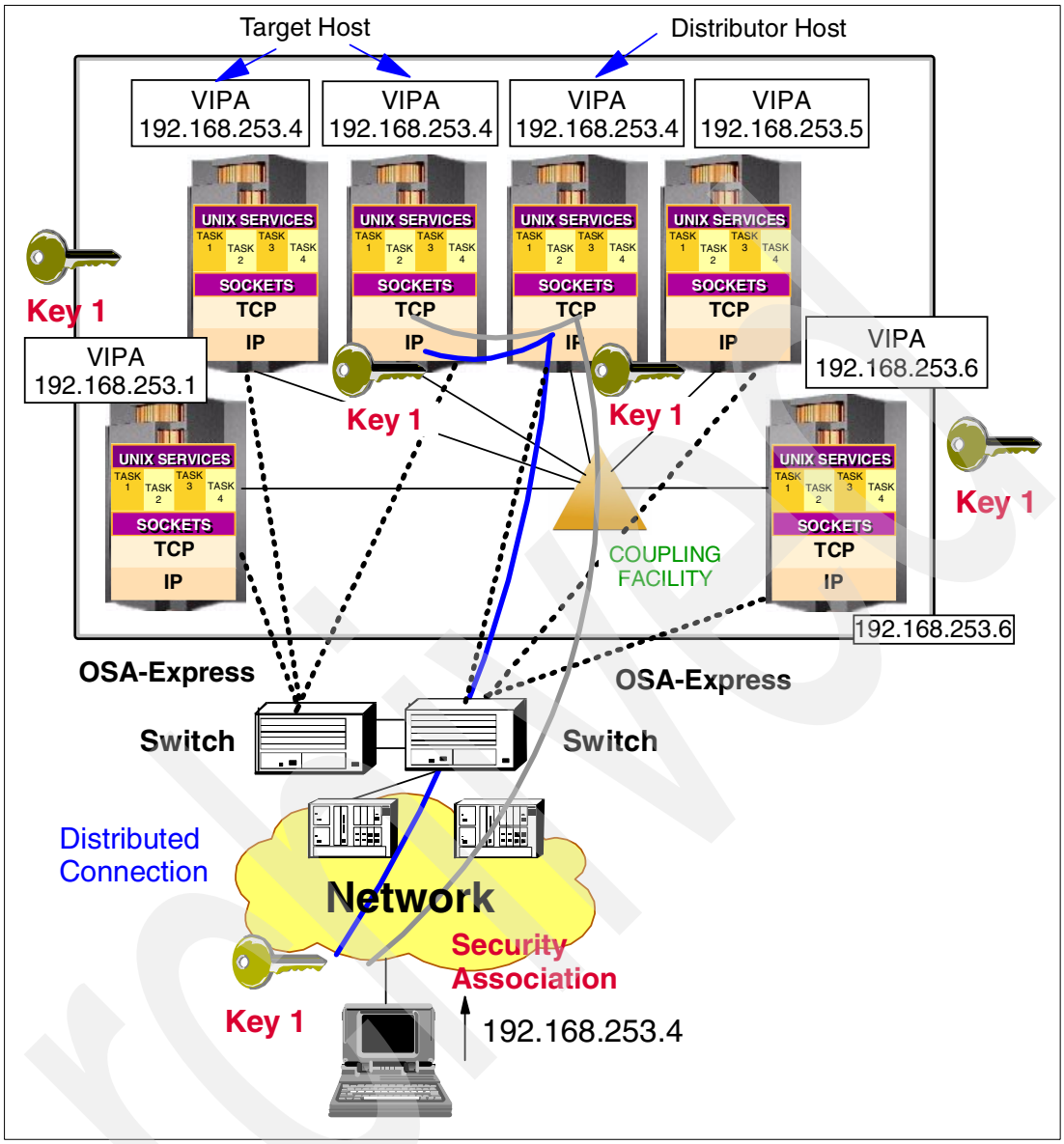


Figure 6-14 Sysplex Wide Security Associations

Sysplex Wide Security Associations (z/OS V1R4)

More details on this topic can be found in the *Communications Server IP Configuration Guide for z/OS V1R4,* SC31-8775. The Security Associations data, including the cryptographic keys used in IKE phase 1 and 2, are kept in the coupling facility and are thus available to all TCP/IP stacks in the sysplex which have been set up to process IPSec-protected communications. This allows:

- Distribution of the IPSec-protected workload with the distribution of connections.
- IPSec phase 1 and 2 Security Associations can be automatically restarted on backup after VIPA takeover.

Important: The distribution of the IPSEc requires that the same filtering rules be used across the participating members.

6.7 Network security configurations

6.7.1 The demilitarized zone (DMZ)

The “demilitarized zone” (DMZ) is a term often used when describing firewall configurations; Figure 6-15 shows a typical example. A DMZ is an isolated subnet between your secure network and the Internet. Much like the no-man’s land between two entrenched armies, anyone can enter it, but the only things present are those which 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.

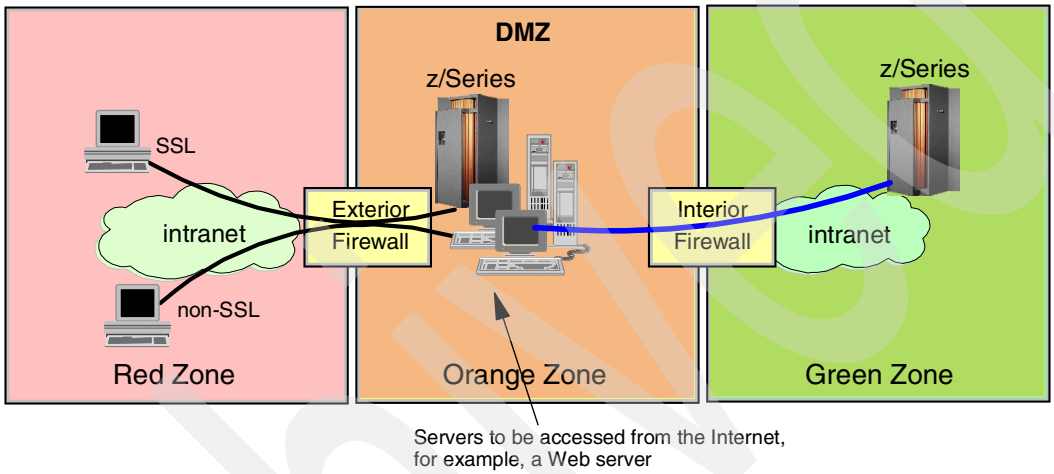


Figure 6-15 The DMZ principle

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) between your secure network and the DMZ. The outer firewall is designed to allow into the DMZ only those requests you wish to receive at your Web servers, but could also be configured to block denial-of-service attacks and to perform network address translation of the servers in your DMZ. The inner firewall is designed to prevent unauthorized access to your secure network from the DMZ and also, perhaps, to prevent unauthorized access from your secure network to the DMZ or the connected non-secure network. When you put a z/OS server into a DMZ, we would strongly suggest that you use z/OS firewall technologies. You should use z/OS firewall technologies 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 insecure network and this server.

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 6-16, 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 V1R2 IDS is activated and monitors inbound packets.

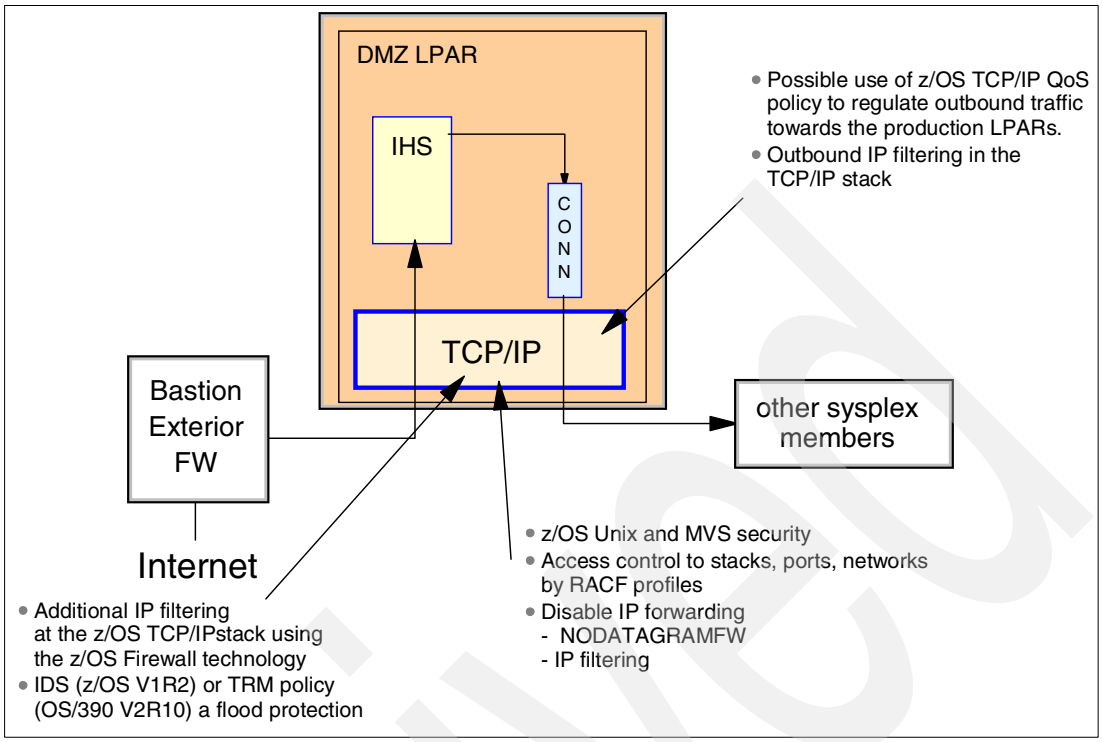


Figure 6-16 Enforcing z/OS DMZ stack security

Additional notes

- It is recommended that IP forwarding be disabled so that all communications with the rest of the sysplex have to go through IHS 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 TCPIP.PROFILE
  - 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.

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 the next section), 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 “by-pass” 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/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.

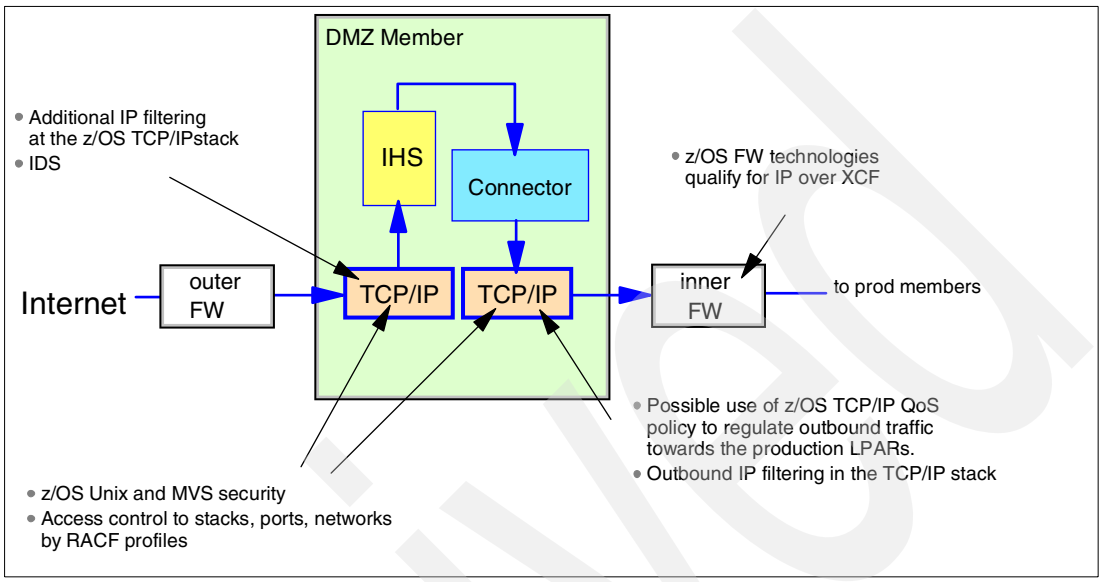


Figure 6-17 Dual-stack configuration

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 (C-INET). The C-INET 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, whereas 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.

Our test setup is illustrated in Figure 6-18. 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 6-3, with the same name given to their STC procedure.

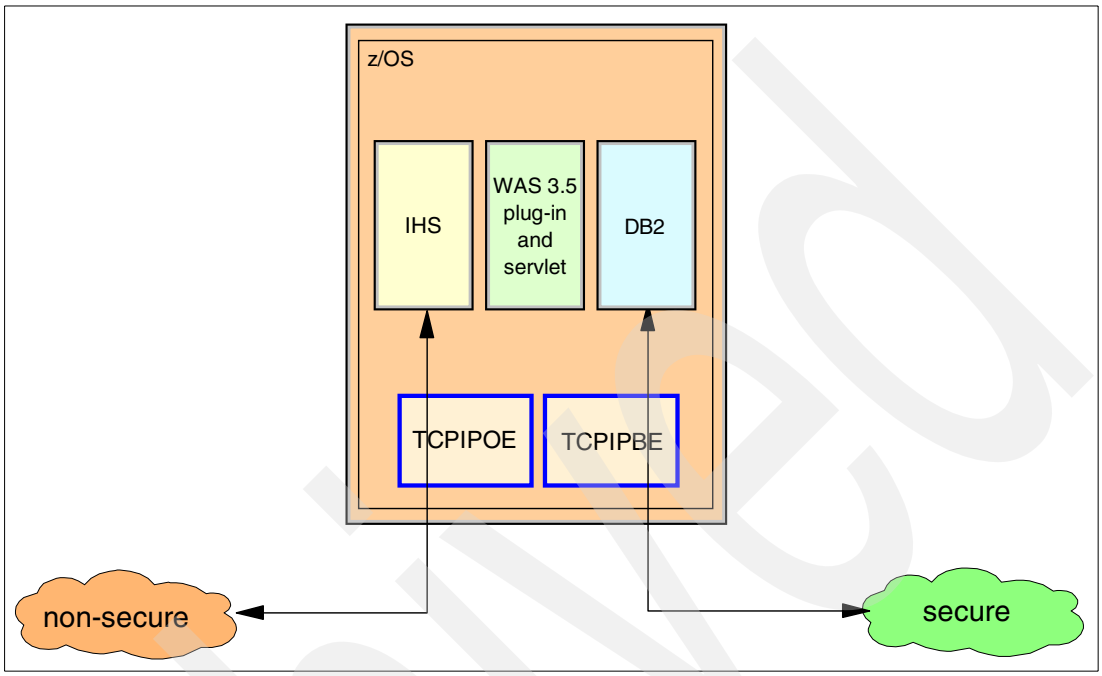


Figure 6-18 Our setup

Example 6-3 TCP/IP definitions in BPXPRMxx

```
/*******************************************************/
/*  Default TCPIP procedure                            */
/*******************************************************/

   SUBFILESYSTYPE NAME(TCPIPOE)
                  TYPE(CINET)
                  ENTRYPOINT(EZBPFINI)
                  DEFAULT
   SUBFILESYSTYPE NAME(TCPIPBE)
                  TYPE(CINET)
```

Intrusion Detection Services

Intrusion Detection Services (z/OS V1R2 and above) can be activated in the outer stack as an additional protection against hostile inbound TCP/IP flow.

Using SAF stack access control

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

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

where:

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

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

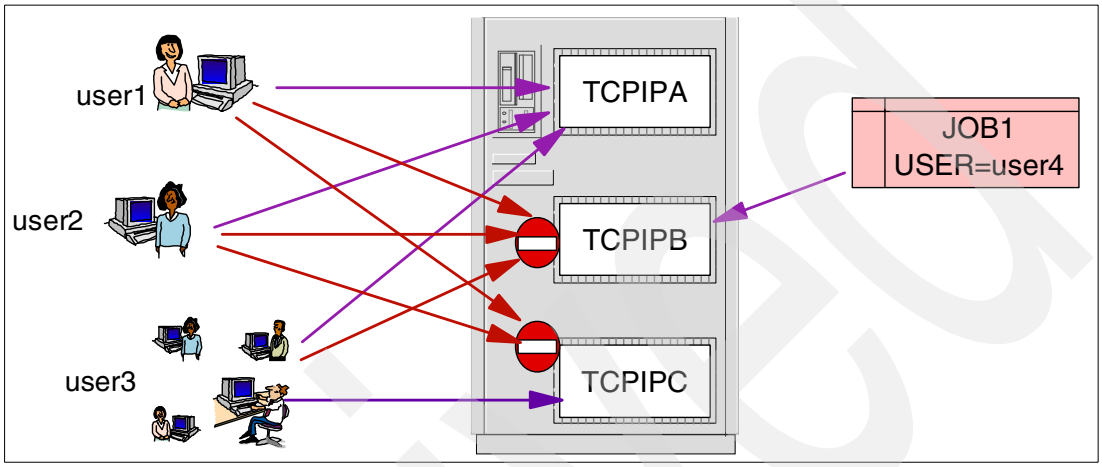


Figure 6-19 Stack access control

Figure 6-19 is an example of stack access control in action. User IDs that are not explicitly SAF authorized will be unable to access a stack. Although this example shows end users, the principle applies equally to user IDs associated with any TCP/IP application.

We have experimented with this facility in the lab, with the RACF profiles defined as in Figure 6-4. The idea here is to strictly control which applications can bind to the inner TCP/IP stack.

Example 6-4 RACF profiles for stack access control

```
SETR CLASSACT(SERVAUTH)
RDEF SERVAUTH EZB.STACKACCESS.POS1.TCPIPBE UACC(NONE)                  00076624
RDEF SERVAUTH EZB.PORTACCESS.POS1.TCPIPBE.SAFDRDA UACC(NONE)           00076624
PE EZB.STACKACCESS.POS1.TCPIPBE CLASS(SERVAUTH) ACC(READ)  ID(OEKERN)  00076624
PE EZB.STACKACCESS.POS1.TCPIPBE CLASS(SERVAUTH) ACC(READ)  ID(ROOT)    00076624
PE EZB.STACKACCESS.POS1.TCPIPBE CLASS(SERVAUTH) ACC(READ)  ID(CSUSER)  00076624
PE EZB.STACKACCESS.POS1.TCPIPBE CLASS(SERVAUTH) ACC(READ)  ID(DBDBOP)  00076624
```

Regulating the traffic flowing through the inner stack

As an additional protection against denial of service type attacks, one might consider regulating the secure network inbound traffic delivered by the inner stack. This can be achieved using a Quality of Service policy in the inner stack.

Note: This does *not* replace the traffic regulation management facility that we have in the Intrusion Detection Services since IDS is not applicable to outbound traffic.

6.7.2 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 6-20, where the connected member constitutes the DMZ enclosed by a dedicated outer firewall box and, for instance, the z/OS firewall technologies on the inner side.

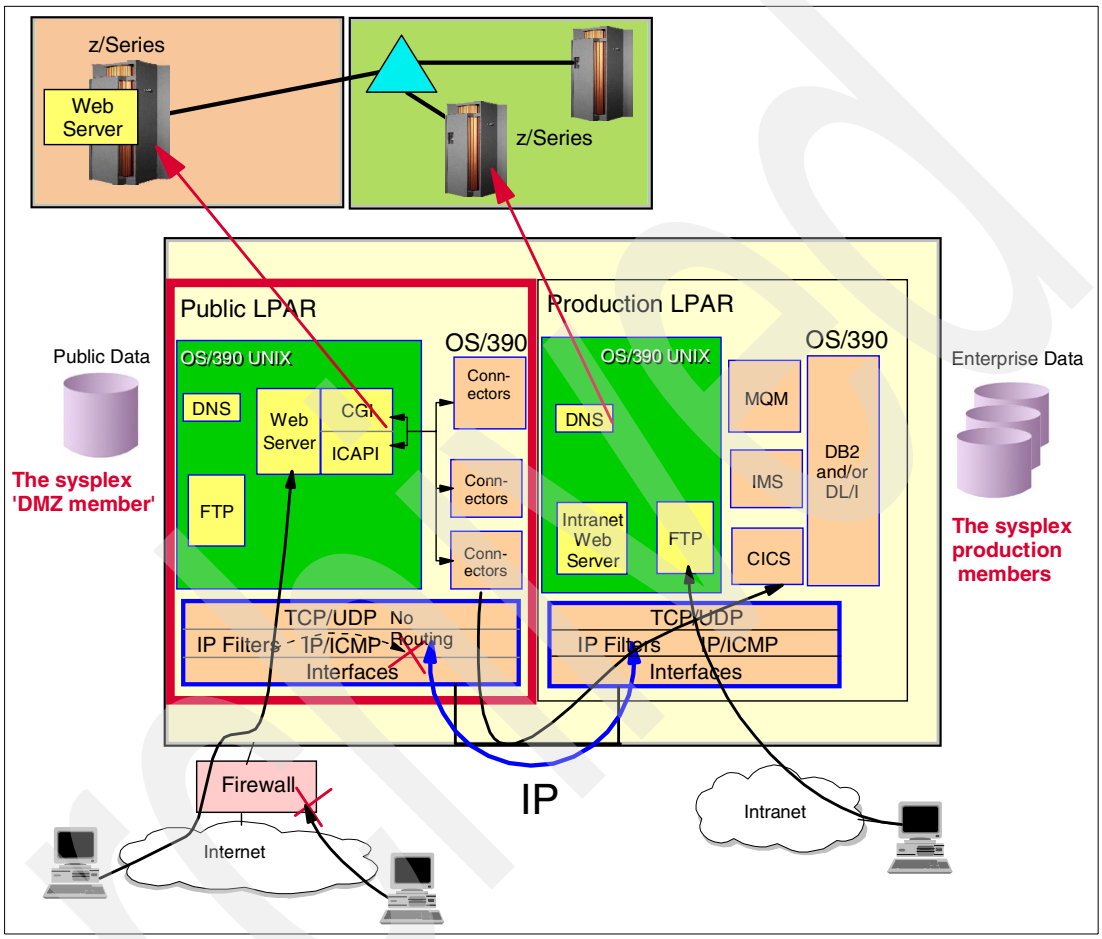


Figure 6-20 Applying the DMZ principle to the sysplex

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 which 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 6-21. For this illustration we assume that the benefits of sysplex are required for the services provided from the DMZ.

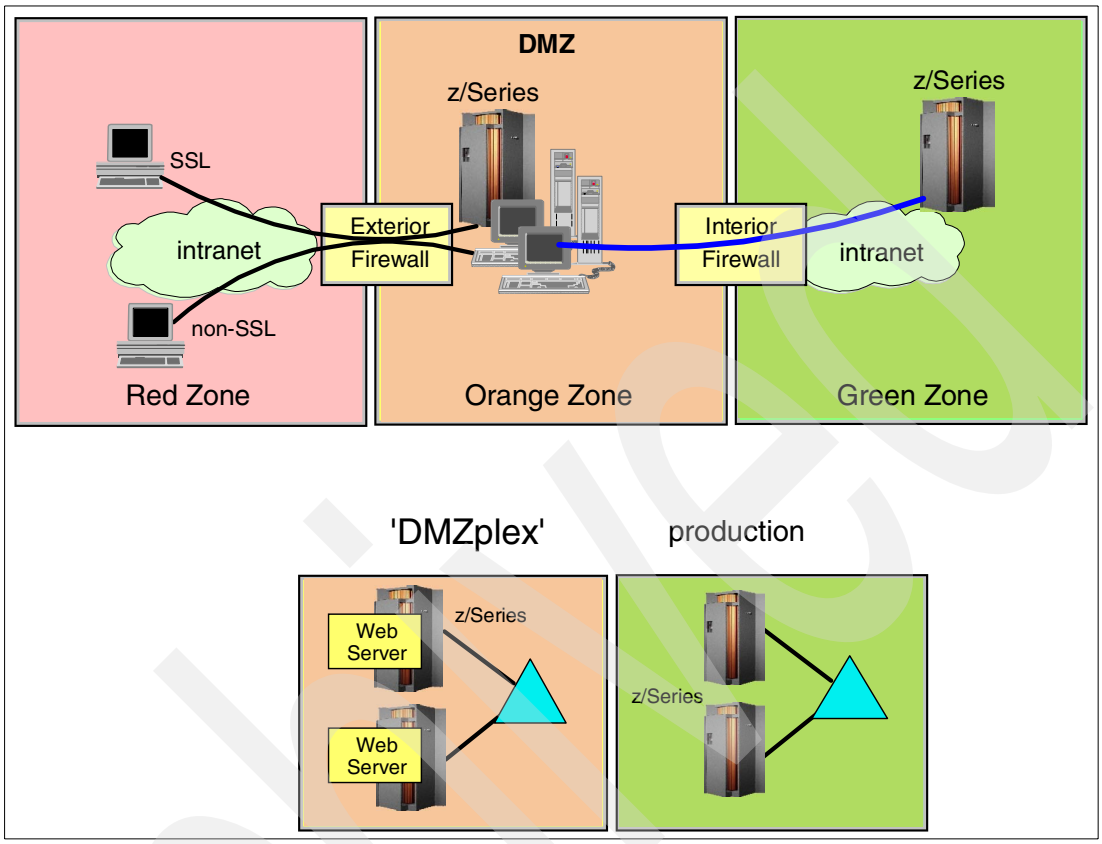


Figure 6-21 The DMZ-plex

Words of caution - RACF user revocation

Another issue when connecting to the Internet and when authenticating users by RACF userid 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.”

Words of caution - Sysplex Distributor

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 firewall technologies.
- 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 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 6-22, 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 dynamic switches such as the Cisco MNLB as shown in Figure 6-23.

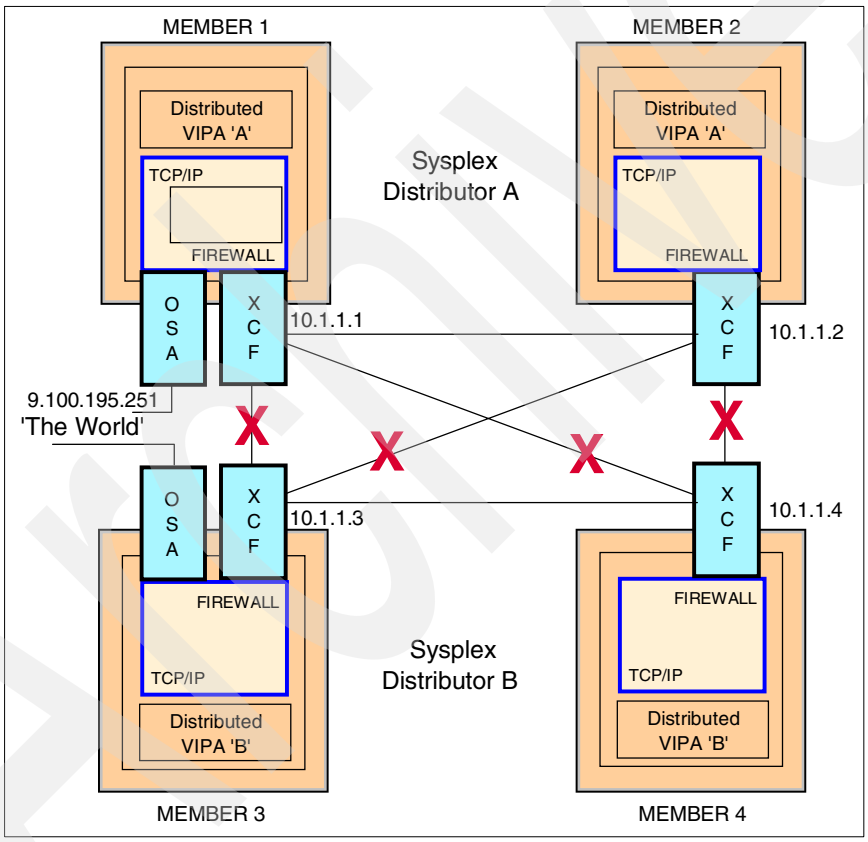


Figure 6-22 Closing IP routes created by DYNAMICXCF

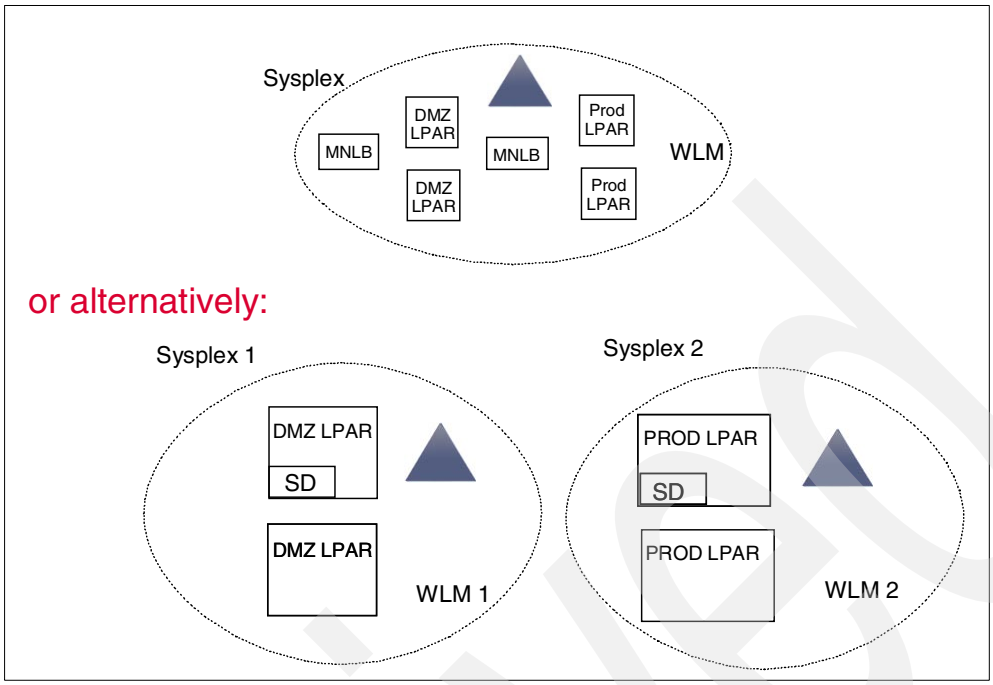


Figure 6-23 More than one Sysplex Distributor case

6.7.3 The shared HFS case

The sysplex shared HFS facility provided beginning with OS/390 V2R9 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. In the initial implementation at OS/390 V2R9:

- The support being introduced provides the ability for all mounted file systems to be visible from all systems 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 system that has left the sysplex for any reason. Those file systems will be randomly moved to surviving systems in the sysplex, providing enhanced file system availability.

There needs to be one OMVS couple data set (CDS) 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 systems 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 CDS, 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 CDS for changes. Those changes are then processed on the other systems, allowing them to maintain a common file system hierarchy.

Figure 6-24 shows the principle of operation of HFS sharing before and after OS/390 R9.

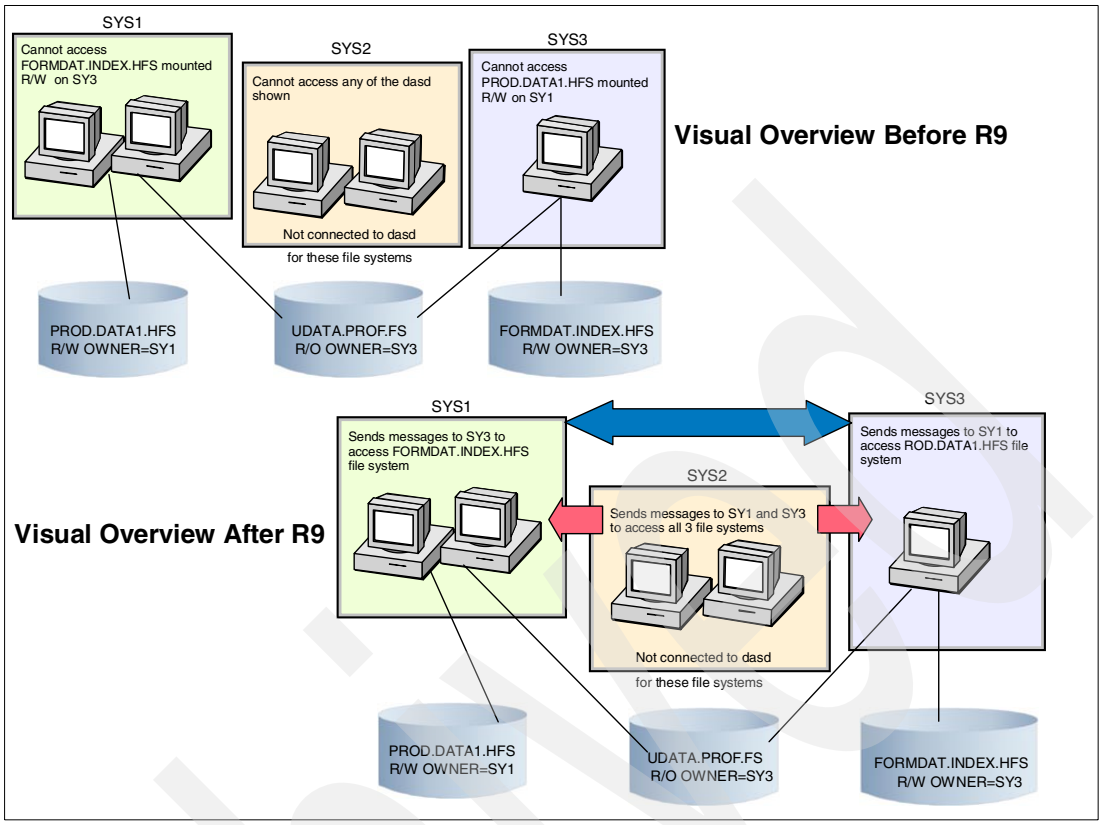


Figure 6-24 HFS sharing before and after OS/390 V2R9

The Byte Range Locking Manager (BRLM)

Byte Range Locking is used via fcntl or v_lockctl interfaces, and provides a means to serialize all or part of a file for reading or writing. This is advisory locking, and only works if all parties participate in using it. Prior to OS/390 R9, each system had its own Byte Range Lock Manager. In the initial implementation of OS/390 R9, however, to provide deadlock detection across the sysplex, there can be only one BRLM. The first system to enter the sysplex in sharing mode will initialize the BRLM for the sysplex. All systems requesting locks will communicate with that system via XCF messaging services. The Byte Range Lock Manager cannot be moved; however, if the owning system fails, then another system in the sysplex will initialize a new Byte Range Lock Manager. In this case, all locks held by the failed system are lost, and are non-recoverable, and all files which were using Byte Range Locking will have to be closed before they can be used again. Once closed and opened again, the file can again be used for locking, reading, and writing.

Loss of the BRLM on one system can therefore cause application failures on any of the other systems involved in the sharing group. This caused some customers to be unable to run some applications in the shared HFS environment. To address this problem, support for a new function called *Distributed BRLM* is implemented in z/OS V1R4 and rolled back into the service stream back to OS/390 R9. This solution provides one lock manager for each system in the sharing group to service file systems owned by that system. The new function has some restrictions for moving file systems, but when successfully moved to a new owner, the file system will be serviced by the BRLM at the new owner.

Security considerations with shared HFS implementation

Security considerations with sysplex HFS revolve mainly around these two concerns when one sysplex member is “exposed” to the Internet:

- Visibility of the common file system hierarchy and files by all contributing members. This is, of course, 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 at z/OS V2R3 or above). If it is an unacceptable security exposure to share a common file system hierarchy with the exposed member, then there are several alternatives:
  - IPL the exposed member with SYSPLEX(NO) in BPXPRMxx, meaning this member can not participate in HFS data sharing.
  - Use a configuration where the Internet system is not part of the secure sysplex (it can be part of another sysplex, though, such as a “DMZplex”).
- Ownership of file system. One may want to restrict file system ownership given to the Internet member, so that a compromise of the Internet 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.

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 it's various command-dependent forms) is AUTOMOVE (as opposed to NOAUTOMOVE), to recover the file system to a new owning system.

This introduces another security concern in that you cannot, prior to z/OS V2R4, control *which* member is getting the automoved file system ownership, and therefore the exposed member is a potential candidate if it has connectivity to the file system to be moved.

At z/OS V2R4, MOUNT now allows either an include or exclude system list on the AUTOMOVE parameter to allow specification of which systems will take over ownership of a file system when the original server system is brought down. The display commands that display file system information now display AUTOMOVE system list information. Commands to change the AUTOMOVE option will allow specifying a system list. A new operand is added to automove in the `mount parmlib` statement and TSO MOUNT command:

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

where *indicator* is either INCLUDE or EXCLUDE, which can also be abbreviated as I or E.

The mount shell command is also changed:

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

Note that the panel on ISHELL for mount allows selection to set the automove attribute and, if selected, provides a new panel to choose automove type and specify a list of up to 32 systems.

Mixed-release sysplex with pre-z/OS R4 systems

AUTOMOVE system list is used during dead system takeover processing; each system runs dead system recovery and attempts to take over automoveable unowned file systems. In z/OS R4 and above, systems honor the system list (include or exclude) if it was specified for that file system. Any pre-z/OS R4 systems will be unaware of the system list and may take over a file system regardless of the presence of a syslist.

> **Note:** There is no way to move BRLM deliberately, and when the system that owns BRLM leaves the sharing group, BRLM is automatically recovered by the first system to perform dead system recovery. In z/OS V2R4 and above with AUTOMOVE with a system list, that would be an included system. Since moving BRLM is not within the administrator's control, there is no message indicating where BRLM was moved, although that can be determined using `D OMVS,O`.

6.7.4 The sysplex and Denial of Services attack

Single packet DoS

The intrinsic robustness of the z/OS TCP/IP stack, together with the IDS being activated to report attempts at such an attack, are expected to provide proper security to the sysplex configuration.

Multiple packet DoS

There are some questions regarding what would happen if the “Internet member” of a sysplex is subject to a multiple packet DoS:

- Can it stall the member, and thus is there an exposure to miss a sysplex status update that would end up in varying offline the Internet member?
- What would happen if the member under attack is holding sysplex-wide locks?

We have tried to assess the effectiveness of WLM in constraining the impact of such an attack to the TCP/IP address space. Although we did not conduct an exhaustive test, the results left us quite confident of the capability of keeping to a minimum the effects of the stack overload to the other address spaces in the system. Furthermore, as explained in the next chapter, IDS provides additional protection regarding traffic regulation.

WLM classification

Here are some examples of WLM classification that we found to be adequate during our testing.

z/OS is able to manage many different workloads, each with distinct business importance and processing characteristics. In order to accomplish this, the installation must categorize the arriving transactions to z/OS.

Classification rules are rules WLM uses to associate a transaction’s external properties (work qualifiers like LU name or user ID) with a transaction goal. These goals are located in a service class definition.

Classification rules and service classes are both defined in a service definition. A service definition also contains one or more service policies. There is just one active policy at a time in the sysplex.

Example 6-5 WLM Policy

```
Service-Policy  View  Notes  Options  Help
------------------------------------------------------------------------------
                      Service Policy Selection List         Row 1 to 2 of 2
Command ===> ____________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              7=Override Service Classes, 8=Override Resource Groups,
              /=Menu Bar
                                                   ----Last Change-----
Action  Name      Description                      User      Date
  __    POPOLW    Policy for weekend               ALAIN     2001/12/12
  __    POPOL1    Policy setup z/OS 2.2            ALAIN     2001/11/16
```

Note: Since OS/390, you can give different goals to the same workload based on the system on which its executes.

6.7.5 TCP/IP classification

TCP/IP is classified under subsystem type STC. If you do not classify TCP/IP it is, by default, associated with SYSSTC service class. If you need it for monitoring purposes you can associate a report service class with TCP/IP still using SYSSTC as service class.

Example 6-6 TCP/IP stack classification example

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type    Row 39 to 46 of 51
Command ===> ___________________________________________ SCROLL ===> CSR

Subsystem Type . : STC         Fold qualifier names?   Y  (Y or N)
Description  . . . system STC  & Users STC

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                          More ===>
          --------Qualifier--------               -------Class--------
Action    Type       Name     Start                Service     Report
                                          DEFAULTS: VEL3OSTC    VEL3OSTC
 ____  1  TN         TCPIP*   ___                  ________    RTCPIP
 ____  1  TN         TNF      ___                  SYSTEM      STC
```

Note: SYSSTC is an internal service class which has DP=253 and IOP=254. This is where important, light CPU address spaces (VTAM, JES2) should run. It is also the default service class for started tasks, unless you provide another default in SUBSYS=STC.

It is a good practice to define this service class with a low importance so that any unclassified STC will not cause performance issues.

TCP/IP classification WEB member consideration

You may want to associate a specific service class with TCP/IP on the WEB LPAR to be able to limit the resource consumed by the TCP/IP started task in case of a denial of service attack. In order to do that you can use the system as qualifier type.

Example 6-7 Shows TCP/IP stack configuration

```
Subsystem-Type  Xref  Notes  Options  Help
 ------------------------------------------------------------------------------
                  Modify Rules for the Subsystem Type    Row 39 to 46 of 52
 Command ===> ___________________________________________ SCROLL ===> CSR

 Subsystem Type . : STC         Fold qualifier names?   Y  (Y or N)
 Description  . . . system STC  & Users STC

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
           --------Qualifier--------                -------Class--------
 Action    Type       Name     Start                Service     Report
                                           DEFAULTS: VEL3OSTC    VEL3OSTC
  ____  1  TN         TCPIP*   ___                  _______     RTCPIP
  ____  2    SY         POS2     ___                TCPIPOE     RCTPIPOE
```

6.7.6 TCP/IP server classification

TCP/IP servers like FTP or TELNET are UNIX processes and are classified under subsystem type OMVS.

INETD

INED is classified under subsystem type STC or OMVS depending on the way you start it. In both cases you can classify job name or user ID.

Example 6-8 Shows INED setup

```
Subsystem-Type  Xref  Notes  Options  Help
 ------------------------------------------------------------------------------
                  Modify Rules for the Subsystem Type    Row 15 to 22 of 52
Command ===> ___________________________________________ SCROLL ===> CSR

Subsystem Type . : STC         Fold qualifier names?   Y  (Y or N)
Description  . . . system STC  & Users STC

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                           DEFAULTS: VEL3OSTC    VEL3OSTC
 ____  1  TN         INETD*   ___                  INETD       _______
```

FTP server

FTP daemon (listener) is classified under subsystem type OMVS, and you can use job name or user ID, for example, to classify this server.

The daemon forks a new process for each demand of logon. In a first step the process has the same characteristics as the daemon (user ID and job name prefix) and is classified the same way as the daemon. In a second step, after the user has specified its user ID and password, the process gets the characteristics of the user and is classified again.

Example 6-9 Shows FTP setup

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type       Row 2 to 7 of 7
Command ===> ___________________________________________ SCROLL ===> CSR

Subsystem Type . : OMVS        Fold qualifier names?   Y  (Y or N)
Description  . . . OMVS SETUP

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                            More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                           DEFAULTS: VEL50_03    OE_DFLT
 ____  1  TN         FTP*     ___                  FTP         RFTP
 ____  1  UI         ALAIN    ___                  USSFORK     USSFORK
```

TELNET server

The INETD daemon forks a new process for each new connection. In a first step the process has the same characteristics (user ID and job name prefix) you specified in INETD.conf and it is classified according to these characteristics. In a second step, after the user has specified its user ID and password, the process gets the characteristics of the user and is classified again.

Example 6-10 otelnet command

```
otelnet stream tcp nowait OMVSUSR /usr/lpp/tcpip/sbin/otelnetd otelnetd -h -m -n -c 180
```

Example 6-11 otelnet command output

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type      Row 2 to 8 of 10
Command ===> ___________________________________________ SCROLL ===> CSR

Subsystem Type . : OMVS        Fold qualifier names?   Y  (Y or N)
Description  . . . OMVS SETUP

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                            More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                           DEFAULTS: VEL50_03    OE_DFLT
 ____  1  TN         OMVSUSR  ___                  VEL50_02    ________
 ____  1  TN         ALAIN    ___                  VEL50_01    ________
```

Note: To limit the resources consumed for abnormal connection attempts, it is important to specify an inactivity timeout value for the negotiation pass during negotiations with the client -c option. You can also disable TCP keep-alive packets -n option.

7

Intrusion detection services

This chapter presents some basic concepts related to intrusion detection, and then provides the details about z/OS Intrusion Detection Service and its components.

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 analyses 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 in the host’s TCP/IP stacks.

7.1.1 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 some 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.

7.1.2 Host-based intrusion detection

Host-based intrusion detection relies on additional capabilities in 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:

- 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.

7.2 The z/OS Intrusion Detection Services

z/OS V1R2 introduces 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. It builds upon new capabilities of the Traffic Regulation Management Daemon (TRMD) available in OS/390 V2 R10. Traffic regulation is provided for TCP and UDP traffic. 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 Denial-of-Service attacks
- Gather data for possible legal actions

The z/OS IDS layout in an installation is shown in Figure 7-1. 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.

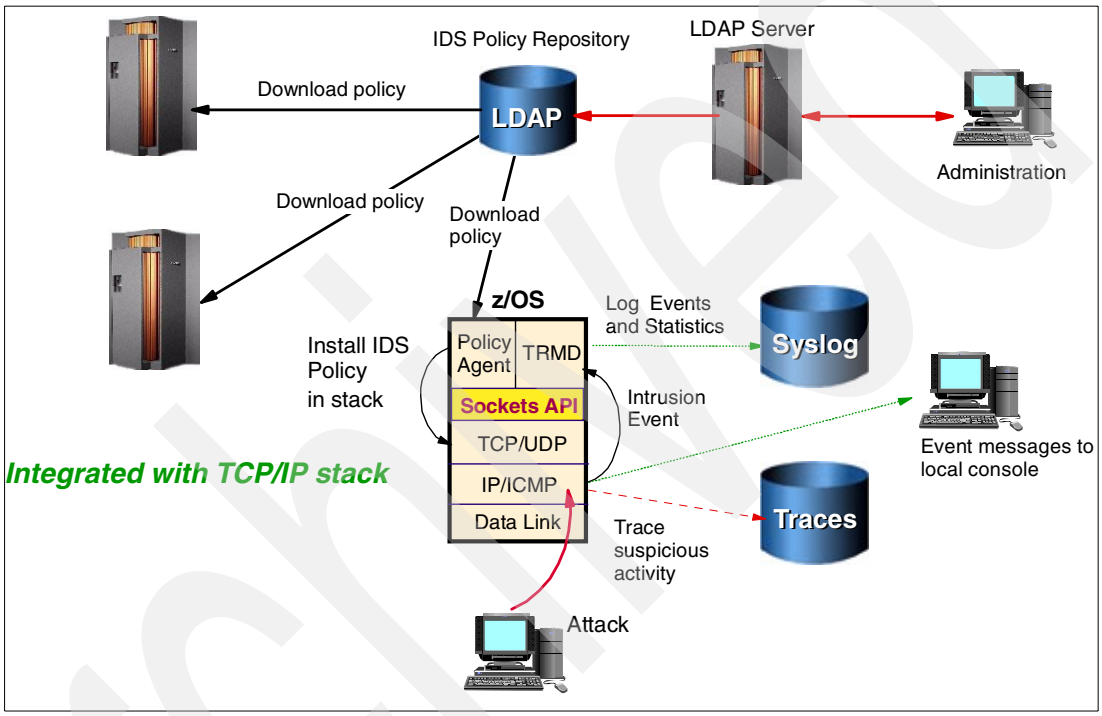


Figure 7-1 Overview of the z/OS IDS infrastructure

7.2.1 Policy-based networking

Businesses typically define goals for network behavior in human terms, for example using service level agreements (SLAs) or defining the behavior of intrusion detection services (IDS). Network implementations provide a wide variety of controls for priority treatment of traffic, bandwidth management, and control of network behavior. The link between the high-level business goals and network implementations is defined as *policy-based networking*. The implementation of SLAs, IDSs and other network controls on network hosts and routers is provided by policies. Policies are an administrative means to define controls for a network, in order to achieve the quality of service (QoS) levels promised by SLAs, or to implement IDS or resource-balancing decisions.

Network administrators can use the z/OS CS Policy Agent to define policies for their users.

The policies supported by the policy agent can be used to specify a quality of service or to define intrusion detection services. As of the writing of this book, the policies categories supported by CS for z/OS are:

- DS (Differentiated Services)
- RSVP (Resource Reservation Protocol)
- IDS (which integrates the Traffic Regulation (TR) policy previously available with OS/390 V2 R10)
- SD (Sysplex Distributor)

Note: These policies are defined by the policy agent based on the information found in the LDAP directory, but they are enforced by the IP stack.

Figure 7-2 shows the policy component in CS for z/OS.

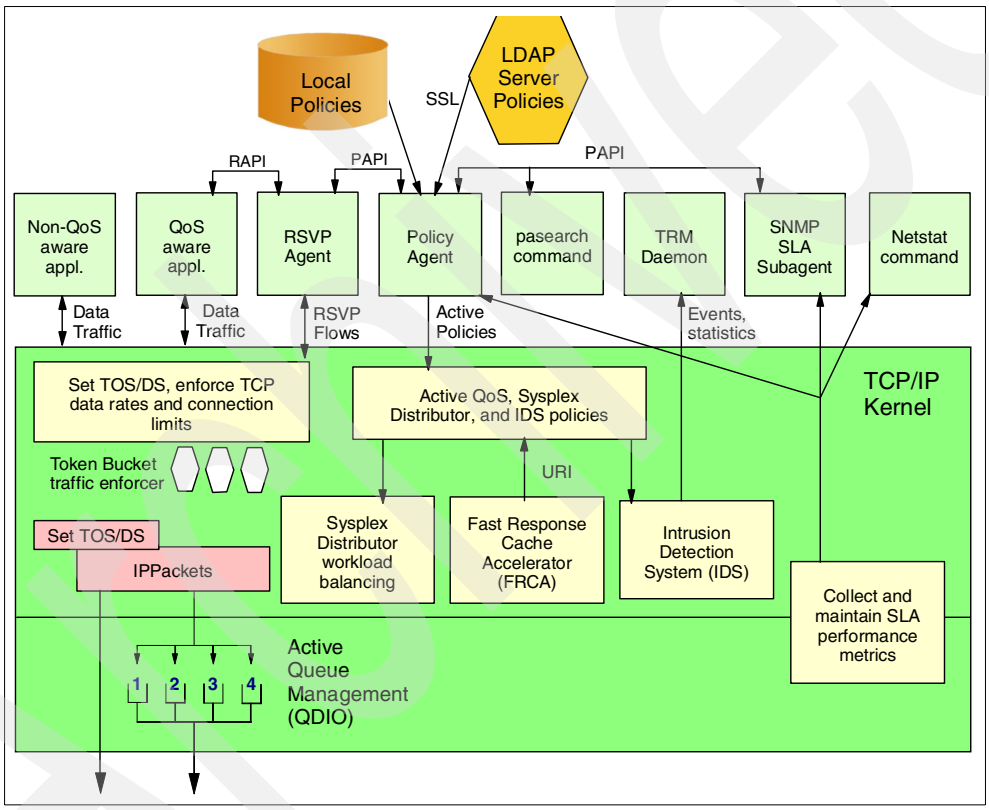


Figure 7-2 Policy component in z/OS CS

7.2.2 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 on 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 software, 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's 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 7-3 shows an overview of the interactions between the z/OS IDS components.

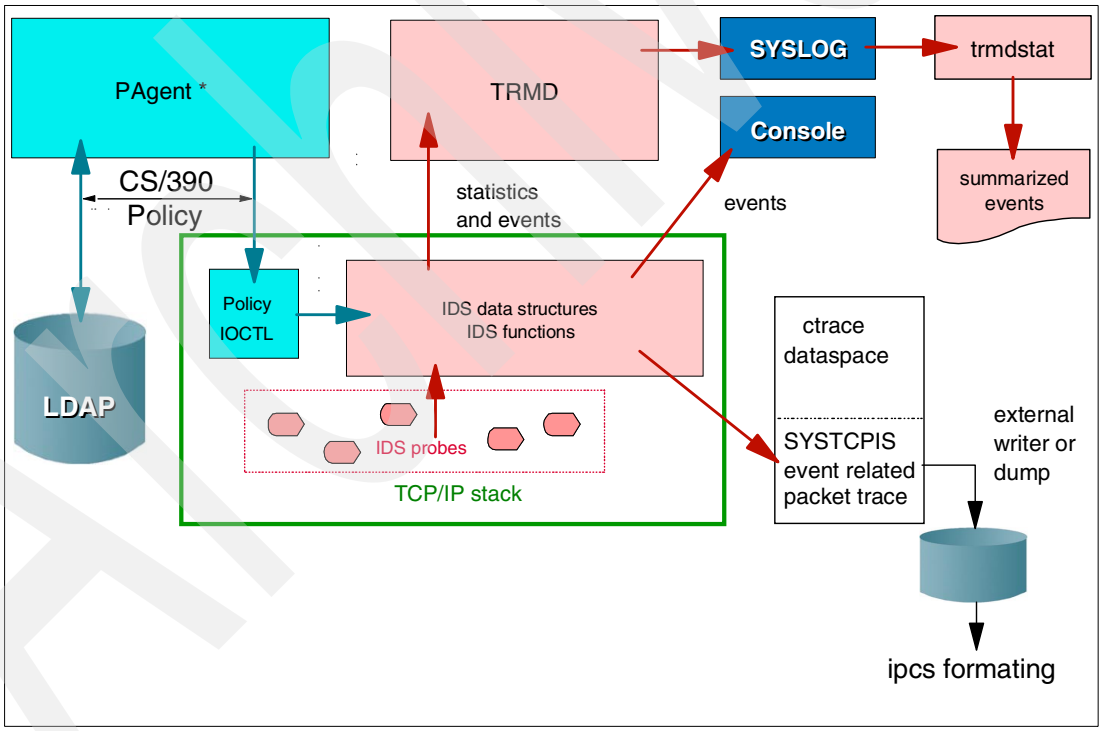


Figure 7-3 Intrusion Detection Services and IP stack Overview

Notes: There is only a single policy agent per z/OS image.
There is one TRMD instance per TCP/IP stack using IDS in the z/OS image.

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 and/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* used to tag the finer detailed information, such as traces, that go along with the event. An example of what meaning is given to “IDS countable event” is shown in the Scan Policy explanation.

Example 7-1 shows messages issued by TRMD and displayed at the MVS operator console; Example 7-2 shows messages received and stored by SyslogD.

Example 7-1 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 7-2 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,correlat
or=3,probeid=01004400,threshold=0
```

7.3 Preparing to run IDS

Important: Complete details on configuration and setup of the facilities described in this section are found in *z/OS Communications Server IP Configuration Reference*, SC31-8776, and *z/OS Communications Server IP Configuration Guide*, SC31-8775.

7.3.1 The z/OS Policy Agent (Pagent)

The Pagent is a UNIX application that can be started from the z/OS shell, as a background process with the “&” ending character in the command, or it can be initiated as a started task. There is one Pagent per z/OS image that can download, install, and maintain policies for one or more TCP/IP stacks.

Notes: Here, policies comprise not only the IDS policy but also the QoS policy. We started the Policy Agent using the PROC shown in Example 7-3 (a sample PROC is found in hlq.SEZAINST(EZAPAGSP))

The Pagent needs a configuration file. We used the default: /etc/pagent.conf.

Example 7-3 PAGENT started task

```
********************************** Top of Data ***************************
//PAGENT   PROC
//*
//* IBM Communications Server for OS/390
//* SMP/E distribution name: EZAPAGSP
//*
//* 5647-A01 (C) Copyright IBM Corp. 1998, 2000
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV2R10
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /
//             etc/pagent.conf'
```

The pagent.conf file contains general configuration information such as the URL of the LDAP directory where the policy resides. You can install the same policies for all TCP/IP stacks, or a unique policy for each TCP/IP stack that you may have in the z/OS image. If there are several TCP/IP stacks with each one of them having its own unique policy, you have to create additional Pagent configuration files (secondary, or “image,” configuration files, one for each stack receiving a unique policy) and to associate the TCP/IP stack and the secondary configuration file with a TcpImage statement. In the example of Pagent configuration in Example 7-4, there is only one stack and one policy. The configuration file contains a TcpImage statement pointing at the stack jobname as entered in TCP Data, and indicates with the FLUSH keyword which specifies that all policies installed in the policy agent and the TCP/IP stack are deleted on the initial startup of Pagent. This single Tcpimage statement, not pointing at a specific Pagent configuration file indicates that the configuration information will follow.

Note: A new keyword is available for the Tcpimage statement:

PURGE: Policies are deleted from the stack when terminating the Pagent.

NOPURGE (the default): Policies are not deleted from the stack at Pagent termination.

Example 7-4 Policy Agent configuration file

```
LogLevel 511
# TcpImage Statement
#   This statement specifies an MVS TCP/IP image/stack and its associated
#   policy control file to be installed to that image.
TcpImage TCPIPOE PURGE FLUSH 600
#
ReadFromDirectory
               {
                 LDAP_Server             9.100.195.251
                 LDAP_Port               3389
                 LDAP_DistinguishedName  cn=LDAPadm
                 LDAP_Password           secret
                 LDAP_ProtocolVersion    3
                 LDAP_SchemaVersion      3
                 SearchPolicyBaseDN cn=TCPIPBE, cn=groups, ou=policy, o=IBMRES
                 SearchPolicyKeyword     POLICY
               }
#
PolicyPerfMonitorForSDR    Enable
               {
                  Samplinginterval          120
                  LossRatioAndWeightFr    10 20
                  LossMaxWeightFr          100
                  TimeoutRatioAndWeightFr  5 50
                  TimeoutMaxWeightFr        100
               }
```

For the policy information to be retrieved from the LDAP server, you have to provide information about the LDAP server URL and the LDAP administrator name and password that can be used to get access to the directory objects. The directory tree is searched starting with the object with the distinguished name specified by SearchPolicyBaseDN. A refresh interval can be specified in the Tcpimage statement to indicate if and when the installed policies can be automatically refreshed from the LDAP directory. In our example they are refreshed every 10 minutes.

If you were to have several stacks in the z/OS image to be served by the pagent instance (let's assume stack TCPIPOE and stack TCPIPBE), the pagent primary configuration file, shown in Example 7-5, will be used to direct Pagent to the appropriate secondary (image) configuration file pertaining to each stack. Each image configuration file contains the statements shown in Example 7-4.

Example 7-5 Pagent configuration file for two TCP/IP stacks

```
LogLevel 511
# TcpImage Statement
#   This statement specifies an MVS TCP/IP image/stack and its associated
#   policy control file to be installed to that image.
TcpImage TCPIPOE /etc/TCPIPOE/pagent.conf PURGE FLUSH 600
TcpImage TCPIPBE /etc/TCPIPBE/pagent.conf PURGE FLUSH 600
```

Note: The communication between the Pagent LDAP client and the Policy LDAP server can be secured using SSL. Other statements are available for the Pagent configuration file to specify SSL parameters.

A sample of the pagent.conf file can be found at /usr/lpp/tcpip/samples/IBM/EZAPAGCO.

7.3.2 TRMD

The Traffic Regulation Management Daemon (TRMD) has been upgraded from the OS/390 V2 R10 TRMD to manage 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 that can be set in different ways. In our case, using the JCL below, we indicate the TRMD environment variable file is specified in the STDENV DD card.

TRMD runs as an authorized program and requires some RACF setup (TRMD must be able to run as a started task and have superuser authority). See the EZARACF member of SEZAINST for sample RACF commands.

Example 7-6 The TRMD Started Procedure

```
//TRMD      PROC
//TRMD   EXEC PGM=EZATRMD,REGION=4096K,TIME=NOLIMIT,
//   PARM=('POSIX(ON) ALL31(ON)','ENVAR("_CEE_ENVFILE=DD:STDENV")/')
//STDENV   DD PATH='/u/res/trmd/trmd.env',PATHOPTS=(ORDONLY)
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSIN    DD DUMMY
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

The environment variables file, /u/res/trmd/trmd.env/ is shown in Example 7-7.

Example 7-7 TRMD Environment Variables

```
LIBPATH=/usr/lib:
RESOLVER_CONFIG=//'TCPIP.TCPPARMS(TCPDATOE)'
```

If you were to use several stacks, the key point to look at is to have the TRMD instances operating with the right stack. TRMD uses the value of the RESOLVER_CONFIG environment variable to find out the name of the stack to attach to. (“Name” here means the value of TCPIPJOBNAME in TCP.DATA.) In z/OS V1 R2 consideration must also be given to the resolver address space, started by default, and the new GLOBALTCPIPDATA keyword. Assuming that we have two instances of TRMD (TRMD to serve stack TCPIPOE and TRMDBE to serve stack TCPIPBE), care must be taken not to give a TCPIPJOBNAME in the GLOBALTCPIPDATA data set. Doing so results in having all the running instances of TRMD attach to this very same stack. Figure 7-4 summarizes what should be done. In that case each TRMD PROC points to its own environment variable file, which in turn points to the TCP.DATA where the TCPIPJOBNAME is. When using a GLOBALTCPIPDATA data set, there must not be a TCPIPJOBNAME in the TCP.DATA member specified, here “TCPDATP0.” If a TCPIPJOBNAME were specified in TCPDATAP0 then it would override both stacks’ environment variables with this single value.

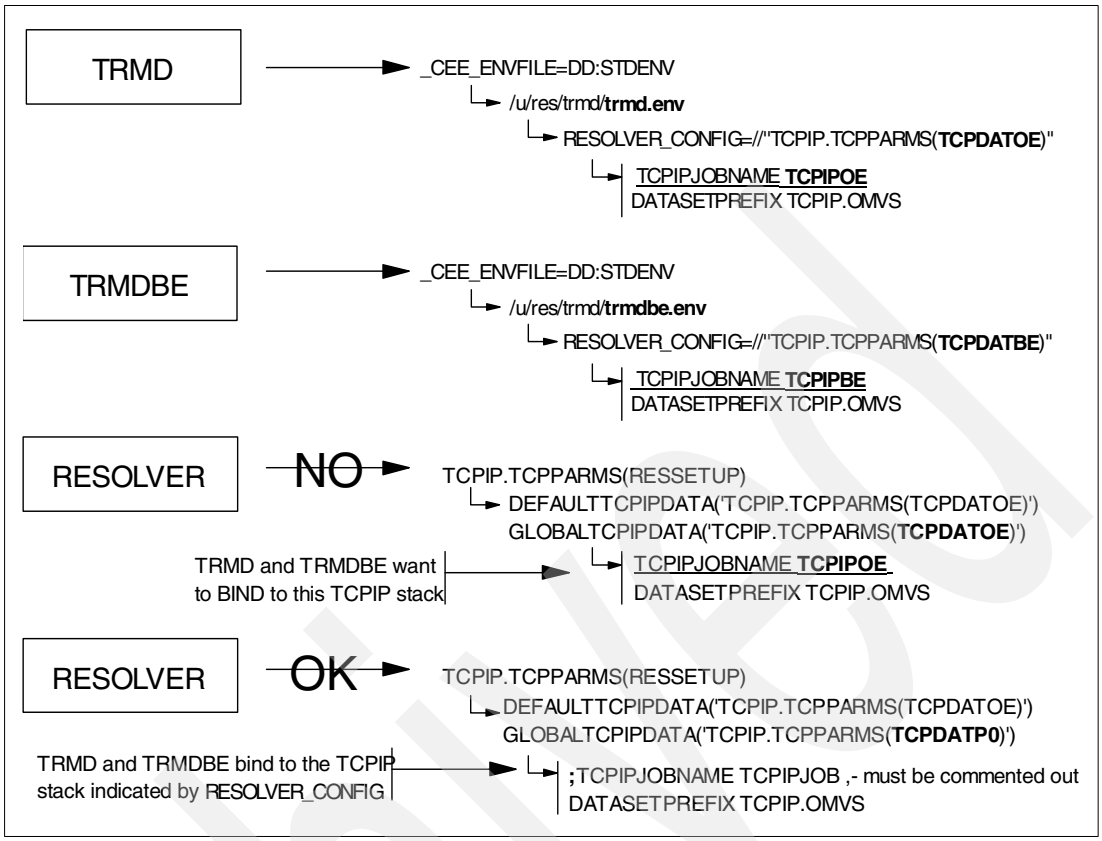


Figure 7-4 Specification of the environment variables file for individual TRMD instances.

7.3.3 SyslogD configuration

The SyslogD UNIX application is used to collect the messages issued by TRM and, depending on the directives setup in its configuration file, it reads and log's system messages to the MVS console, log files, SMF, other machines, or users.

To capture intrusion information, you must start TRMD and SyslogD.

> **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 by and to remote servers. If it is not necessary that SyslogD receive messages from remote servers, which lies as a security exposure, the SyslogD start command must be issued with the -i parameter, as explained in the following section.

Syslog facility names are specified in /etc/syslog.conf with associated log files which are stored in the Hierarchical File System (HFS). Depending on how the application has been coded, debug messages may go to one file and trace messages to another. The documentation for each server should indicate where the log debug messages, error messages, information messages, and warning messages should be written. Some will log all of them to the same syslogd destination.

The SyslogD has predefined "facility" names, which can be used for making decisions on 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.

In the case of the IDS, messages issued by 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 *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383 for more information.

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 7-8, which tells SyslogD that all messages issued by the user FWKERN, whatever the jobname, 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 7-8 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 7-9.

Example 7-9 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 (available starting with CS for OS/390 V2R10 IP) 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 userid and jobname in the record.

Using these options permits you to isolate messages into separate logs and to restrict communication with remote syslog servers more easily. This is given the generic term of "SyslogD Isolation."

7.4 IDS policy definition and installation

Attention: We are assuming here that the reader is somewhat familiar with the LDAP concepts. Should more information be needed, the following resources are useful:

- IBM Redbooks
  - *Understanding LDAP*, SG24-4986
  - *OS/390 Security Server*, Ready for e-business, SG24-5158
  - *LDAP Implementation Cookbook*, SG24-5110
  - *OS/390 Security Server 1999 Updates, Implementation Guide*, SG24-4629
  - *Putting the Latest z/OS Security to Work*, SG24-6540
- z/OS library
  - *z/OS Security Server LDAP Server Administration and Usage Guide*, SC24-5923
  - *z/OS LDAP Client Application Development Guide and Reference*, SC28-5924

The IDS policy is stored in an LDAP directory, such as one from the IBM SecureWay directory family, which includes the z/OS LDAP server as its TDBM backend. The z/OS TDBM backend actually allows you to install and maintain the directory objects in DB2 tables, which appear to the connected or local LDAP clients as entries in an LDAP Directory tree. The characteristics of the contents of an LDAP directory need to be described to the LDAP server; this is achieved by providing to the LDAP server "schema files" which contain information about LDAP object classes and attributes.

A z/OS IDS policy is actually stored in an LDAP directory and retrieved from it by the Policy Agent for installation inside a z/OS TCP/IPstack. Therefore, an IDS policy is built as a set of directory objects, or "entries," where keywords are represented as LDAP object attributes with a value given by the user setting up an IDS policy.

The initial installation of an IDS policy can be roughly described as a two-step process:

1. The installation of the IDS policy LDAP schemas.
2. The installation of the actual IDS policy, where the object classes and attributes used must be strictly the ones described in the schemas.

Note: The LDAP protocol and directory can be currently found at LDAP Version 2 or LDAP Version 3. The format of the schema information is sensitive to the LDAP version, and the z/OS IDS policy schema is only provided with an LDAP V3 format. The vehicle to provide the schemas is HFS and the files within HFS are called LDIF (LDAP Data Interchange File). These files are processed offline by the LDIF UNIX utility provided with the LDAP server (the LDIF2TDBM utility if the LDAP server runs on z/OS), or online by the LDAPMODIFY command.

Example 7-10 Example of ldapmodify command to install the IDS schemas

```
ldapmodify -h 127.0.0.1 -p 389 -D "cn=administrator" -w secret -f
/usr/lpp/tcpip/samples/pagent_idsschema.ldif
```

Historically, the Policy Agent that runs in the z/OS environment and reads policy definitions to install them in the TCP/IP stacks was initially reading the policy definitions from a local configuration flat file. This has been extended by giving the option to store the policies in an LDAP directory acting as a central repository serving multiple TCP/IP stacks. The z/OS IDS policies must be stored in an LDAP directory, with an exception for the Traffic Regulation policies built at a previous release (OS/390 V2 R10 or z/OS V1 R1) which, for compatibility reasons, can still be provided in flat files.

Note: The directory holding the IDS policies can be local to the system, an LDAP server and TDBM backend in the same z/OS image as the TCP/IP stack, or it can be a remote directory reached through the TCP/IP network. In the latter case the directory can be implemented on any platform providing LDAP V3 services.

The Policy Agent installs policies in one or more z/OS CS stacks. It can be used to replace existing policies or update them as necessary. Regarding overall policies support, z/OS V1R2 provides the following new enhancements above previous releases:

- The “core” schema required to define any policy (QoS or IDS) in an LDAP directory have been upgraded to LDAP Version 3 schema.
- A new schema is available to be used when defining IDS policies.
- The Policy Agent usability has been improved with:
  - New MODIFY command
  - New PURGE/NOPURGE parameter on the TCPIPIMAGE statement
- The PASEARCH command was enhanced.

These enhancement allow you to define the policy schemas using version 3 and refresh the policy dynamically using the MODIFY command. Details on Policy Agent and the pasearch command are in 7.5.3, “pasearch utility” on page 199.

7.4.1 The z/OS Communications Server policies schema

The schema versions

You may be using a Version 1, Version 2, or Version 3 LDAP schema to define policies. The PAGENT configuration file must indicate which schema should be retrieved in a statement called ReadFromDirectory. *Version 1* applies to policy definitions using the schema that was available prior to CS for OS/390 V2R10 IP, *Version 2* refers to the schema introduced with OS/390 V2R10 IP, and *Version 3* refers to the schema introduced with z/OS V1R2. (Version 1 remains valid with OS/390 V2R10 IP for migration and compatibility purposes; Versions 1 and

2 remain valid with z/OS V1R2.) Schema version 2 and version 3 objects must coexist on a single server and must be defined using a single schema definition.

The following schema file samples can be found in /usr/lpp/tcpip/samples:

- /usr/lpp/tcpip/samples/pagent_schema.ldif

  The schema version 2 core and QoS schema object class and attribute definitions
- /usr/lpp/tcpip/samples/pagent_v3schema.ldif

  The schema version 3 additions to the schema version 2 core and QoS schemas
- /usr/lpp/tcpip/samples/pagent_schema_updates.ldif

  Changes to the schema version 2 core and QoS schema definitions in support of schema version 3
- /usr/lpp/tcpip/samples/pagent_idsschema.ldif

  The schema version 3 IDS schema definitions

For LDAP protocol version 3, the schema definition is shipped in ldif format and installed on the LDAP server as a modification to the generic schema entry, known as a subschema. The existing schema entry must be modified to include the supported schema as a subschema, by using the `ldapmodify` command. Copy these files to another directory and make the appropriate changes *before* they are installed in the LDAP server via the ldapmodify command. Example 7-11 shows a sample of the pagent_schema.ldif file. The only thing that needs to be changed in this file is the suffix in the distinguished name of the entry (the line preceding `changetype: modify`) to reflect your system definitions. In our case we changed <suffix> to o=IBMRES, as o=IBMRES is declared as a suffix in our TDBM configuration file.

Example 7-11 pagent_schema.ldif file

```
dn:cn=schema, <suffix>
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.1801
  NAME 'ibm-idsActionType'
  DESC 'Specifies the type of IDS actions associated with a policy rule.
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.1801
  DBNAME( 'idsActionType'  'idsActionType' )
  ACCESS-CLASS normal
  LENGTH 32
  )
```

To install the policies, you can, as an example:

1. Copy the schema file or files:

   ```
   cp /usr/lpp/tcpip/samples/pagent_idsschema.ldif /home/myuser/idsschema.user.ldif
   ```

2. Edit the /home/myuser/idsschema.user.ldif file and replace
   `dn: cn-schema, <TDBM_suffix>`
   with, as an example (which implies that o=IBMRES is declared as a TDBM suffix in the LDAP server configuration file)
   `dn:cn=schema, o=IBMRES`

3. Run the **ldapmodify** command:

```
ldapmodify -h 127.0.0.1 -p 1800 -D "cn=admin" -w xxxxxx -f /home/myuser/idsschema.user.ldif
```

7.4.2 The IDS policy definition

Very helpful:

1. Detailed information on policies, their structures and attributes, is in *z/OS CS IP Configuration Guide*, SC31-8775, and in the very detailed comments at the beginning of the policy definition samples in the files identified in **7.4.7, "Policy Rules samples" on page 195**.
2. In the lab, we used an LDAP browser to maintain the policies in the LDAP directory. The LDAP browser provides a convenient graphical interface to easily access and maintain entries in the directory tree. We know of two LDAP browsers that can be downloaded from the Internet:
   - IBM's SecureWay Directory Management Tool

     This tool is part of the SDK for SecureWay Directory Server and can be obtained at:

     http://www.ibm.com/software/network/directory/downloads
   - The LDAP Browser/Editor by Jarek Gawor

     This tool is available for download from:

     http://www-unix.mcs.anl.gov/~gawor/ldap

     It is also commercially available through Argonne National Laboratory.

   Documented examples on the use of these tools can be found in *OS/390 Security Server 1999 Updates - Installation and Implementation Guide*, SG24-5629

7.4.3 Policy object model

Policies consist of several related directory objects. The main object is the policy rule. A policy rule object refers to one or more policy conditions, policy actions, or policy time period condition objects, and also contains information on how these objects are to be used. Policy time period objects are used to determine when a given policy rule is active. These objects are identified and located in the directory naming space using properly structured distinguished names, as shown in Figure 7-5 on page 176.

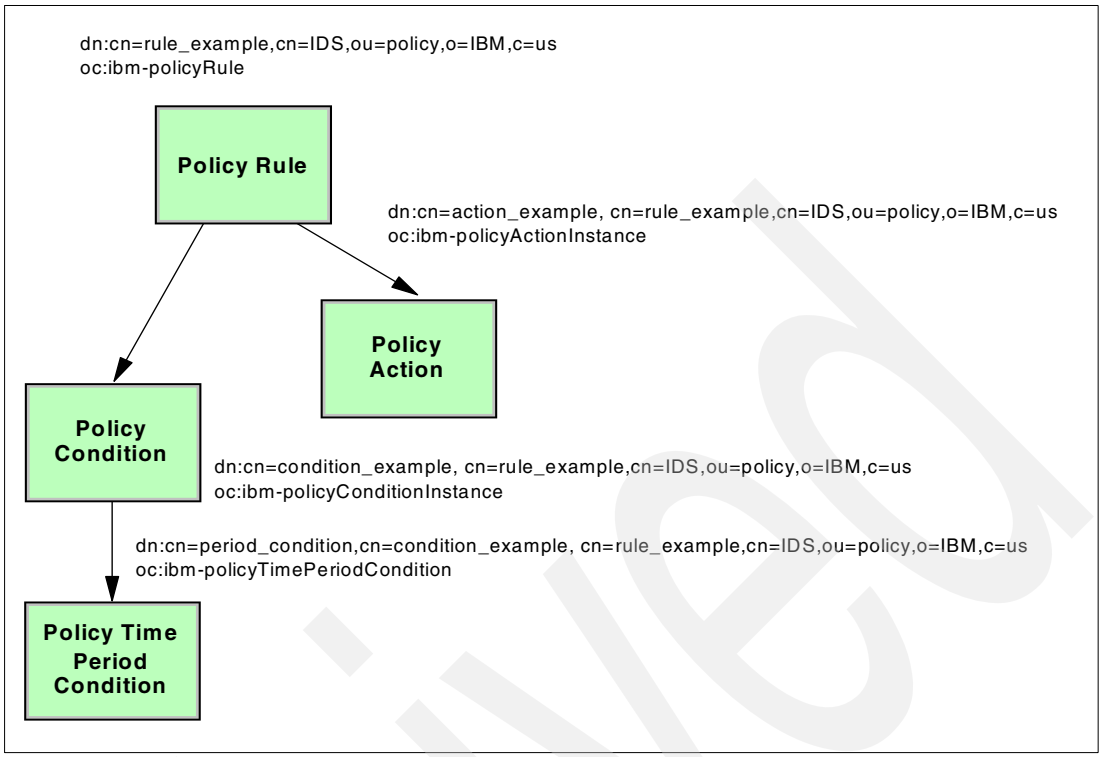


Figure 7-5 Basic IDS policy objects

The policy rule object also contains the attributes:

- ibm-policyRuleEnabled with the following values:
  1 The rule is enabled.
  2 The rule is disabled.
  3 The rule is in debug mode.
- ibm-policyRulePriority is a positive integer used for prioritizing a policy rule relative to other policy rules. A larger value means higher priority. Given two rules that overlap (they both cover the same IP traffic), a rule with higher priority will be applied. The maximum supported value is 255.

Figure 7-6 shows examples of policy rule, condition, and action objects.

Rule Object

cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBMRES

```
ibm-policyrulename: ScanGlobal-rule
description: Example of IDS global scan rule
ibm-policyruleconditionlisttype: 2
objectclass: ibm-policyRule
objectclass: IBM-POLICY
objectclass: TOP
ibm-policyrulepriority: 2
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
ibm-policyruleenabled: 2
cn: scanglobal-rule
```

Condition Object

cn=condassoc1, cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBMRES

```
ibm-policyconditiongroupnumber: 1
ibm-policyconditionnegated: FALSE
description: Rule-specific condition
objectclass: ibm-policyRuleConditionAssociation
objectclass: ibm-idsConditionAuxClass
objectclass: ibm-idsScanConditionAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYCONDITIONAUXCLASS
objectclass: TOP
ibm-idsconditiontype: SCAN_GLOBAL
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
ibm-policyconditionname: ScanGlobal-condition
cn: condassoc1
```

Action Object

cn=actassoc1, cn=scanglobal-rule, cn=IDS, cn=starter, ou=policy, o=IBMRES

```
ibm-idstracedata: RECORDSIZE
ibm-idstracerecordsize: 200
ibm-idssthreshold: 10
ibm-idsloggingelevel: 4
ibm-idsnotification: SYSLOG
ibm-idsnotification: SYSLOGDETAIL
ibm-idsfsinterval: 2
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
objectclass: ibm-policyRuleActionAssociation
objectclass: ibm-idsActionAuxClass
objectclass: ibm-idsScanActionAuxClass
objectclass: ibm-idsNotificationAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYACTIONAUXCLASS
objectclass: TOP
ibm-policyactionname: ScanGlobal-action
ibm-idsfsthreshold: 5
cn: actassoc1
ibm-idsactiontype: SCAN_GLOBAL
ibm-policyactionorder: 1
description: Rule-specific action - Fast scan = 5 in 2 minutes,
             Slow scan = 10 in 8 hours
ibm-idsssinterval: 480
ibm-idstypeactions: LOG
```

Figure 7-6 Examples of IDS objects

Rule-specific and reusable conditions and actions

The IDS LDAP directory tree structure can accommodate pools of already set up conditions and actions that can be shared (that is, are reusable) between several policy rules. The attributes ibm-policyConditionDN and ibm-policyActionDN are used for this purpose.

> **Note:** These attributes are defined in the object classes ibm-policyRuleConditionAssociation and ibm-policyRuleActionAssociation respectively, as shown in the example in Figure 7-7. Looking at the distinguished names used in this example, you can see that the LDAP directory tree has a root of o=IBM,c=US, then it splits into two branches with relative distinguished names of ou=policy and cn=repository. The proposed convention with the IDS samples delivered with z/OS is to put the reusable objects in the cn=repository subtree.

A “rule-specific” condition or action refers to a condition or action defined in the specific rule subtree itself.

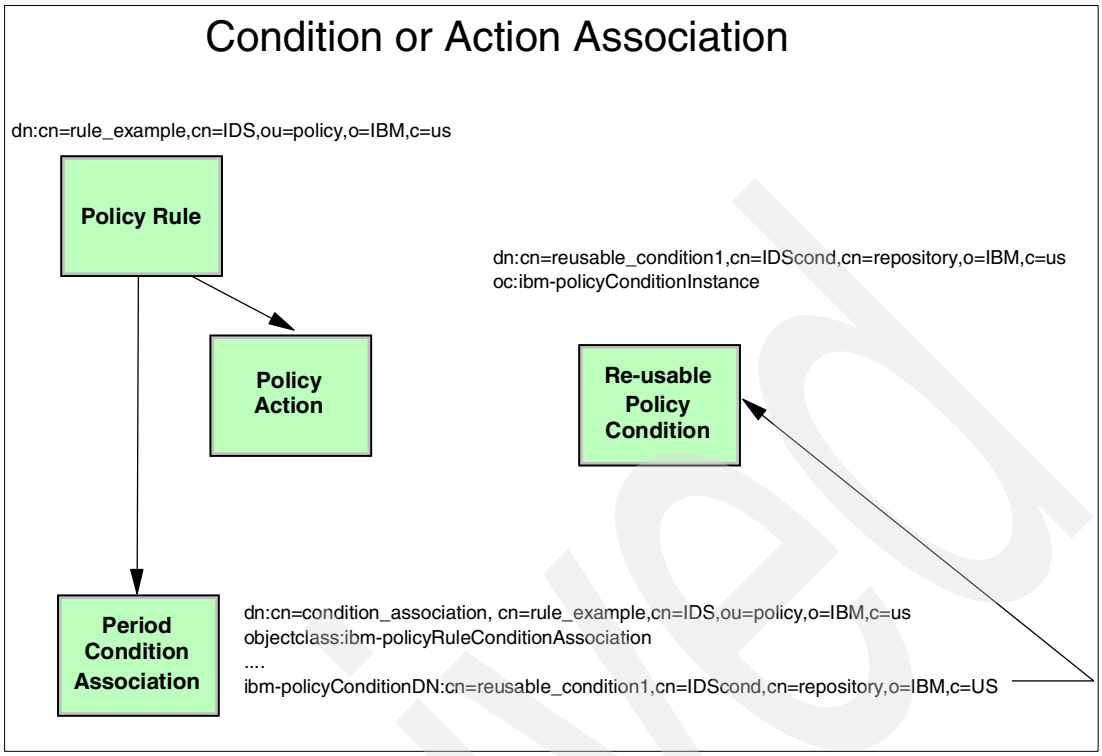


Figure 7-7 Reusable IDS policy conditions

Establishing logical relationships between conditions

It is often desirable to establish a set of logical relationships between condition objects such that an action is taken depending on a logical OR or AND relation, or combination of both, between the conditions. For that purpose the condition objects are related to each other by the value given to the attribute ibm-policyConditionGroupNumber in their respective entries. For instance, all condition objects in the policy with ibm-policyConditionGroupNumber:1 will be associated in the same logical evaluation of the conditions. The type of logical evaluation is specified in the Policy Rule object, using the attribute ibm-policyRuleConditionListType:

- ibm-policyRuleConditionListType:1

  The conditions will be evaluated as an OR of groups of AND'ed conditions. Also called the Disjunctive Normal Form (DNF), this is illustrated in Figure 7-8.

- ibm-policyRuleConditionListType:2

  The conditions will be evaluated as an AND of groups of OR'ed conditions. Also called the Conjunctive Normal Form (CNF), this is illustrated in Figure 7-9.

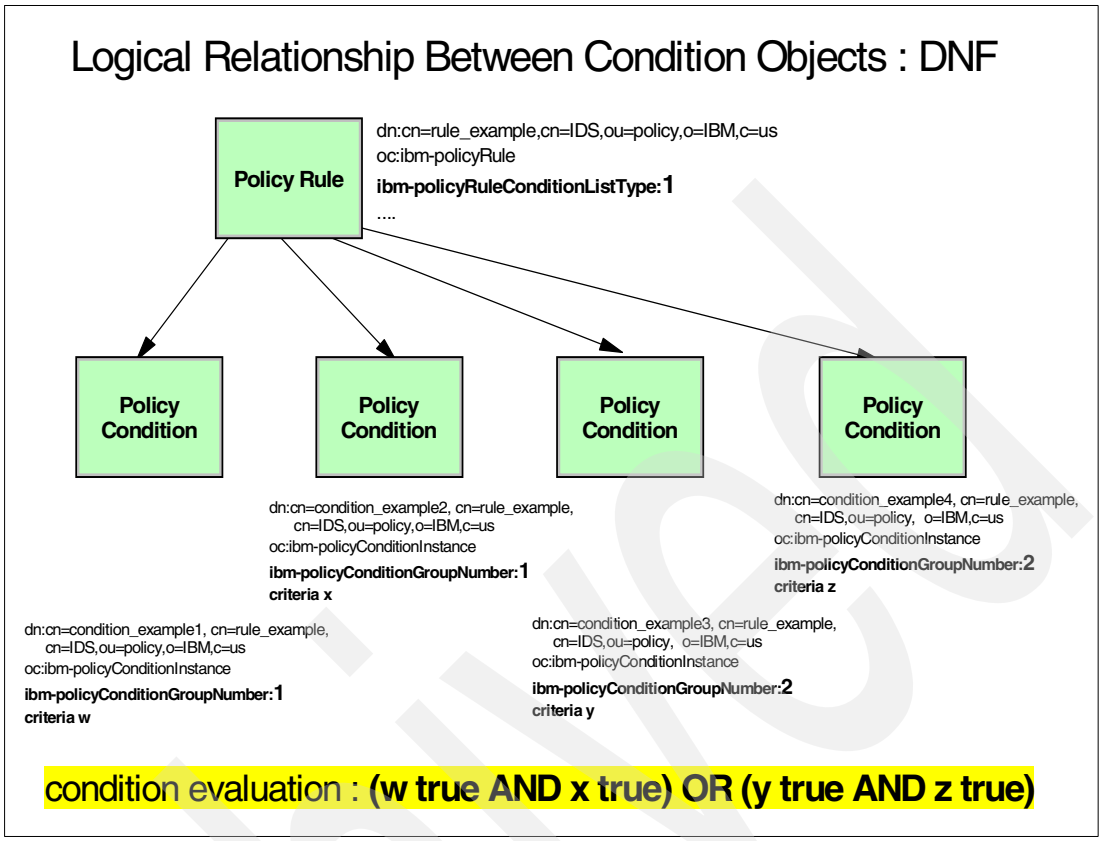


Figure 7-8 Disjunctive Normal Form

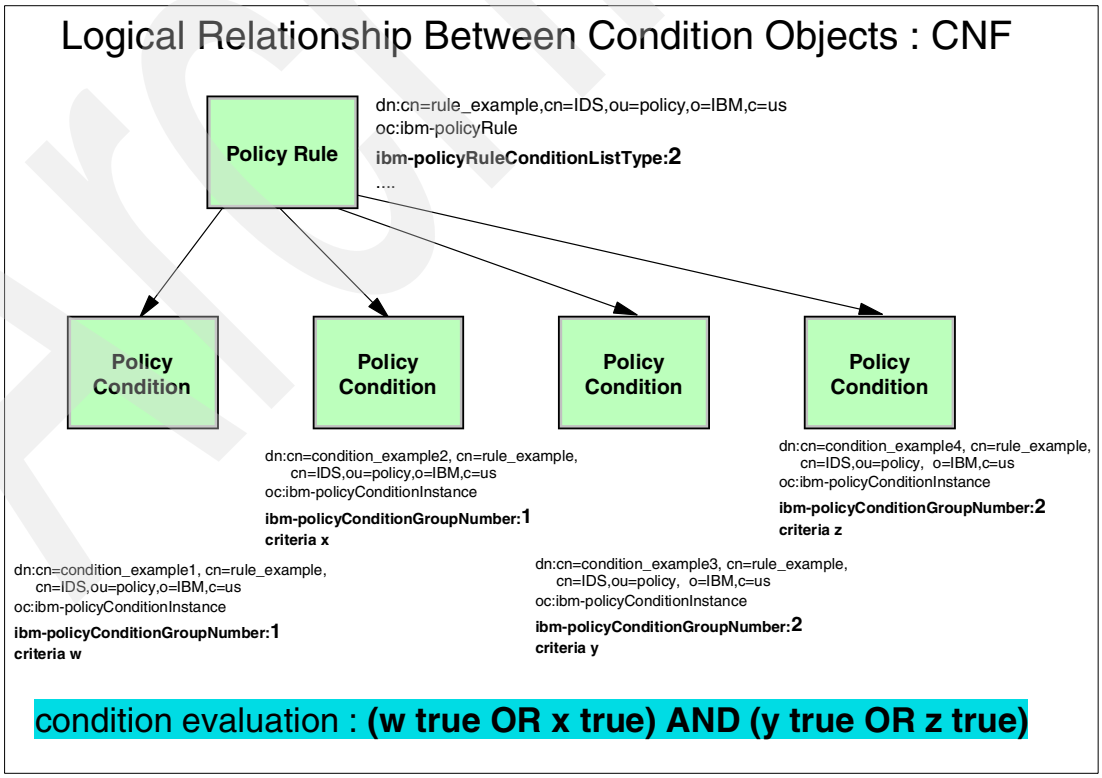


Figure 7-9 Conjunctive Normal Form

Helpful: Detailed information on policies, their structures and attributes, is in the policy examples given in the *z/OS CS IP Configuration Guide* SC31-8775, and in the very detailed comments at the beginning of the policy definition samples in the files indentified in **7.4.7, "Policy Rules samples" on page 195**.

7.4.4 Policies for scan detection

Scan detection

The IDS support defines a scanner as the same source host that accesses multiple unique resources (ports or interfaces) over a specified period of time in the IDS host TCP/IP stack. The number of unique resources (Threshold) and the time period (Interval) can be specified via policy.

IDS supports two categories of scan:

- Fast Scan: Many resources rapidly accessed in a short time period. In z/OS IDS a Fast Scan is defined as 5 unique events in 2 minutes from a single source IP address.
- Slow Scan: Different resources intermittently accessed over a longer period of time. In z/OS IDS a Slow Scan is defined as 10 unique events in 480 minutes (8 hours).

Scan events are counted by the IDS, which compares the number of occurrences of "unique" events within a given time frame with the threshold and interval defined in the scan policy. IDS reports Scan events and, if required by the active policy, traces the Scan IP packets. There are no protective measures triggered for Scan.

Scan unique events

IDS needs to identify several "unique events," meaning non-identical actions from the same source that demonstrate a potential attempt to explore the system's network resources. For that purpose IDS records, in internal tables, inbound requests and their source IP addresses. A scan "unique event" occurs when the same source TCP/IP host switches to another access pattern, like changing the IP protocol, the destination IP address and/or port, or the ICMP type. The event is then potentially "countable" within the given interval time frame. The scan z/OS IDS identifies the following scan types:

- ICMP scans
- TCP port scans
- UDP port scans

Initially the detected request is a "probe instance" and is classified for ICMP as shown in Figure 7-10, for UDP as shown in Figure 7-11 and TCP as in Figure 7-12. This gives the probe event a level of "suspiciousness" which is matched against a sensitivity threshold setup in the scan policy. The probe instance classification and the sensitivity threshold tell IDS whether the probe instance has to be considered as potential scanning and the source host is therefore subject to counting of "unique" events. Figure 7-13 shows the criteria to make a unique event "countable." The full process that leads to detect a scan is depicted in Figure 7-14.

ICMP Scan Probe Instance Classification

Request Type	Destination Address	Event Classification
any	subnet base or broadcast	very suspicious
Information req	single host	possibly suspicious
Subnet Mask req	single host	possibly suspicious
Echo with IP Option Record Route	single host	possibly suspicious
Echo with Record Timestamp	single host	possibly suspicious
Echo or Timestamp, denied by QOS policy	single host	normal
Echo or Timestamp	single host	normal

Figure 7-10 ICMP Scan Probe Instance classification

UDP Scan Probe Instance Classification

Socket State	Event	Event Classification
RESERVED to no one	recv any packet	very suspicious
Unbound, not RESERVED	recv any packet	possibly suspicious - app may be temporarily down
Bound	packet rejected by QOS policy	normal
Bound	packet rejected by FW filtering	possibly suspicious
Bound	recv any packet	normal

Figure 7-11 UDP Scan Probe Instance classification

TCP Scan Probe Instance Classification

Socket State	Event	Event Classification
Any state	recv unexpected flags (SYN+FIN...)	very suspicious
RESERVED	recv any packet	very suspicious
Unbound, not RESERVED	recv any packet	possibly suspicious - app may be temporarily down
Listen	recv SYN	classification deferred if syn queued.
Half open connection	recv ACK	normal - connection handshake completed
Half open connection	recv RST	possibly suspicious - scanner covering tracks?
Half open connection	final time out (and not syn flood)	very suspicious - scanner abandoning handshake?
Any connected state	seq# out of window	normal - perhaps duplicate packet
Any connected state	recv standalone SYN	normal - perhaps peer reboot
Any connected state	final time-out	possibly suspicious - peer abandoned connection

Figure 7-12 TCP Scan Probe Instance classification

Scan sensitivity determines whether a scan event is "countable"

- ►sensitivity established by port for UDP and TCP
- ►sensitivity established by highest priority rule for ICMP

Sensitivity (from policy)	Normal Event	Possibly Suspicious Event	Very Suspicious Event
Low			Count
Medium		Count	Count
High	Count	Count	Count

Figure 7-13 Scan sensitivity

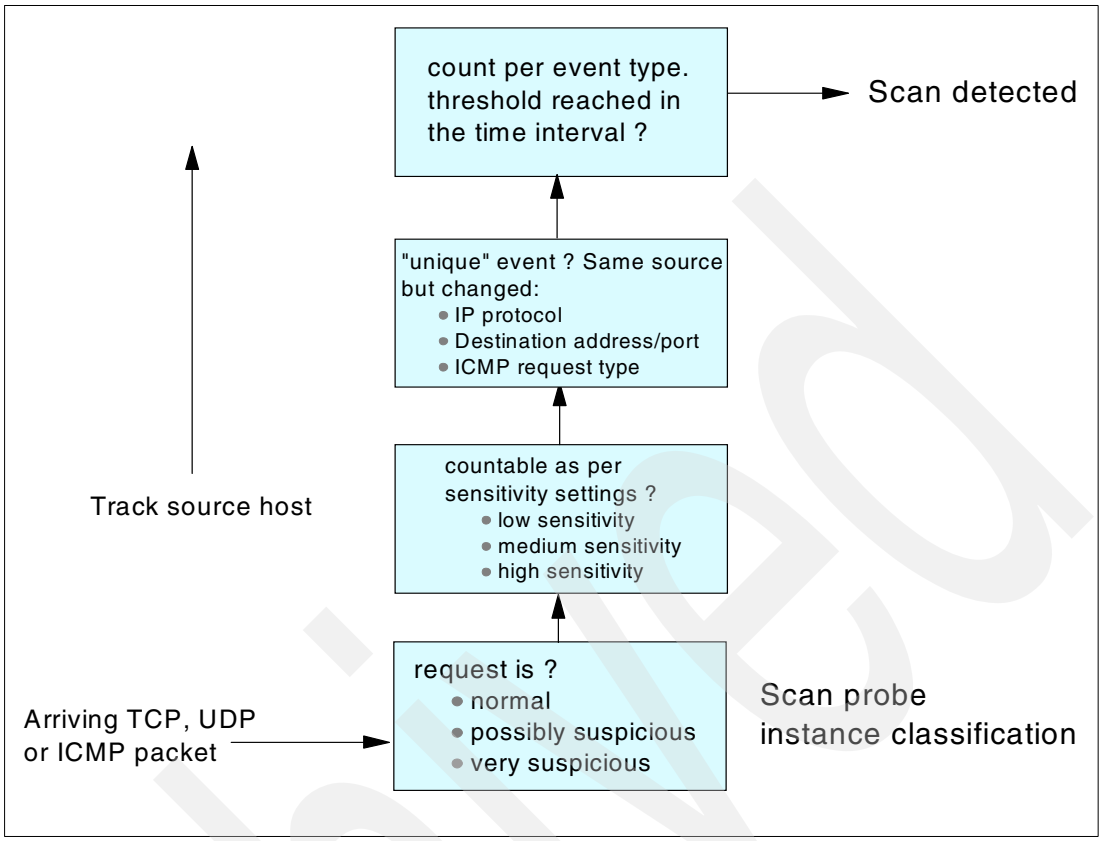


Figure 7-14 Countable events and Scan detection

The SCAN_GLOBAL rule, condition and action

IDS needs a Scan_Global policy to define the Slow Scan and Fast Scan parameters as well as reporting and tracing actions to take when a Scan is detected.

> **Note:** Other Scan detection parameters such as protocol, threshold, interval, and so forth are defined in the SCAN_EVENT policies.

The SCAN_GLOBAL policy entries for conditions and actions contain the following attribute/value pairs:

- ibm-idsConditionType: SCAN_GLOBAL
- ibm-idsActionType:SCAN_GLOBAL

The SCAN_GLOBAL action object contains the following attributes:

- ibm-idsTypeActions with possible values:

  LOG: log detected intrusion
- ibm-idsNotification with possible values:

  NONE

  SYSLOG: Send summary message to SyslogD

  SYSLOGDETAIL: Send into SyslogD but with additional details

  CONSOLE: The MVS operator console

- ibm-idsLoggingLevel with possible values:

  0 to 7, corresponding to the intended SyslogD priority level. For consistency it is recommended that you set ibm-idsLoggingLevel:4, since 4 is the SyslogD warning priority level.

- ibm-idsTraceData with possible values:

  NONE

  HEADER: Trace only the IP datagram header contents.

  FULL: Trace the whole IP datagram.

  RECORDSIZE: For each datagram, trace as many bytes as indicated in the ibm-idsTraceRecordSize attribute value.

- ibm-idsTraceRecordSize, with a value from 0 to 4294967295, the default being 100.
- ibm-idsFSInterval with possible values from 1 to 1440 (Fast Scan interval of 1 minute to 24 hours); the default is 1.
- ibm-idsFSThreshold with possible valuesfrom 1 to 64 (Fast Scan 1 to 64 unique events from a single IP source address); the default is 5.
- ibm-idsSSInterval with possible values from 1 to 1440 (Slow Scan interval of 1 minute to 24 hours); the default is 120.
- ibm-idsSSThreshold with possible values from 1 to 60 (Slow Scan 1 to 60 unique events from a single IP source address); the default is 10.

The SCAN_EVENT policy

The SCAN_EVENT policy entries for conditions and actions contain the attribute/value pairs of:

- ibm-idsConditionType: SCAN_EVENT
- ibm-idsActionType:SCAN_EVENT

The SCAN_EVENT condition object uses the following attributes and possible values:

- ibm-idsProtocolRange with possible values 1,6, 17 (the numeric values for recognized IP protocols):

1	ICMP
6	TCP
17	UDP

- ibm-idsLocalHostIPAddress with possible value of any valid IP address. The default is 0, meaning all IP addresses. This may not be set if the protocol is ICMP (ibm-idsProtocolRange:1).
- ibm-idsLocalPortRange with possible values from 0 to 65535. The default is 0, meaning all ports. This may not be set if the protocol is ICMP (ibm-idsProtocolRange:1).

The SCAN_EVENT action objects use the following attributes:

- ibm-idsSensitivity with the possible values:

  NONE (default)
  HIGH
  MEDIUM
  LOW

  How the sensitivity level applies to Scan countable events is described in Figure 7-13.

- ibm-idsScanExclusion with possible value of any valid IP addresses, 0 to 65535. The default is 0, meaning none. These should be the source IP addresses of “legitimate” scanners that the user knows of. Valid specifications are:

  `<remote_IPV4_address>-<prefix_mask_length>-<from_remote_port>-<to_remote_port>`

Simple setup to trigger a scan detection

> **Attention:** We edited the policy samples provided with z/OS, as opposed to creating our own policies from scratch. It remains, in our opinion, a valid recommendation to start from the samples and to adapt them whenever possible, since building error-free policies involves dealing with a very large variety of attributes.

As an example of simple setup, consider the action for SCAN_GLOBAL described in Example 7-12. The fast scan threshold is set to “1” in the condition, meaning the first countable event will be considered as a scan.

Example 7-12 Scan_Global action for scan simulation

```
url: ldap://9.100.195.42:389/cn=actassoc1, cn=scanglobal-rule, cn=IDS, cn=custom,
ou=policy, o=IBMRES
ibm-idstracedata: RECORDSIZE
ibm-idstracerecordsize: 200
ibm-idsssthreshold: 2
ibm-idsloggginglevel: 4
ibm-idsnotification: SYSLOG
ibm-idsnotification: SYSLOGDETAIL
ibm-idsnotification: CONSOLE
ibm-idsfsinterval: 2
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
objectclass: ibm-policyRuleActionAssociation
objectclass: ibm-idsActionAuxClass
objectclass: ibm-idsScanActionAuxClass
objectclass: ibm-idsNotificationAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYACTIONAUXCLASS
objectclass: TOP
ibm-policyactionname: ScanGlobal-action
ibm-idsfsthreshold: 1
cn: actassoc1
ibm-idsactiontype: SCAN_GLOBAL
ibm-policyactionorder: 1
description: Rule-specific action - Fast scan = 1 in 2 minutes, Slow scan = 2 in 8hours
ibm-idsssinterval: 480
ibm-idstypeactions: LOG
```

Now the corresponding condition for SCAN_EVENT is depicted in Example 7-13, where the IP protocol is “6” (TCP) and the port range spans from 1 to 1023 (low ports).

Example 7-13 TCP low ports scan condition for scan simulation

```
url: ldap://9.100.195.42:389/cn=ScanTcpLowPorts, cn=IDScond, cn=repository, o=IBMRES
ibm-idsprotocolrange: 6
description: Reusable IDS Scan TCP Low Ports condition
ibm-idslocalportrange: 1-1023
objectclass: ibm-policyConditionInstance
objectclass: ibm-idsConditionAuxClass
```

```
objectclass: ibm-idsScanEventConditionAuxClass
objectclass: ibm-idsTransportConditionAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYCONDITIONAUXCLASS
objectclass: IBM-POLICYINSTANCE
objectclass: TOP
ibm-idsconditiontype: SCAN_EVENT
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
ibm-policyconditionname: ScanTcpLowPorts-condition
cn: ScanTcpLowPorts
```

Last, the SCAN_EVENT action, shown in Example 7-15, has a sensitivity of high, which implies that normal TCP requests to low ports will be considered as scan countable events.

Example 7-14 Scan event action for scan simulation

```
url: ldap://9.100.195.42:389/cn=actassoc1, cn=scaneventhigh-rule, cn=IDS, cn=custom,
ou=policy, o=IBMRES
ibm-policyactionname: ScanEventHigh-action
ibm-idssensitivity: HIGH
ibm-policyactionorder: 1
description: Rule-specific action - high sensitivity
objectclass: ibm-policyRuleActionAssociation
objectclass: ibm-idsActionAuxClass
objectclass: ibm-idsScanSensitivityActionAuxClass
objectclass: ibm-idsScanExclusionActionAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYACTIONAUXCLASS
objectclass: TOP
ibm-idsactiontype: SCAN_EVENT
ibm-policykeywords: Scan
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
cn: actassoc1
```

With this security setup, sending a single HTTP request from a browser to the z/OS TCP/IP stack port 79 results in reaching the fast scan threshold, with the MVS console and SyslogD messages shown in Example 7-15 and Example 7-16.

Example 7-15 MVS console message with scan simulation

```
EVENT TYPE: FAST SCAN DETECTED
CORRELATOR 1439 - PROBEID 0300FFF1
SOURCE IP ADDRESS 9.100.203.108 - PORT 0
IDS RULE ScanGlobal-rule
IDS ACTION ScanGlobal-action
IDS EVENT DETECTED 485
```

Example 7-16 SyslogD stored message with scan simulation

```
Dec 4 20:03:23 MVN8/OEKERN   TRMD1    TRMD.TCPIPOE[105]: EZZ8643I TRMD SCAN threshold
exceeded:12/04/2001 20:03:16.86,sipaddr=9.100.203.108,
scantype=F,pthreshold=1,pinterval=2,vs=0,ps=0,norm=1,correlator=1439
Dec 4 20:03:23 MVN8/OEKERN   TRMD1    TRMD.TCPIPOE[105]: EZZ8644I TRMD SCAN
detail:12/04/2001 20:03:16.86,sipaddr=9.100.203.108,correlator=1439,event count=1,event
list:17,9.100.203.108,161,N; ; ;
```

7.4.5 Policies for attack detection and prevention

Based on directives provided in the active policies, IDS monitors specific network activities generically termed “attacks.” On attack detection, IDS can trace packets, if required to do so, and as a protective measure received packets can be discarded. Another option is to have IDS also collecting statistics on the observed attacks.

Attack categories

For all attack categories except flood, a single packet triggers an event. To prevent message flooding to syslogd, a maximum of 100 event messages per attack category will be logged to syslogd within a 5 minute interval.

Attack checking is done on inbound packets, with identification of the following categories of attacks:

- Malformed packets events

  There are numerous attacks designed to crash a system’s protocol stack by providing incorrect or partial header information. These packets are always discarded when received regardless of IDS policy. The source IP address is rarely reliable for these attacks. You can use IDS policy to provide notification of malformed packet attacks.

- Inbound fragment restrictions

  Many attacks are the result of fragment overlays in the IP or transport header. This support allows you to protect your system against future attacks by detecting fragmentation within the first 256 bytes of a datagram. You can use IDS policy to provide notification of a packet that results from a datagram being fragmented in the first 256 bytes, as well as to discard the packet.

- IP protocol restrictions

  While there are 256 possible valid IP protocols, only a handful are in common usage today. This support allows you to protect your system against future attacks by prohibiting those protocols that you are not actively and intentionally using. You can use IDS policy to provide notification of a packet with a restricted IP protocol, as well as to discard the packet.

- IP option restrictions

  As with IP protocols, there are 256 possible IP options, with only a small number currently in common use. This support allows you to prevent misuse of options you are not intentionally using. Note that checking for restricted IP options is performed on all inbound packets, even those forwarded to another system. You can use IDS policy to provide notification of a packet with a restricted IP option, as well as to discard the packet.

- UDP perpetual echo

  Some UDP applications unconditionally respond to every datagram received. In some cases, such as Echo, CharGen or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases it may be polite application behavior to send error messages in response to incorrectly formed requests. If a datagram is inserted into the network with one of these applications as the destination and another of these applications spoofed as the source, the two applications will respond to each other continually. Each inserted datagram will result in another perpetual echo conversation between them. This support allows you to define the application ports that exhibit this behavior. You can use IDS policy to provide notification of a perpetual echo packet, as well as to discard the packet.

- ICMP redirect restrictions

  ICMP redirect packets can be used to modify your routing tables. The IGNOREREDIRECT statement in the TCPIP profile disables ICMP Redirects. You can use IDS policy to provide notification of attempts to modify your routing tables in this manner. You can also use IDS policy to disable ICMP Redirects. ICMP Redirect packets will be ignored or discarded if either IGNOREREDIRECT is specified in the TCPIP profile or if IDS policy is active for ICMP redirect attacks and the associated policy action requests that the packet be discarded (ibm-idsTypeActions:LIMIT).

- Outbound raw restrictions

  Most network attacks require the ability to craft packets that would not normally be built by a proper protocol stack implementation. This support allows you to detect and prevent many of these crafting attempts so that your system is not used as the source of attacks on other systems. As part of this checking, you can restrict the IP protocols allowed in an outbound RAW packet. It is recommended that you restrict the TCP protocol (6) on the outbound raw rule. You can use IDS policy to provide notification of an outbound raw packet that is considered an attack, as well as to discard the packet.

- TCP SYNflood Flood events

  One popular denial of service attack is to flood a public server with connection requests from invalid or nonexistent source IP addresses. The intent is to use up the available slots for connection requests and thereby deny legitimate access from completing. z/OS CS provides protection from this attack regardless of IDS policy. You can use IDS policy to provide notification of an attack so that you can address the situation with your network administrators and service providers in a timely manner. Notification of a flood can include flood start and flood end event messages and tracing of the first 100 packets discarded due to the flood.

For each attack category (for example, restricted IP protocol) the single highest priority rule is mapped at policy change.

Statistics

For IDS attack policy the statistics action provides a count of the number of attack events detected during the statistics interval. The count of attacks is kept separately for each category of attack (for example, malformed) and a separate statistics record is generated for each. If you want to turn on statistics for attacks, it is recommended that you specify exception statistics (ibm-idsTypeActions:EXCEPTSTATS). With exception statistics, a statistics record will only be generated for the category of attack if the count of attacks is non-zero. If statistics is requested (ibm-idsTypeActions:STATISTICS) a record will be generated for every statistics interval regardless of whether an attack has been detected during that interval or not.

Attack policies

Attack policies have the following attributes in their condition objects:

- ibm-idsConditionType with the value ATTACK
- ibm-idsAttackType with possible values (no default):
  - MALFORMED_PACKET
  - FLOOD
  - OUTBOUND_RAW
  - PERPETUAL_ECHO
  - IP_FRAGMENT
  - RESTRICTED_IP_OPTIONS
  - RESTRICTED_IP_PROTOCOL
  - ICMP_REDIRECT

- ibm-idsIPOptionRange (used for RESTRICTED_IP_OPTIONS only) with possible values 0 to 255 (default is 0), indicating the setting of the bits in the IP option byte to be monitored in the datagrams' header.
- ibm-idsLocalPortRange (used for PERPETUAL_ECHO only) with possible values 0 to 65535 (default is 0).
- ibm-idsRemotePortRange(used for PERPETUAL_ECHO only) with possible values 0 to 65535 (default is 0).
- ibm-idsProtocolRange (used for RESTRICTED_IP_PROTOCOL and OUTBOUND_RAW) with possible values 1 to 255 (default is 0), indicating the setting of the bits in the IP protocol byte to be monitored in the datagrams' header.

The action objects for ATTACK policies can use the following attributes and their possible values:

- ibm-idsTypeActions with possible values:
  - STATISTICS: Provide a count of the attack events detected during the statistic interval.
  - EXCEPTSTATS: Generate statistics record only if the count of attacks is not zero during the interval.
  - LOG: Send a message to SyslogD and a subset of information to the MVS operator console.
  - LIMIT: Drop detected attack packets. For attack categories malformed and flood, "ibm-idsTypeActions:LIMIT" has no effect. Malformed packets are always discarded (even with no IDS policy). Syn flood processing always discards SYNs to limit the impact of a syn flood (even with no IDS policy).
- ibm-idsNotification with possible values:
  - NONE
  - SYSLOG
  - SYSLOGDETAIL
  - CONSOLE
- ibm-idsLoggingLevel with possible values 0 to 7; the SyslogD messages desired priority level.
- ibm-idsStatInterval with possible values 0 to 4294967295 seconds; the default is 60.
- ibm-idsMaxEventMessage: The maximum number of attack events messages issued to the MVS console in a 5 minutes interval. Possible values 0 to 4294967295.
- ibm-idsTraceData with possible values:
  - NONE
  - HEADER: Trace only the IP datagram header contents.
  - FULL: Trace the whole IP datagram.
  - RECORDSIZE: For each datagram, trace as many bytes as indicated in the ibm-idsTraceRecordSize attribute value.
- ibm-idsTraceRecordSize, with a value from 0 to 4294967295; the default is 100.

Simple setup to trigger an attack detection

There is actually no simple setup that we can provide to exercize Attack Detection as we did with Scan Detection. Our initial thought was to specify TCP to be a restricted protocol and then send TCP requests that would therefore be considered as an attack. It turned out that the z/OS IDS always ignores, by design, specification of protocol restrictions for the ICMP, UDP, and TCP protocols.

7.4.6 Policies for Traffic Regulation

A Traffic Regulation (TR) policy is intended to have IDS limit the memory resource consumption and queue delay time when TCP connections and UDP inbound queued datagrams reach abnormally high peak values. These peak loads can be an extraordinary volume of legitimate activities, or can be the result of a Denial of Service attack.

The TR function in CS for z/OS 1.2 is an enhancement to the Traffic Regulation Management (TRM) function that was made available with OS/390 V2 R10. TRM policies can still be used with z/OS 1.2 and can reside, contrary to all other IDS policies, in flat files as opposed to an LDAP directory.

Traffic Regulation for TCP

IDS TR policies limit the total number of TCP connections that can be active on a particular port at one time.The z/OS IDS can also limit the number of connections that the same single source IP address can hold on a specific port. This is the so called “fair share” algorithm, where granting of a new connection is based on the percentage of remaining available connections already held by a source IP address. Connection requests in excess of these limits are denied by the IDS TCP/IP stack.

TCP rules are mapped when a local application does a listen on a socket or when an inbound connection handshake completes.

The TCP Traffic Regulation events as reported by IDS are:

- Constrained event: The specific port is at 90% of its connections limit.
- Unconstrained event: The number of currently active TCP connections to a specific port goes below 88% of the port’s connections limit.
- Each connection denied for exceeding either the port’s connection limit or the percent available limit.
- Connection allowed because of a QoS override policy - This is explained later in this chapter.

Fair share algorithm

The fair share algorithm is based on 3 parameters:

- The maximum permitted number of connections to the port: N
- A percentage applied against the number of connections still available for the port: P
- The number of connections already held by the TCP client (recognized by its IP address) requesting a new connection: X

The client will get a new connection if X<P*A, otherwise the new connection request will be denied. The principle of the regulation here is that as fewer connections become available, clients are allowed fewer new connections.

> **Note:** As long as connections are available, a “new” client will always get at least one connection

Examples are shown in Figure 7-15.

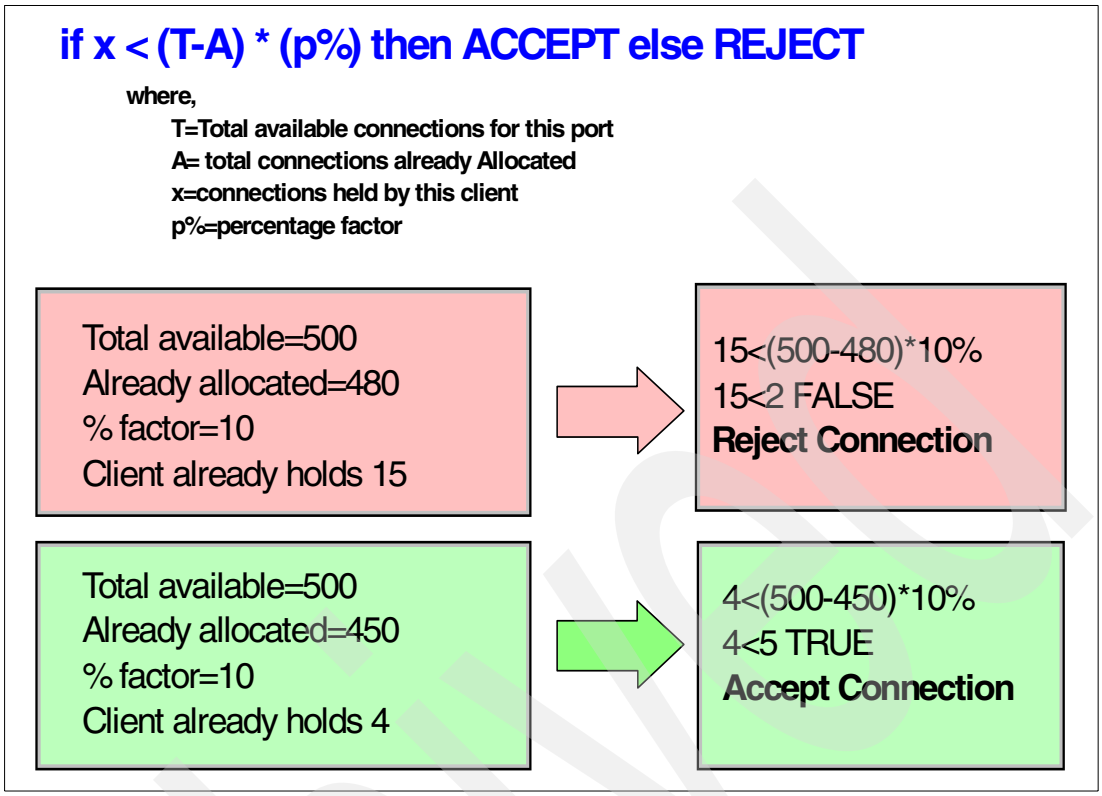


Figure 7-15 Traffic Regulation Management "Fair Share" algorithm.

QoS override policy

Installations may elect to protect the application TCP/IP stack using the *proxy* technology, by which all connection requests actually go through an intermediate host, the proxy, which stands between the non-secure network where the requesting clients are and the serving host, thus protecting the host from most attacks. In such a case the serving application TCP/IP stack sees all requests as originating from the same IP address—that is, the proxy's address. This leads to inconsistency if one wants to use the IDS TCP fair share algorithm in this stack. In that case a QoS-differentiated policy can be used along with the IDS TR policy. As long as the target port is not in the constrained state and the specific source server port and specific outbound client destination IP address match the active QoS policy, then Traffic Regulation honors the QoS policy and can therefore override the fair share algorithm result. As soon as the target port is in the constrained state, Traffic Regulation does not make any more QoS exception.

Traffic Regulation for UDP

As the UDP protocol is connectionless, the traffic regulation operates at the UDP datagram queue length level in the IDS stack. Previous to z/OS 1.2 the only way to introduce regulation, or not, is by setting the UDPQueueLimit keyword to ON or OFF in the TCP/IP profile. The UDP datagrams queue limit is fixed internally in the TCP/IP stacks to 160 KB for Pascal API sockets, or to 2000 datagrams or 2880 KB (whichever is reached first) for sockets that use other APIs. It is recommended always to set UDPQueueLimit to ON, even though the setting is superseded for the ports covered by the IDS Traffic Regulation policy.

IDS UDP Traffic Regulation policy brings finer controls in that it allows you to specify four abstract queue sizes for specific bound IP addresses and ports (others are controlled by UDPQueueLimit):

- VERY_SHORT: Corresponds to an internally fixed value in the TCP/IP stack. As of the writing of this book the queue size is 16 datagrams or 32 KB.
- SHORT: Queue size of 256 datagrams or 512 KB.
- LONG: 2048 datagrams or 4 MB.
- VERY_LONG: 8192 datagrams or 16 MB.

UDP rules are mapped when an inbound packet arrives at a local bound socket.

The UDP Traffic Regulation events as reported by IDS are:

- Constrained Event when the port reaches about 90% of its queue limit.
- Unconstrained Event when the port falls below 88% of its queue limit.

Traffic Regulation policies

TR actions define the reporting, statistics, and tracing actions for covered ports. If the policy specifies action LIMIT, connections or packets that exceed the limits are discarded.

Notes:

1. For TCP, a total connection limit or percentage available limit of zero, with an action of LIMIT, effectively quiesces the application.
2. For TCP, ibm-idsLocalHostIPAddress cannot be specified in any conditions if ibm-idsTRtcpLimitScope PORT is specified.
3. For UDP, a policy for a port without an action of LIMIT effectively makes the application unlimited.
4. Each IDS TR action must specify at least one ibm-idsTypeActions.

IDS TR policies have the following attributes in their condition objects:

- ibm-idsConditionType with the value TR
- ibm-idsProtocolRange with possible values:
  - 6: TCP
  - 17: UDP
- ibm-idsLocalHostIPAddress: The IDS host adapter IP address upon which the policy must apply. The default is “0,” for *all* IP addresses.
- ibm-idsLocalPortRange: The IDS host TCP or UDP ports that must be monitored by the policy. The default is “0,” for *all* ports.

The action objects for TR policies have the ibm-idsActionType attribute with a value of TR. The action objects can also use the following attributes and their possible values:

- ibm-idsTypeActions with possible values:
  - STATISTICS: Provide a count of the TR events detected during the statistic interval.
  - EXCEPTSTATS: Generate statistics record only if the count of TR events is not zero during the interval.
  - LOG: Send a message to SyslogD and a subset of information to the MVS operator console.

  - LIMIT: Discard UDP packets in excess of the fixed Queue Limit, or deny TCP connections above fixed maximum or percentage.
- ibm-idsNotification with possible values:
  - NONE
  - SYSLOG: Log to SyslogD.
  - SYSLOGDETAIL: Log more details to SyslogD.
  - CONSOLE: Log to the MVS console.
- ibm-idsLoggingLevel: The SyslogD messages desired priority level; possible values 0 to 7.
- ibm-idsStatInterval: Possible values 0 to 4294967295 seconds; default is 60.
- ibm-idsTraceData withs possible values:
  - NONE
  - HEADER: Trace only the IP datagram header contents.
  - FULL: Trace the whole IP datagram.
  - RECORDSIZE: For each datagram, trace as many bytes as indicated in the ibm-idsTraceRecordSize attribute value.
- ibm-idsTraceRecordSize, with a value from 0 to 4294967295; the default is 100.
- ibm-idsTRtcpTotalConnections: The total number of TCP connection requests that can be held in the IDS host TCP/IP stack. Connection requests in excess of this value are discarded. The value is 0 to 65535.
- ibm-idsTRtcpPercentage: The percentage used in the fair share algorithm.
- ibm-idsTRtcpLimitScope, with possible values:
  - PORT: The traffic regulation parameters apply to the aggregate of all sockets bound to the target port.
  - PORT_INSTANCE (default): Traffic regulation parameters apply to each socket bound to the target port individually.
- ibm-idsTRudpQueueSize, with possible value:
  - VERY_SHORT
  - SHORT
  - LONG
  - VERY_LONG (default)

Simple setup to trigger a Traffic Regulation event

> **Attention:** We have edited the policy samples provided with z/OS, as opposed to creating our own policies from scratch. It remains, in our opinion, a valid recommendation to start from the samples and to adapt them whenever possible, since building error-free policies involves dealing with a very large variety of attributes.

As an example of simple setup, consider the condition for TR described in Example 7-17, stating that the TR policy applies to port 80 when using the TCP (6) protocol. In the condition object shown in Figure 7-18 on page 194, the maximum number of TCP total connections is set to one, implying that at the first connection request a constrained event is generated and as soon as the connection is reset, an unconstrained event is generated.

Example 7-17 Traffic Regulation condition for attack simulation

```
url: ldap://9.100.195.42:389/cn=trtcpHttpPorts, cn=IDScond, cn=repository, o=IBMRES
```

```
ibm-idsprotocolrange: 6
description: Reusable IDS TR TCP Low Ports condition
ibm-idslocalportrange: 80
objectclass: ibm-policyConditionInstance
objectclass: ibm-idsConditionAuxClass
objectclass: ibm-idsTrafficRegulationConditionAuxClass
objectclass: ibm-idsTransportConditionAuxClass
objectclass: IBM-POLICY
objectclass: IBM-POLICYCONDITIONAUXCLASS
objectclass: IBM-POLICYINSTANCE
objectclass: TOP
ibm-idsconditiontype: TR
ibm-policykeywords: TR
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
ibm-policyconditionname: TrTcpLowPorts-condition
cn: TrTcpLowPorts
cn: trtcpHttpPorts
```

Example 7-18 Traffic Regulation action for attack simulation

```
url: ldap://9.100.195.42:389/cn=trtcpactHttp, cn=IDSact, cn=repository, o=IBMRES
ibm-idstrtcptotalconnections: 1
ibm-idstracedata: RECORDSIZE
ibm-idstracerecordsize: 200
ibm-idsloggingl evel: 4
ibm-idsnotification: SYSLOG
ibm-idsnotification: CONSOLE
ibm-policykeywords: TR
ibm-policykeywords: IDS
ibm-policykeywords: POLICY
ibm-idstrtcppercentage: 50
objectclass: ibm-policyActionInstance
objectclass: ibm-idsActionAuxClass
objectclass: ibm-idsNotificationAuxClass
objectclass: ibm-idsTRtcpActionAuxClass
objectclass: IBM-IDSTRAFFICREGULATIONACTIONAUXCLASS
objectclass: IBM-POLICY
objectclass: IBM-POLICYACTIONAUXCLASS
objectclass: IBM-POLICYINSTANCE
objectclass: TOP
ibm-policyactionname: trtcpHttpLog-action
cn: trtcpactHttp
ibm-idsactiontype: TR
description: IDS TR TCP action TCP(64K,100%) LOG(SYSLOG(4) NOCONSOLE) NOLIMIT TRACE(HEADER)
STATISTICS(60)
ibm-idstypeactions: LIMIT
ibm-idstypeactions: LOG
ibm-idstypeactions: STATISTICS
ibm-idsstatinterval: 5
```

With this policy setup, sending a single HTTP request from a browser to the z/OS TCP/IP stack port 80 results in getting into a constrained, then unconstrained situation, with the MVS console and SyslogD messages shown in Example 7-19 and Figure 7-20.

Example 7-19 MVS Console messages with the attack simulation

```
EZZ8761I IDS EVENT DETECTED 485
 EZZ8762I EVENT TYPE: TCP PORT CONSTRAINED
```

```
 EZZ8763I CORRELATOR 1441 - 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 486
 EZZ8762I EVENT TYPE: TCP PORT UNCONSTRAINED
 EZZ8763I CORRELATOR 1441 - PROBEID 01002400
 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 EVENT TYPE: FAST SCAN DETECTED
EZZ8767I IDS ACTION trtcpHttpLog-action
```

Example 7-20 SyslogD stored messages with the attack simulation

```
Dec 4 20:03:53 MVN8/OEKERN   TRMD1    TRMD.TCPIPOE[105]: EZZ9321I TRMD TCP constrained
entry:12/04/2001
20:03:27.82,lhost=0.0.0.0,port=80,host=9.139.240.34,available=0,total=1,percent=50,correlat
or=1441,probeid=01004400,threshold=0
Dec 4 20:03:53 MVN8/OEKERN   TRMD1    TRMD.TCPIPOE[105]: EZZ9323I TRMD TCP constrained
exit:12/04/2001
20:03:27.90,lhost=0.0.0.0,port=80,host=9.139.240.34,available=1,total=1,percent=50,correlat
or=1441,probeid=01002400,threshold=0,duration=0
```

7.4.7 Policy Rules samples

Communication server provides the following files as examples of policy definitions in LDAP:

- /usr/lpp/tcpip/samples/pagent.ldif

  The top level directory structure for the set of sample QoS and IDS policies.

- /usr/lpp/tcpip/samples/pagent_starter_QOS.ldif

  The starter set sample of LDAP definitions of QoS objects. This file requires the directory structure defined in sample file pagent.ldif.

- /usr/lpp/tcpip/samples/pagent_starter_IDS.ldif

  The starter set sample of LDAP definitions of IDS objects. This file requires the directory structure defined in sample file pagent.ldif.

- /usr/lpp/tcpip/samples/pagent_advanced_QOS.ldif

  The advanced set sample of LDAP definitions of QoS objects. This file requires the objects defined in the QoS starter set sample file pagent_starter_QOS.ldif and the directory structure defined in sample file pagent.ldif.

- /usr/lpp/tcpip/samples/pagent_advanced_IDS.ldif

  The advanced set sample of LDAP definitions of IDS objects. This file requires the objects defined in the IDS starter set sample file pagent_starter_IDS.ldif and the directory structure defined in sample file pagent.ldif.

The samples are structured as indicated in Figure 7-16. The structure is broken into:

- Started set of policies.
- Advanced set of policies.
- Anchor point for customer set of policies created by the users.polample file structure.

The structure also groups objects that act as pointers to subtrees within the entire policy tree. Note that reusable conditions and actions are placed in the *repository* subtree.

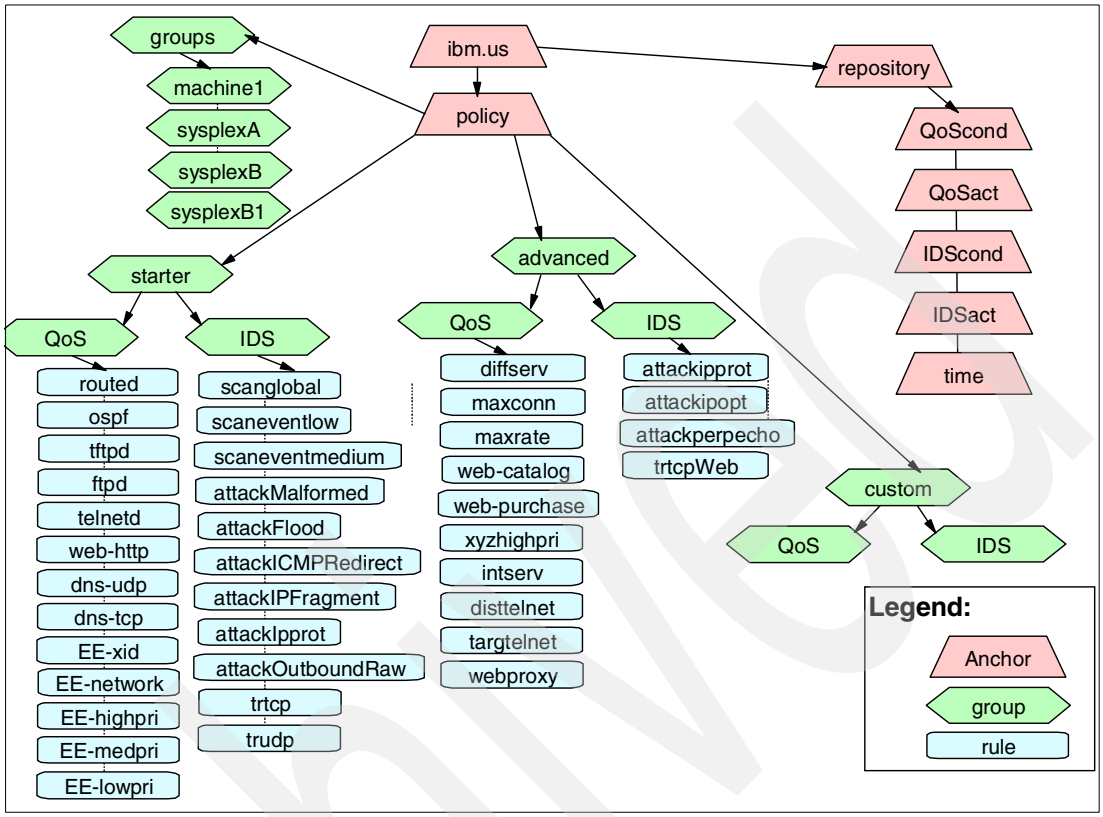


Figure 7-16 z/OS IDS sample policies

7.5 Putting the IDS policy to work

Important: Detailed information on the commands explained here can be found in *z/OS Communications Server: IP System Administrators Commands*, SC31-8781

7.5.1 Starting TRMD and SyslogD

Startup messages are sent to SyslogD and can be retrieved in the log files designated by the SyslogD configuration file.

Note: The SyslogD priority level for IDS messages is set using the attribute "ibm-idsLoggingLevel" in the policy. However, IDS statistics data are always sent with priority level 6 (Information).

7.5.2 Loading the policies with Pagent

Arranging policies in groups

ibm-policyGroup objects can be defined and used as a starting point for the policy load by the Pagent. This provides the capability, by specifying the ibm-policyGroup as the initial search

object in the ReadFromDirectory statement in the Pagent configuration file, to load only specified subtrees, as shown in Figure 7-17.

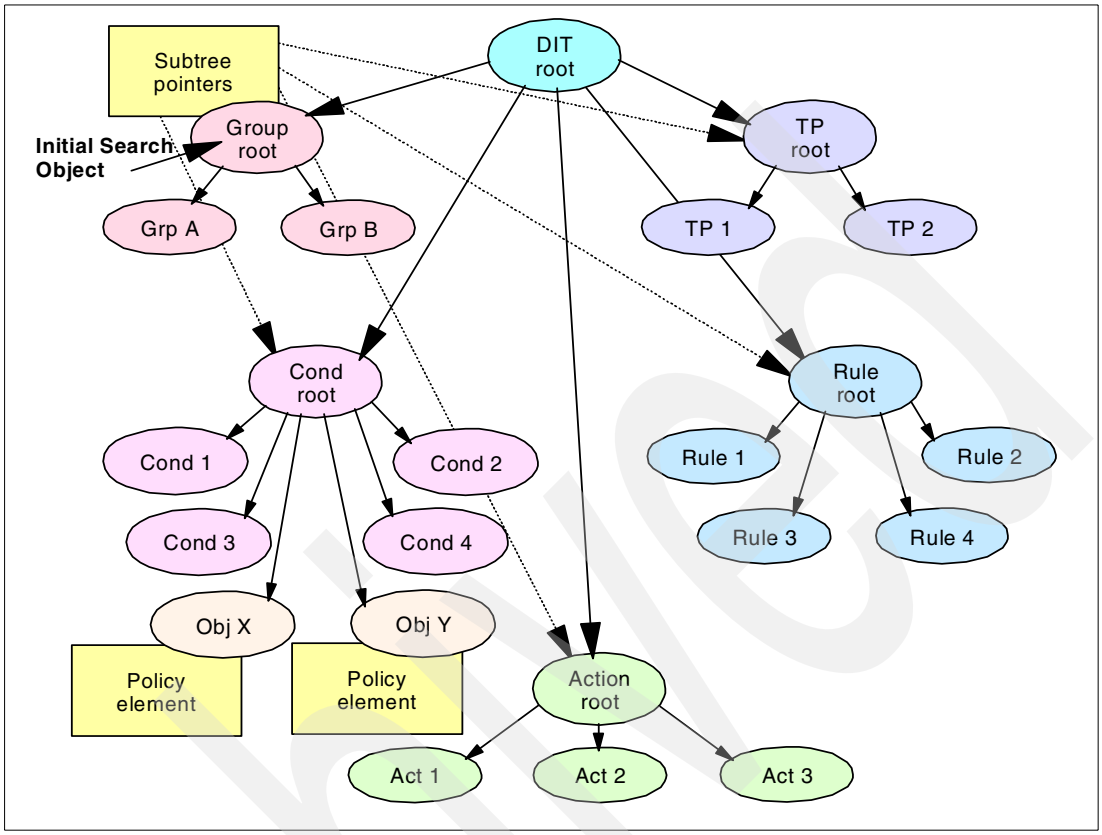


Figure 7-17 Policies groups

Checking for error-free policy load

To verify that policies are correctly defined and functioning properly, consider the following points:

- Are the policies active?
- Is the expected traffic mapping to the correct policies?
- Are the IDS Policy functions working correctly?
- Does anything need to be tuned?

The following sections provide more details about these considerations.

Are the correct policies active?

- Check your LDAP server log or command output for errors encountered when your policies were loaded into LDAP. Some LDAP servers treat consecutive blank lines in an LDIF file as end of file; ensure that all of the policy objects in your LDIF files are acknowledged by LDAP.
- Check your Policy Agent log file for errors while processing your policy.
- Use the `pasearch` command to verify that the intended policies are active and have the expected attributes for the target stack. The pasearch command is explained later in this chapter.

Is the expected traffic mapping to the correct policies?

Use the onetstat -k SUMmary command to ensure that the intended policy has been mapped for each of the Attack types, Scan-Global type, and the Scan-Event type for protocol ICMP. These IDS functions each select the single highest priority policy for their respective types at each policy change.

Use the onestat -k PROTOcol TCP and onestat -k PROTOcol UDP commands to ensure that the intended Scan-Event and TR policies have been mapped to the intended local sockets. These IDS functions select the highest priority policy for the protocol, local port, and local IP address when there is relevant activity against the socket. For TCP this usually entails either a listen or the completion of an inbound connection handshake. For UDP this usually entails either a bind or an inbound datagram. Scan policies are also selected on some inbound error paths.

The onetstat command is explained later in this chapter.

Are the IDS policy functions working correctly?

IDS Policies that include TypeAction:STATISTICS or TypeAction:LOG and Notification:SYSLOG cause the stack to make log record information available to TRMD. If TRMD is running, you can run the IDS report generator TRMDSTAT against the appropriate log files to produce reports on the area of interest.

> **Note:** The Policy Agent does not end when any (or all) stacks end. When the stacks are restarted, active policies are automatically reinstalled. When the Policy Agent is shut down normally (KILL or STOP), and if the TcpImage statement option PURGE was coded, all policies will be purged from this stack.

Refreshing the policies

The MODIFY command can be used to interactively cause the Policy Agent to reread the configuration information and, if requested, download objects from the LDAP server. In addition to this, the Policy Agent will also accept SIGHUP signals to perform the refresh function.

Following is an example of how you would use the modify command:

`MODIFY` or `F procname, REFRESH`

This triggers the Policy Agent to reread the configuration files, and, if requested, download objects from the LDAP server. Basically you download objects from the LDAP server only if a ReadFromDirectory statement is included in the configuration file.

> **Note:** Policies are also refreshed if the SIGHUP signal is received by the Policy Agent. This signal can be sent using the UNIX `kill` command.

Example 7-21 PAGENT refresh command output

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8448I PAGENT DOES NOT HAVE QOSLISTENER AND QOSCOLLECTOR PORTS DEF 5
INED
EZZ8769I ICMP WILL IGNORE REDIRECTS DUE TO INTRUSION DETECTION POLICY
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIPOE
```

Pagent security

Because the Policy Agent can affect system operation significantly, security product authority (for example, RACF) is required to start the Policy Agent. Refer to the EZARACF sample in SEZAINST for sample commands needed to create the profile name and permit users to it.

If Policy Agent's clients (pasearch) are *not* defined as superusers, then security product authority in the SERVAUTH CLASS for that client *must* be defined to retrieve policies. These profiles can be defined by TCP/IP stack (TcpImage) and policy type (ptype = QoS or IDS). Specifying wildcards for profile names is allowed.

```
EZB.PAGENT.<sysname>.<TcpImage>.<ptype>
```

where:

<sysname>	System name defined in sysplex
<TcpImage>	Tcp name for policy information that is being requested
ptype	Policy type that is being requested: QOS - Policy QoS IDS - Policy IDS

Note: Wildcards are allowed on segments of the profile name.

Policy Agent will check all client's requests to verify SERVAUTH class is active and the profile exists for the TcpImages and policy types in the request. If a client's request is for *all* TcpImages and policy types defined then Policy Agent will only return information for which permission is granted to the user. For example, if the request is for *all* policy types, and both QoS and IDS policy types are defined, but the user is only granted permission for the QoS policy types, then only QoS policy information will be returned.

If SERVAUTH class is absent (not RACLIST) or profiles are absent for client's request (TcpImage, policy type), permission is denied and data is not returned.

If SERVAUTH class is active and profiles are present for client's request (TcpImage and policy type) and MVS user is defined for all profiles, permission is granted and data is returned.

If SERVAUTH class is active and profiles are present for client's request (TcpImage and policy type) and MVS user (meaning Policy Agent client) is not defined for all profiles, permission is refused and data is not returned.

Refer to the EZARACF sample in SEZAINST for sample commands needed to create the profile name and permit users to it.

7.5.3 pasearch utility

The `pasearch` command is issued from the z/OS Shell and results in a query to the Pagent to get status information on the installed policies. A few useful keywords for pasearch in the IDS context are:

- -A -i to display all active IDS policies.
- -A -i -n to display the policy names only.
- -A -i -e -f *PolicyFilterName* to display policies, rules, conditions and actions where the attribute ibm-policyRuleName has the value PolicyFilterName (case sensitive).

- -A -i -p StackName to display policy information pertaining to the stack name specified in the Tcpimage statement (when Pagent is serving several stacks in the same z/OS image).

Following is an example of pasearch, where the policy shown in Figure 7-18 is displayed as a pasearch output in Example 7-22.

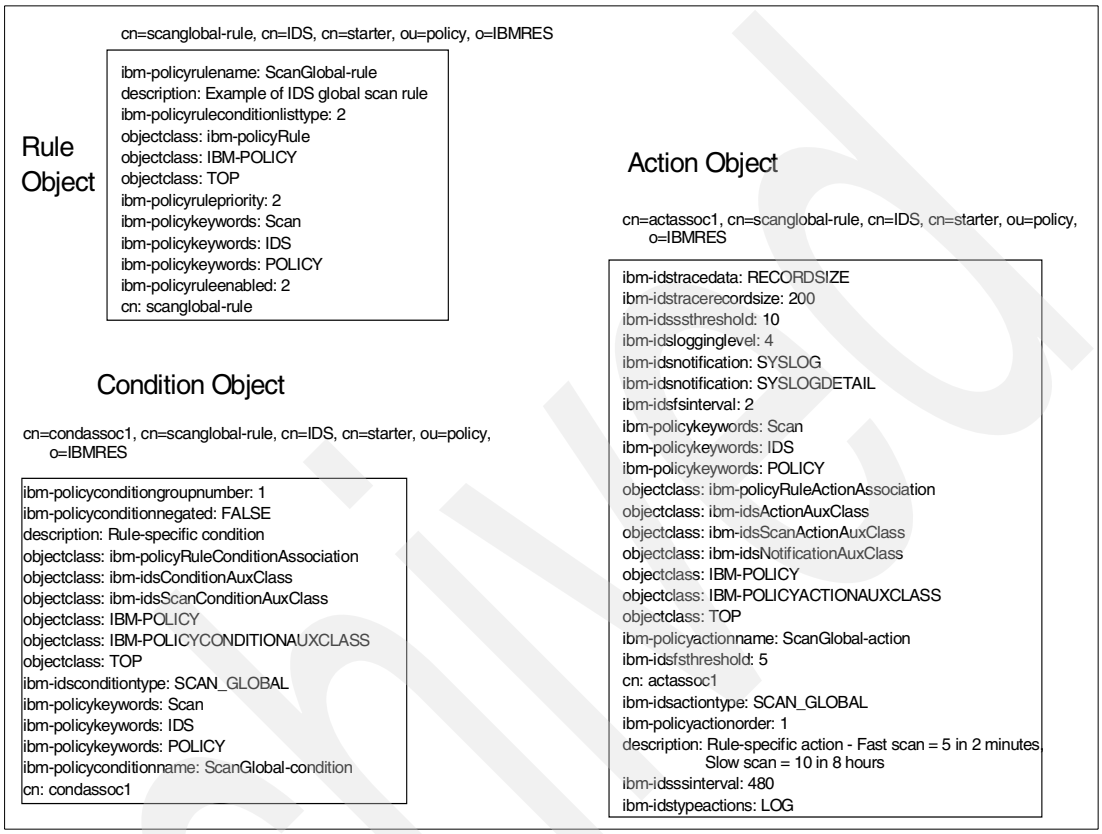


Figure 7-18 Policy objects

Example 7-22 Output of the pasearch command

```
pasearch -A -e -f ScanGlobal-rule

MVS TCP/IP pasearch CS V1R2              TCP/IP Image:      TCPIPOE
  Date:                 02/07/2002       Time:  10:45:13

policyRule:             ScanGlobal-rule
  Version:              3                Status:            Active
  Distinguish Name:     cn=scanglobal-rule,cn=IDS,cn=custom,ou=policy,o=IBMRES
  Group Distinguish Nm: cn=IDS,cn=custom,ou=policy,o=IBMRES
  Weight:               102              ForLoadDist:       FALSE
  Priority:             2                Sequence Actions:  Don't Care
  ConditionListType:    CNF              No. Policy Action: 1
  IdsType:              policyIdsScanGlobal
  policyAction:         ScanGlobal-action
   ActionType:          IDS              Action Sequence:   1
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
```

```
   Month of Year Mask:  111111111111
   Day of Week Mask:    1111111  (Sunday - Saturday)
   Start Date Time:     None
   End Date Time:       None
   From TimeOfDay:      00:00             To TimeOfDay:      24:00
   From TimeOfDay UTC:  00:00             To TimeOfDay UTC:  00:00
   TimeZone:            Local
  Ids Condition Summary:                  NegativeIndicator: OFF
   AttackCondition:
    IdsAttackType:      None
    IPOptionFrom:       0                 IPOptionTo:        0
   TransportCondition:
    LocalPortFrom:      0                 LocalPortTo:       0
    RemotePortFrom:     0                 RemotePortTo:      0
    ProtocolNumFrom:    0                 ProtocolNumTo:     0
   HostCondition:
    LocalIpAddrFrom:    0.0.0.0           LocalIpAddrTo:     0.0.0.0
RemoteIpAddrFrom:   0.0.0.0           RemoteIpAddrTo:    0.0.0.0

  Ids Action:           ScanGlobal-action
    Version:            3                 Status:            Active
    IdsType:            policyIdsScanGlobal
    Distinguish Name:   cn=actassoc1,cn=scanglobal-rule,cn=IDS,cn=custom,ou=policy,o=IBMRES
    Notification Attributes
     Notification:      Syslog SyslogDetail Console
     TraceData:         RecordSize
     TypeActions:       Log
     StatInterval:      60                LoggingLevel:      4
     TraceRecordSize:   200
    Scan Global Attributes
     FastScanInterval:  2                 FastScanThreshold: 5
     SlowScanInterval:  480               SlowScanThreshold: 10
```

The pasearch requestor must be a superuser or must have specific authorizations to specific profiles in the security product database (for example, RACF) SERVAUTH class of security profiles. The name of the profile is of the form:

```
EZB.PAGENT.<sysname>;<tcpimage>.<ptype>
```

where:

<sysname> is the system name

<tcpimage> is the TCPIP stack name

<ptype> is QOS or IDS

Note: Use of wildcards is possible in the segment of the profile names, for example:

```
EZB.PAGENT.SYSPO.*.*
```

7.5.4 Netstat command and options

Netstat can provide IDS-related information. In its operator console form:

```
D TCPIP,procname,N,IDS SUM
D TCPIP,procname,N,IDS,PROTO UDP or D TCPIP,procname,PROTO TCP
```

Or in its TSO form:

```
NETSTAT IDS SUM
```

```
NETSTAT IDS PROTO UDP or NETSTAT IDS PROTO TCP
```

Use the onetstat -k SUMmary command to ensure that the intended policy has been mapped for each of the Attack types, Scan-Global type and the Scan-Event type. Use the onetstat -k PROTOcol TCP and onetstat -k PROTOcol UDP commands to ensure that the intended Scan-Event and TR policies have been mapped to the intended local sockets.

Example 7-23 and Figure 7-24 are examples of the netstat command.

Example 7-23 onetstat summary command output

```
onetstat -k SUM

MVS TCP/IP onetstat CS V1R2       TCPIP Name: TCPIPOE          11:20:44
Intrusion Detection Services Summary:
Scan Detection:
  GlobRuleName: ScanGlobal-rule
  IcmpRuleName: ScanEventMedium-rule
  TotDetected:  1409        DetCurrPlc: 0
  DetCurrInt:   0           Interval:   60
  SrcIPsTrkd:   0           StrgLev:    00000
Attack Detection:
  Malformed Packets
    PlcRuleName: AttackMalformed-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  OutBound RAW Restrictions
    PlcRuleName: AttackOutboundRaw-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  Restricted Protocols
    PlcRuleName: AttackIPprot-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  Restricted IP Options
    PlcRuleName: AttackIPopt-resA
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  ICMP Redirect Restrictions
    PlcRuleName: AttackICMPRedirect-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  IP Fragment Restrictions
    PlcRuleName: AttackIpFragment-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
  UDP Perpetual Echo
PlcRuleName: *NONE*
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   0
  Floods
    PlcRuleName: AttackFlood-rule
    TotDetected: 0           DetCurrPlc: 0
    DetCurrInt:  0           Interval:   5
Traffic Regulation:
  TCP
    ConnRejected: 0           PlcActive: Y
  UDP
    PckDiscarded: 0           PlcActive: Y
```

Example 7-24 onetstat protocol command output

```
onetstat -k PROTO TCP

MVS TCP/IP onetstat CS V1R2       TCPIP Name: TCPIPOE          11:21:05
Intrusion Detection Services TCP Port List:
TcpListeningSocket: 0.0.0.0..21
  ScStat: C  ScRuleName: ScanEventLow-rule
  TrStat: C  TrRuleName: trtcpFtp-rule
  TrPortInst: Y  TrCorr: 0           MxApp: 0           MxHst: 0
  SynFlood:   N
TcpListeningSocket: 0.0.0.0..80
  ScStat: S  ScRuleName: ScanEventLow-rule
  TrStat: S  TrRuleName: trtcpHttp-rule
  TrPortInst: Y  TrCorr: 0           MxApp: 0           MxHst: 0
  SynFlood:   N
TcpListeningSocket: 0.0.0.0..512
  ScStat: S  ScRuleName: ScanEventLow-rule
  TrStat: S  TrRuleName: TRtcp-rule
  TrPortInst: Y  TrCorr: 0           MxApp: 0           MxHst: 0
  SynFlood:   N
TcpListeningSocket: 0.0.0.0..513
  ScStat: S  ScRuleName: ScanEventLow-rule
  TrStat: S  TrRuleName: TRtcp-rule
  TrPortInst: Y  TrCorr: 0           MxApp: 0           MxHst: 0
  SynFlood:   N
TcpListeningSocket: 0.0.0.0..514
  ScStat: S  ScRuleName: ScanEventLow-rule
  TrStat: S  TrRuleName: TRtcp-rule
  TrPortInst: Y  TrCorr: 0           MxApp: 0           MxHst: 0
  SynFlood:   N
```

Securing the NETSTAT/onetstat command

The NETSTAT command can be secured by RACF profiles in the SERVAUTH class, with a name of the form:

```
EZB.NETSTAT.mvsname.tcpprocname.IDS
```

See the sample EZARACF member for examples of the security product commands used to create the resource names. If the SERVAUTH class is not active or if the security product resource name is not defined, access to the NETSTAT/onetstat command will not be restricted.

7.5.5 TRMDSTAT utility

The `trmdstat` command is a UNIX utility that gives a consolidated view of the log messages written out by the Traffic Regulation Management daemon (TRMD). It is entered in the z/OS shell. One useful command syntax is:

```
trmdstat -I logfilename
```

to get an IDS events summary. If no initial time (-i) is specified, trmdstat builds the summary starting with the first record in the log file. If no final time (-f) is specified, the summary ends with the last record in the log file.

Additional commands yield the following information:

`trmdstat -N -D` Scan event details

`trmdstat -A -D` Attack event details

`trmdstat -S -A` Statistics on attacks

The options for TRMDSTAT are shown in Table 7-1. Example 7-25 and Example 7-26 show samples of the TRMDSTAT outputs.

Table 7-1 TRMDSTAT options

`-A`	Specifies the attack summary.
`-C`	Specifies the connection summary.
`-F`	Specifies the flood summary.
`-I`	Specifies the enhanced summary.
`-N`	Specifies the scan summary.
`-T`	Specifies the TCP summary.
`-U`	Specifies the UDP TR summary.
no value	Specifies the overall summary (R10 format).
`-D`	Specifies a detailed report. This option is available only when an A/C/F/N/T/U option is specified.
`-E`	Specifies an extended summary report. This option is available only when a T option is specified.
`-S`	Specifies a statistics report. This option is available only when an A/T/U option is specified.
`-h ip_address`	Specifies that information is to be collected about the ip address. Available with the A/C/N/Q/U and D options.
`-k ip_address`	Specifies that information is to be collected about the peak ip address. Available with the T and S options.
`-s ip_address`	Specifies that information is to be collected about the source ip address. Available only with the A,T and D options.
`-t ip_address`	Specifies that information is to be collected about the destination ip address. Available only with the A,T and D options.

Example 7-25 TRMDSTAT enhanced summary output

```
trmdstat -I /tmp/trmd/trmd.log1

trmdstat for z/OS CS V1R2          Thu Feb  7 11:42:11 2002

Stack Name           : ALL
Log Time Interval    : Feb  4 17:44:23  - Feb  4 20:40:53
Stack Time Interval  : Feb  4 17:44:18  - Feb  4 20:40:38
TRM Records Scanned  : 1533
Port Range           : ALL

TCP - Traffic Regulation
------------------------------------------------
Connections would have been refused :          0
Connections refused                 :          0

Constrained entry logged            :          0
Constrained exit logged             :          0
Constrained entry                   :         20
Constrained exit                    :         20

QOS exceptions logged               :          0
QOS exceptions made                 :          0

UDP - Traffic Regulation
------------------------------------------------
Constrained entry logged            :          0
Constrained exit logged             :          0
Constrained entry                   :          0
Constrained exit                    :          0

SCAN Detection
------------------------------------------------
Threshold exceeded                  :         82
Detection delayed                   :          0
Storage constrained entry           :          0
Storage constrained exit            :          0

ATTACK Detection
------------------------------------------------
Packet would have been discarded    :          0
Packet discarded                    :          0

FLOOD Detection
------------------------------------------------
Accept queue expanded               :          0
SYN flood start                     :          0
SYN flood end                       :          0
```

Example 7-26 TRMDSTAT scan summary output

```
trmdstat -N /tmp/trmd/trmd.log1

trmdstat for z/OS CS V1R2          Thu Feb  7 11:52:17 2002

Stack Name           : ALL
Log Time Interval    : Feb  4 17:44:23  - Feb  4 20:40:53
Stack Time Interval  : Feb  4 17:44:18  - Feb  4 20:40:38
TRM Records Scanned  : 1545
Port Range           : ALL

                              SCAN  Summary

  IP Address           Scans                   Suspicion Level
                   Fast      Slow       Very      Possibly     Normal
--------------- ---------- ---------- ---------- ---------- ----------
32.238.67.190           2          0          2          0          0
9.100.203.108          78          0          0          0         78
9.139.240.34            1          0          0          0          1
9.26.166.125            1          0          0          0          1
```

8

IDS configuration using zIDS Manager

This chapter serves as a pseudo user's guide for the zIDS Manager tool. The zIDS Manager is a tool designed to allow a network administrator to produce the following:

- A file the LDAP Server can process (using either LDAP Version 2 or 3)
- A configuration file for the Policy Agent (PAGENT)

This chapter provides an overview of zIDS Manager, and describes how to install and use this tool.

8.1 What a zIDS is

zIDS Manager enables centralized configuration of Intrusion Detection policy for z/OS V1R2 using LDAP as a policy repository. It provides a user-friendly interface with help panels to free network administrators from having to know LDAP Policy Schema and the complexity of directly writing to an LDAP Server.

The zIDS Manager is a tool designed to allow a network administrator to produce the following:

- A file the LDAP Server can process (using either LDAP version 2 or 3)
- A configuration file for the Policy Agent (PAGENT)

zIDS Manger's Graphical User Interface (GUI) provides a user-friendly front end for the entry of policy information. There is also the flexibilty to save the (possibly incomplete) information you entered in an XML file format on your local machine for future use. Once the policies are complete, the tool can covert the XML file to an LDIF file and send the information to the LDAP Server and/or optionally save the file locally in the LDIF file format.

zIDS Manager also produces a configuration file with the LDAP Server information required by PAGENT. This file can be sent via FTP or moved to and placed in the PAGENT configuration file manually.

The IDS functionality provided by PAGENT must use the LDAP Server to retrieve its policy information. The LDAP Server requires that the policies be coded in accordance with RFC 2849 and as per the RFC, transferred to the LDAP Server in an LDIF file format. IDS policies cannot be configured in the PAGENT configuration file. Based on this premise, there are two methods for generating the LDIF file:

- Manually coding the policies (LDIF) file, and transferring the information to the LDAP Server
- Using the zIDS Manager to generate the LDIF file and using it to automatically send the information to the LDAP Server

We at the ITSO recommend using the zIDS Manager tool to generate and transfer the policy information. The primary reason is that the network administrator does not need to learn LDAP syntax in order to create a policy. Also, the amount of time saved by using the zIDS Manager to generate the LDIF file as opposed to manually coding the LDIF file is significant.

Figure 8-1 on page 209 shows the communication flow between the zIDS Manager, the LDAP Server, and the PAGENT application. This chapter focuses on the first flow, the building of the policies and transfer of the policies to the LDAP Server.

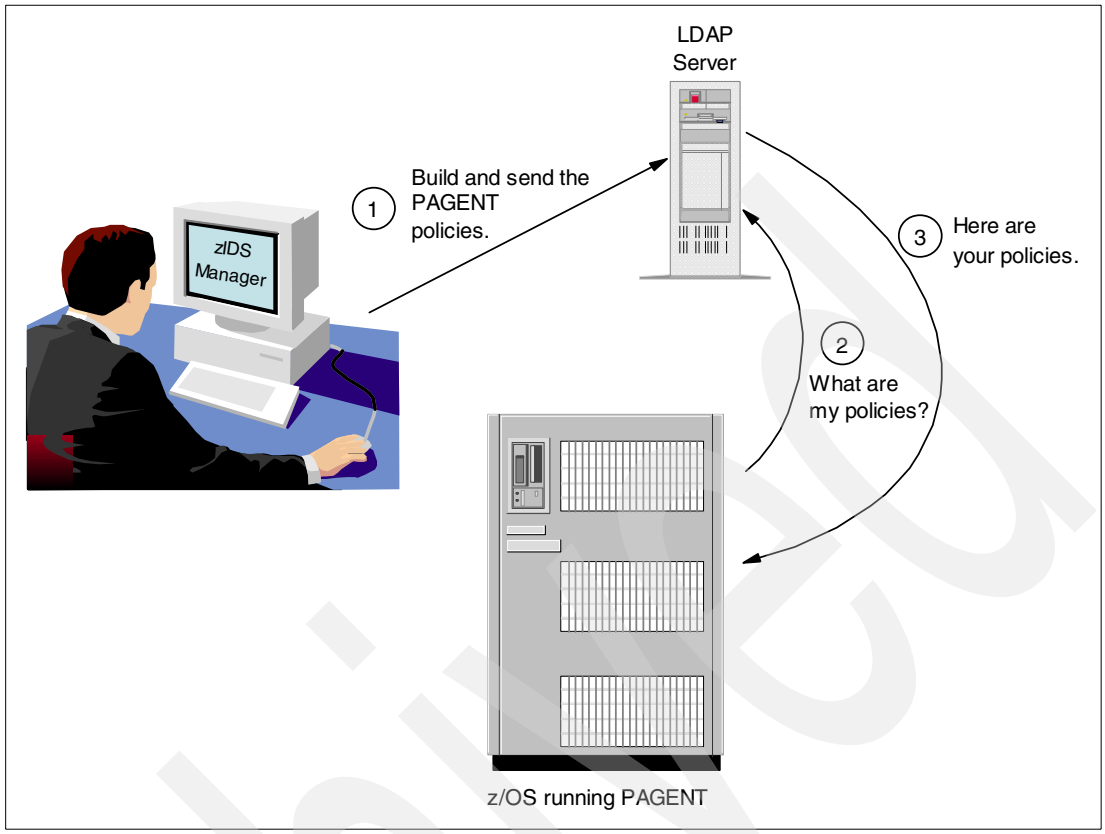


Figure 8-1 Policy flow

zIDS Manager is a tool for network administrators. Therefore, before you begin you should:

- Read the chapter on policy-based networking in *z/OS V1R2 Communications Server: IP Configuration Guide,* SC31-8775.
- Have information about the LDAP Server to be used, that is, the server address and port number, the LDAP protocol version (2 or 3), whether a backup LDAP Server is used, and whether SSL is used.
- Be familiar with your particular environment so that you can make decisions on what events are to be detected under what circumstances and what action to take.

8.2 Requirements and support

This section outlines the requirements and support of the zIDS GUI.

8.2.1 Requirements

The IDS Manager requires Java 1.3.1 and Windows 2000 or Linux to run. The Java executable can be obtained at the following URL:

http://java.sun.com/

8.2.2 Support - Legal notice

IBM provides this code on an *as is* basis without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

Support is provided on a “best effort” basis through a newsgroup. Visit the newsgroup ibm.software.commserver.os390.zids-manager on the server news.software.ibm.com.

The tested operating systems have been Linux and Windows 2000. Any operating system that can support Java 1.3.1 should be able to run the application.

8.3 Download and installation

The download and installation instructions are written for Windows 2000 and Linux. The following information and executables are located at:

`http://www-3.ibm.com/software/network/commserver/downloads/zidsmanager.html`

8.3.1 Windows 2000 steps

The steps for Windows 2000 are:

1. Download this file to your Windows system: zIDSManager.zip.
2. Use an unzip program to extract zIDSManager.exe.
3. Execute zIDSManager.exe.
4. Go to **Start -> Programs -> zIDS Manager**.

8.3.2 Linux steps

The steps for Linux are:

1. Download this file to your Linux system: zidsmgr.tar.
2. Untar the file with `tar -xvf zidsmgr.tar`.
3. Execute `./zidsmgr`.

8.4 Using the GUI

This section is intended to help the network administrator manage and understand the Graphical User Interface provided. Each first level directory will be discussed and screen captures provided to assist in the education. The first level directories are:

- zIDS Manager configuration (8.4.1, “zIDS Manager configuration” on page 211)
- PAGENT configuration (8.4.2, “PAGENT configuration” on page 212)
- Work with IDS objects/rules (8.4.3, “Work with IDS objects/rules” on page 216)

Upon completion of this chapter, the reader will have created

- Reusable objects
- A scan global policy
- An attack, scan event, and TR TCP (condition, action, and policy)

The first window displayed when starting the zIDS Manager is shown in Figure 8-2 on page 211.

Note: zIDS Manager help is available via the **Help** menu option. If detailed information is needed for a particular field, place the cursor in the desired field and press the F1 key.

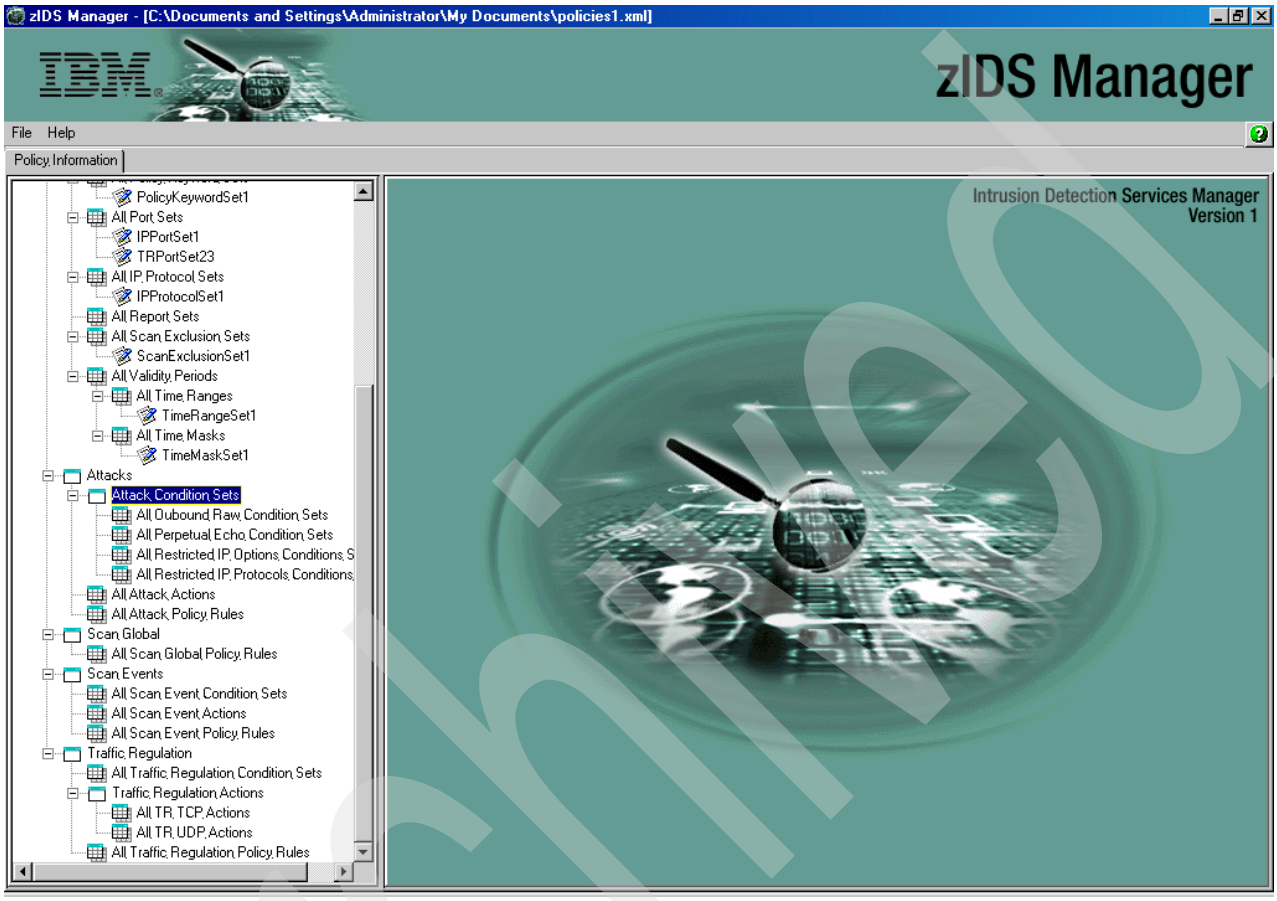


Figure 8-2 zIDS Manager

8.4.1 zIDS Manager configuration

One of the first steps in using zIDS Manger is to configure the LDAP Server settings. This is done from the section zIDS Manager Configuration by selecting **LDAP Information**. The settings we used are shown in Figure 8-3 on page 212. This sets up the configuration information that is needed for communication between the zIDS Manager and the LDAP Server.

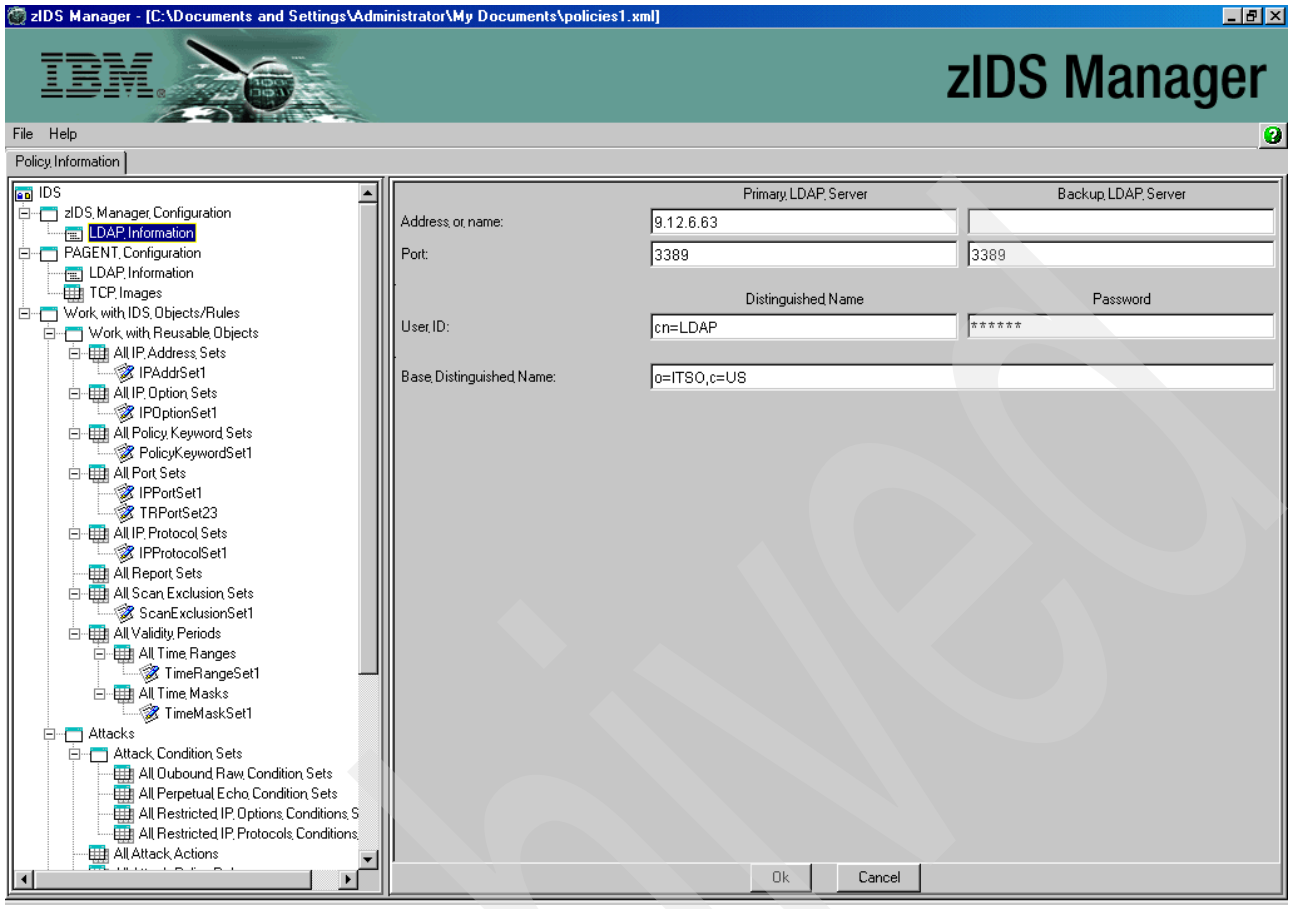


Figure 8-3 zIDS LDAP configuration information

> **Note:** There is not an option for SSL. The connection between the zIDS Manager and the LDAP Server is a non-secure connection.

zIDS Manager to LDAP Server communication

Verify that you can communicate with the LDAP Server by clicking **File -> Send to LDAP**. This is considered successful if you receive the Creating LDIF icon with the corresponding text `Updating LDAP. Please wait` for a few seconds and are then returned control with the icon disappearing and no error messages. Keep in mind that we are not sending any new policy information to the LDAP Server; we basically sent the default IBM-provided policy information for connectivity verification. This same step must be repeated after the policies have been coded and saved to a file by selecting **File -> Save** or **File -> Save As** from the menu.

8.4.2 PAGENT configuration

Next we will configure the PAGENT configuration information by selecting **PAGENT Configuration**. Now select **LDAP Information** and the screen appears as shown in Figure 8-4 on page 213. We will use the same information as entered for the LDAP Information in the zIDS Manger Configuration section.

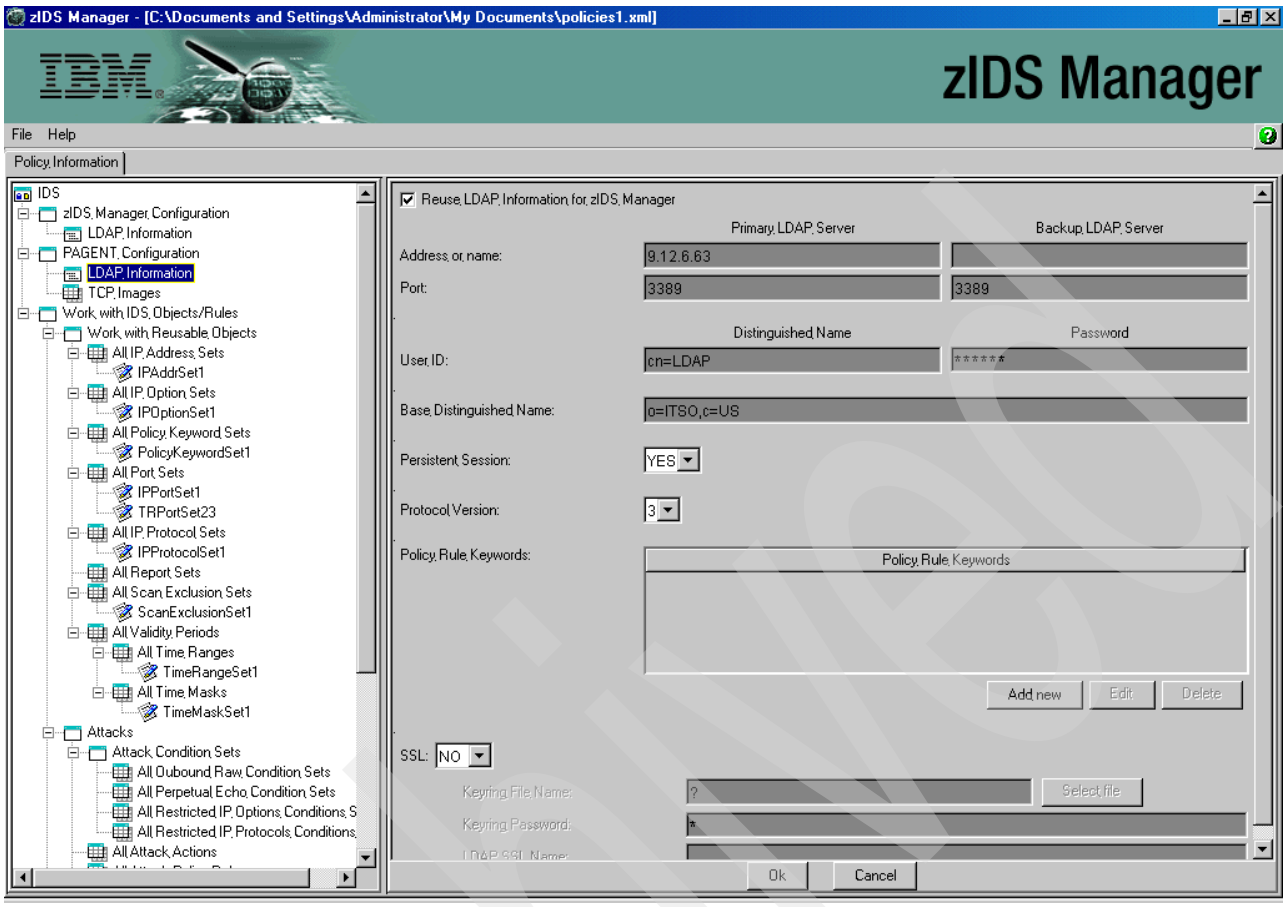


Figure 8-4 PAGENT LDAP configuration information

We can now configure the TCP images. This will specify what TCP/IP stacks are supporting the policies. First, click **TCP Images**. You will see a summary of information in the right pane. If you right-click you will see the window shown in Figure 8-5 on page 214. This allows you to add multiple TCP/IP images to the configuration file.

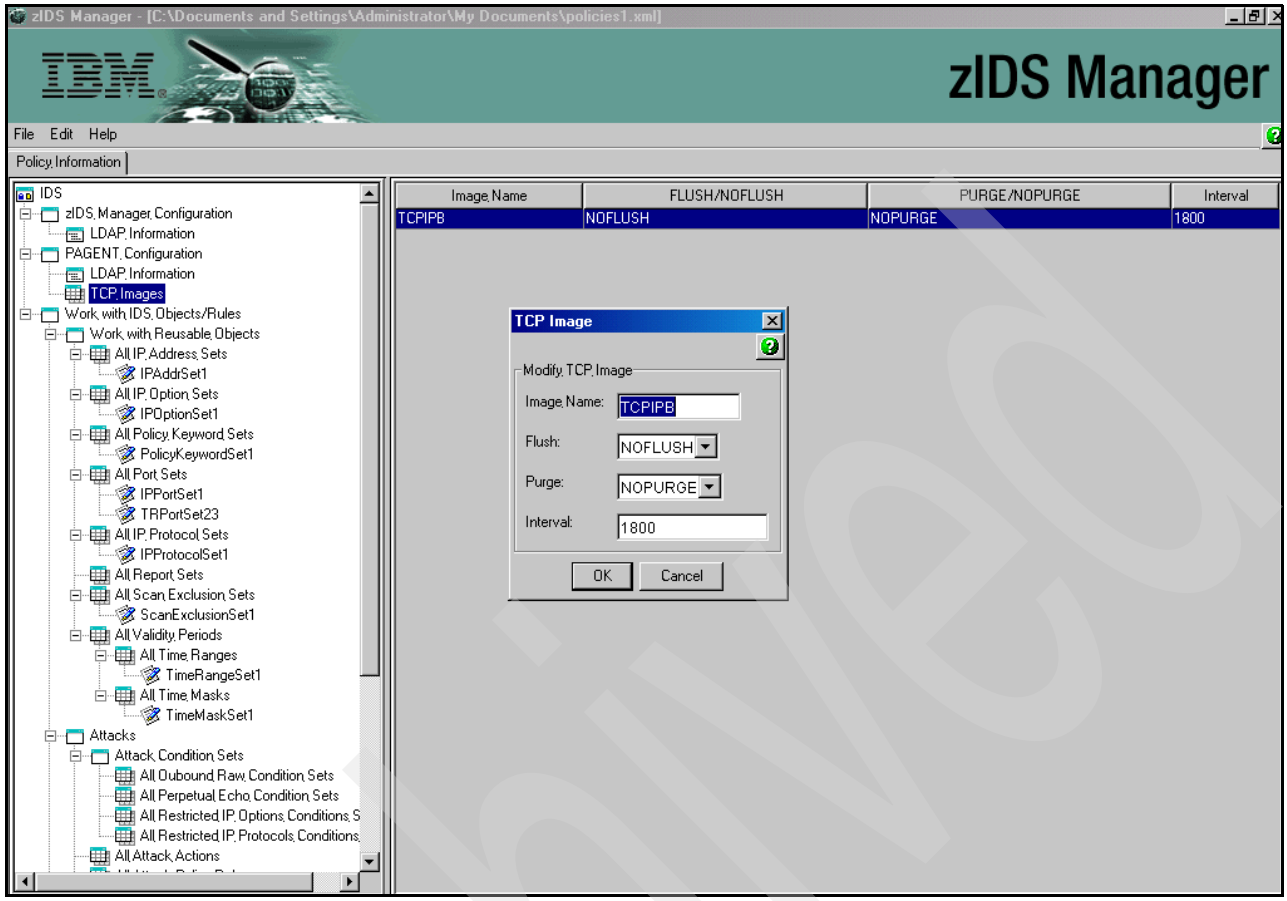


Figure 8-5 PAGENT LDAP TCP/IP configuration

Once your TCP/IP image has been created you have the ability to add, modify, or delete the TCP/IP Image object by highlighting it in the right pane right-clicking the object. The pop-up menu that appears is shown in Figure 8-6 on page 215.

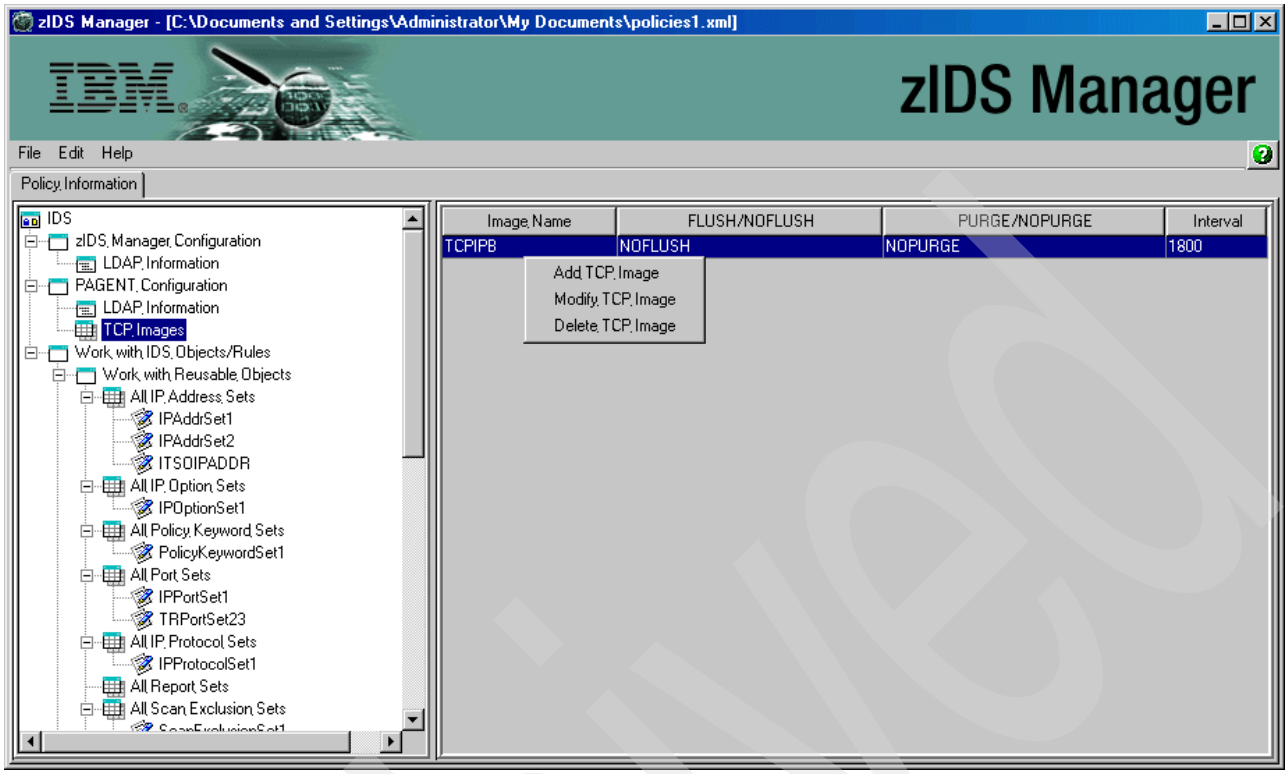


Figure 8-6 Add, modify, or delete TCP/IP image object

At this point we are ready to generate the PAGENT configuration file.

PAGENT configuration file

The PAGENT configuration information must be saved to a text file after providing the necessary information, which we did above. The file is save by selecting **File -> Save As** and then choosing the .conf format, which represents PAGENT LDAP configuration file. An example of the text output is displayed in Figure 8-7.

```
ReadFromDirectory
{
     LDAP_Server                  9.12.6.63
     LDAP_Port                    3389
     LDAP_DistinguishedName       cn=LDAP
     LDAP_Password                secret
     LDAP_SessionPersistent       Yes
     LDAP_ProtocolVersion         3
     LDAP_SchemaVersion           3
     SearchPolicyBaseDN           o=ITSO,c=US
}

TcpImage TCPIPB NOFLUSH NOPURGE 1800
```

Figure 8-7 PAGENT configuration file

This file is the PAGENT configuration file. For more information on the PAGENT configuration file you can reference “The Policy Configuration File” in *z/OS V1R2 Communications Server: IP Configuration Reference*, SC31-8776. This information must be manually transferred (for example, FTPed, cut and pasted, or retyped) to the configuration file located on the zOS

V1R2.0 system. Typically the file is located in /etc/pagent.conf file and is used when the PAGENT application is started.

8.4.3 Work with IDS objects/rules

This section specifies the IDS policy rules. This is the most critical task and typically will be done iteratively until the final policy rules are defined.

- You are required to establish one Condition Set and one action set in at least one policy rule.
- You may optionally specify that rules apply only during validity periods.
- You may optionally associate rules with keywords to speed up their retrieval from LDAP.

Use this section to specify IDS policy rules, which can include Condition Sets, actions, policy keyword sets, or validity periods. Only one policy rule and associated actions can be applied to a particular packet.

The first step in creating a policy rule is to create the reusable objects that will be used in your action and Condition Sets. Next, create the actions and conditions based on the reuseable objects. Finally, build your policy rule from the available condition and action sets. The following sections walk you through this process.

When you have finished specifying IDS policy rules, select **File -> Send to LDAP** to store the policy information into the LDAP Server.

Even before you have finished specifying all the policies, you can save the (possibly incomplete) information that you have entered in an XML file on the workstation running zIDS Manager by selecting **File -> Save**. If you select **File -> Save As** you then have the option to save as an XML, LDIF, or CONF file.

Note: Only XML files can be read back in the zIDS Manager and edited. LDIF and CONF files are generated and are not reuseable by the zIDS Manager.

Work with reuseable objects

The reuseable objects are designed to be incorporated into multiple policies, conditions, or actions, depending on the type of object. Figure 8-8 illustrates the various object sets available in the Work with Reuseable Objects folder. The objects are stored in an object set and the collection of object sets is shown in the folder's with the prefix `All` followed by the object set identifier. For example, all of the unique IP address sets created are located in the All IP Address Sets folder.

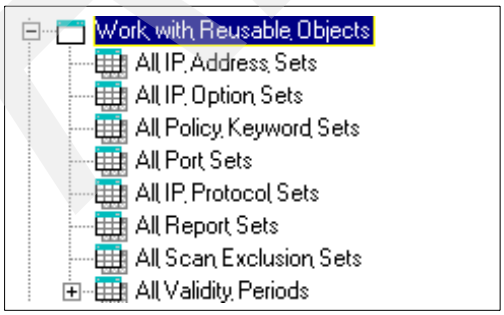


Figure 8-8 Reuseable objects

Let us create an IP address set. First, right-click the **All IP Address Sets** folder. You will see the screen shown in Figure 8-9.

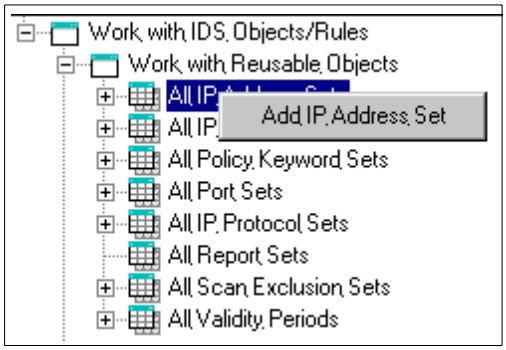


Figure 8-9 Add IP address set

Select **Add IP Address Set**. You will now be prompted for a name of the IP address set. In our example we called the object set ITSOIPADDR, as shown in Figure 8-10. Select **OK** and the new object set has been created. Notice that you can view the object sets either in the left pane by expanding the All IP Address Sets folder or the right pane through the summary information, as shown in Figure 8-11 on page 218.

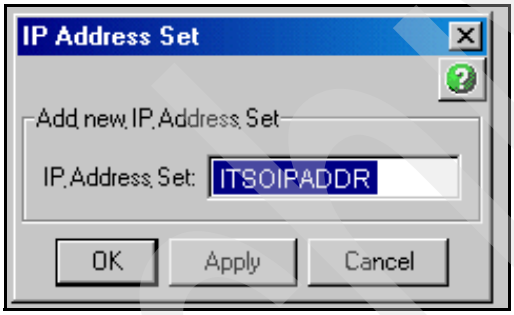


Figure 8-10 IP address set name

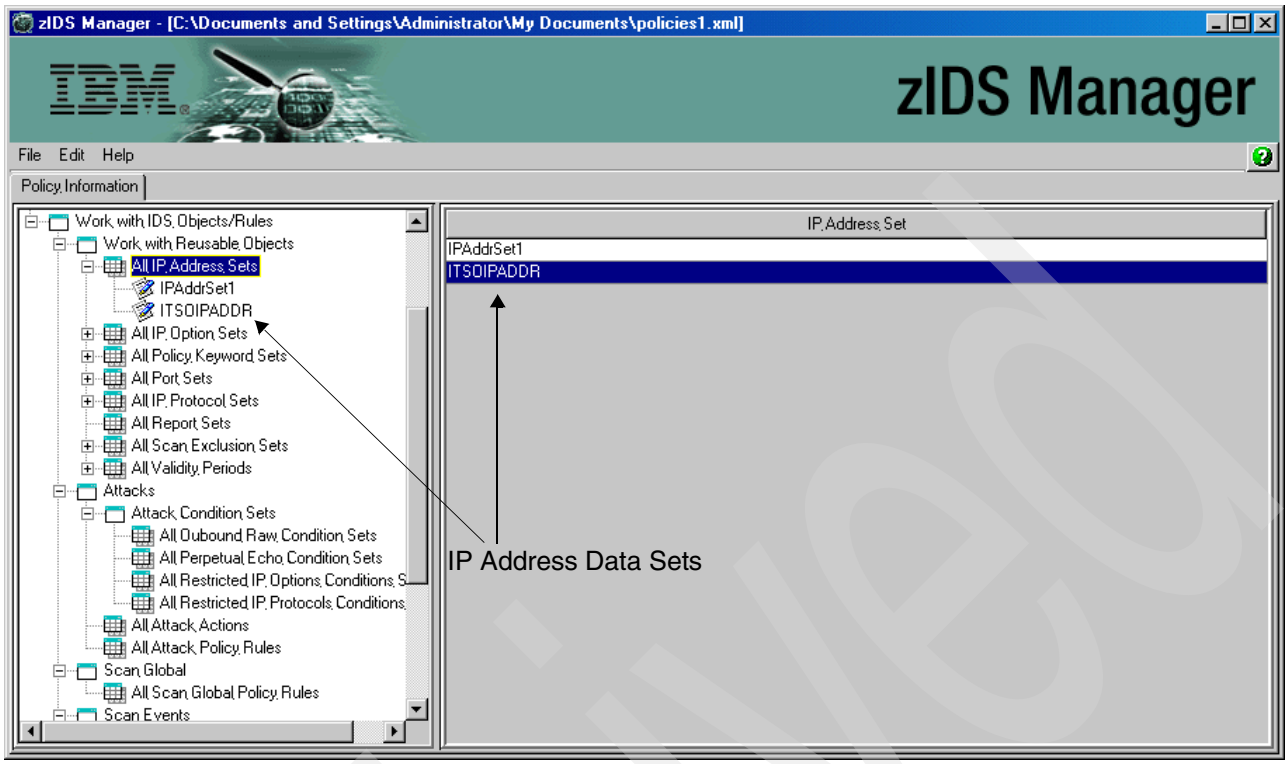


Figure 8-11 IP address sets

In this scenario there was already a predefined IP address set (that is, object set) named IPAddrSet1. Next we want to modify the information that is in the object set ITSOIPADDR.

Click the ITSOIPADDR object set making this the active element in the left pane. The item that is highlighted in the left pane is the active window and dictates the information you see in the right pane, as shown in Figure 8-12 on page 219.

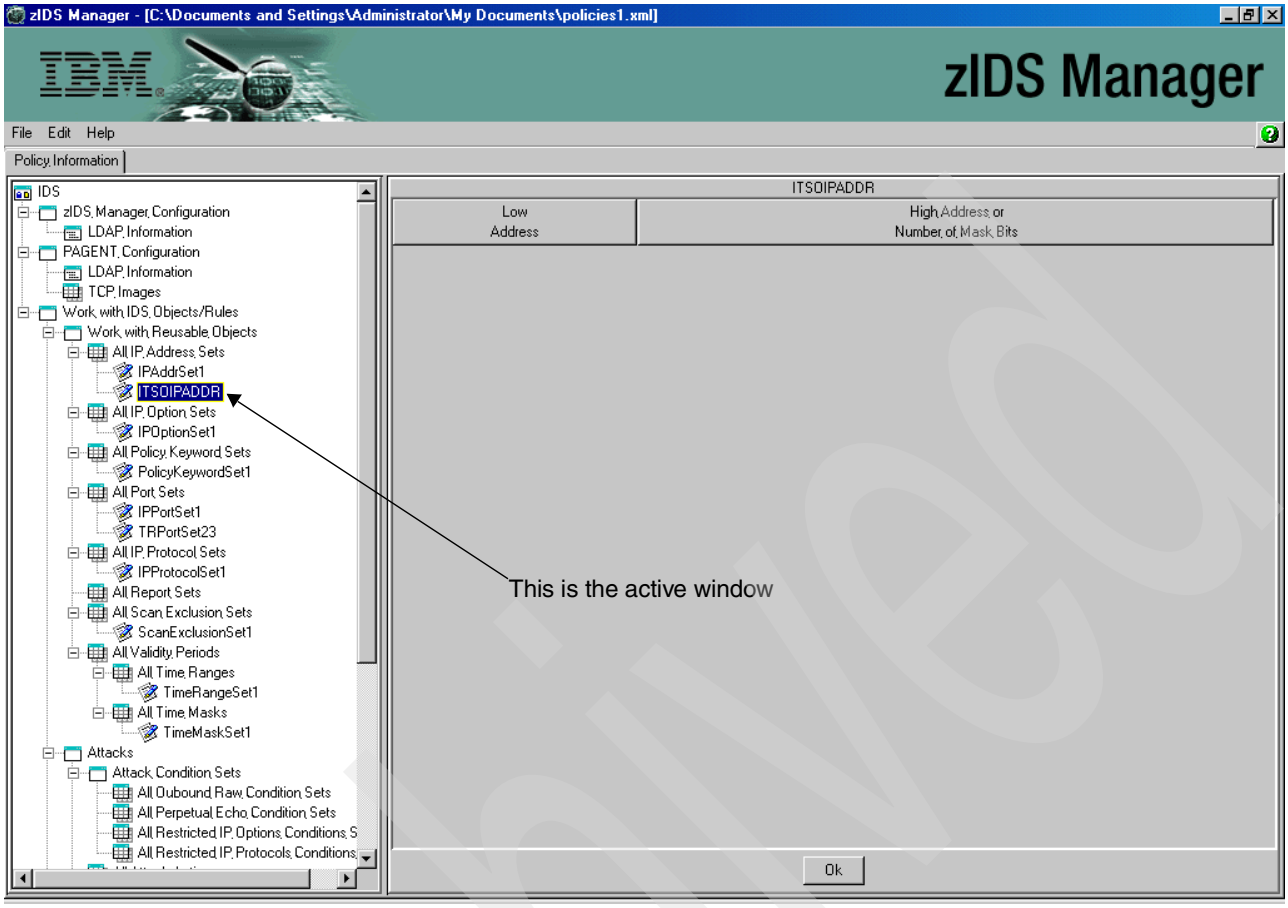


Figure 8-12 IPAddr object set ITSOIPADDR

Now right-click and the pop-up menu appears, as shown in Figure 8-13 on page 220.

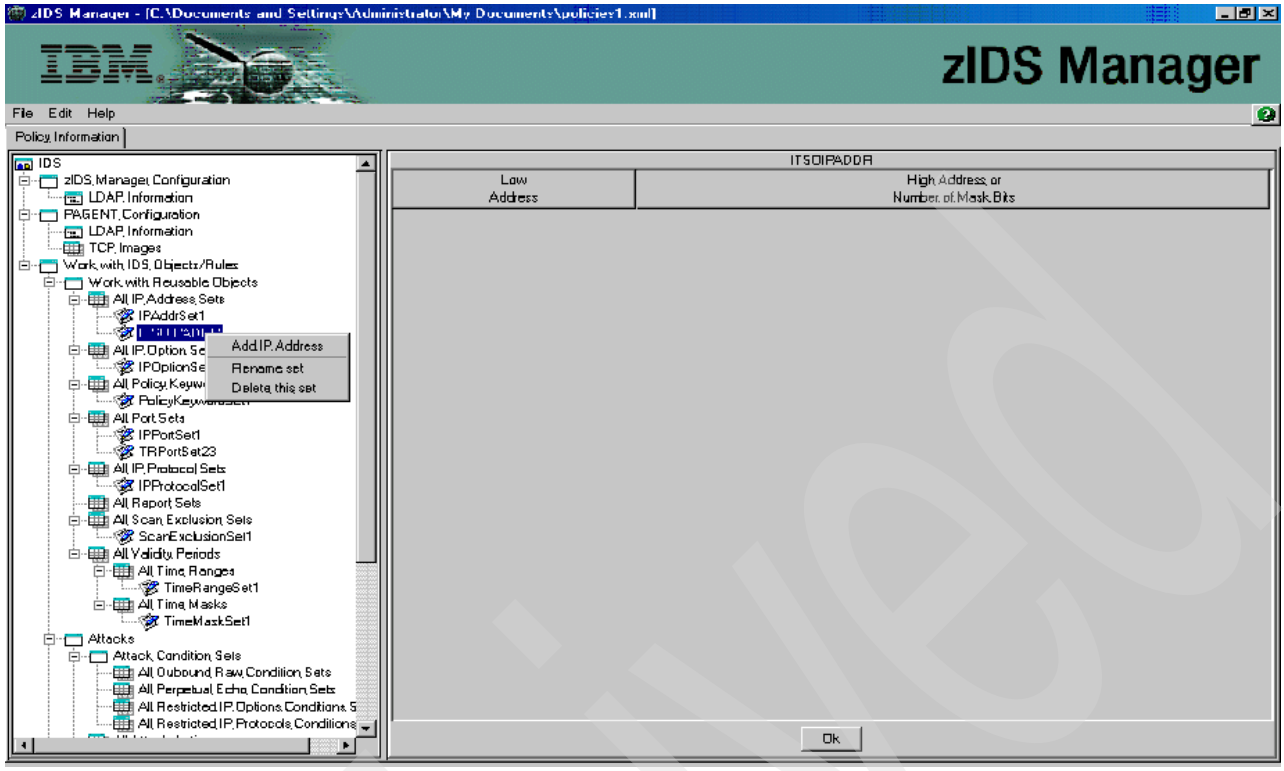


Figure 8-13 Object set options

The available options are:

Add IP address Select this option to enter a low IP address and a high IP address or number of mask bits.

Delete this set Select this option to delete the highlighted IP address set and all IP address ranges in the set. The deleted IP address set will be removed from all associated Condition Sets.

Rename set Select this option to rename the highlighted IP address set. The original IP address set name will be removed from all associated Condition Sets. The new set name will have to be re-associated with the desired Condition Sets.

Select **Add IP Address** and the box shown in Figure 8-14 appears.

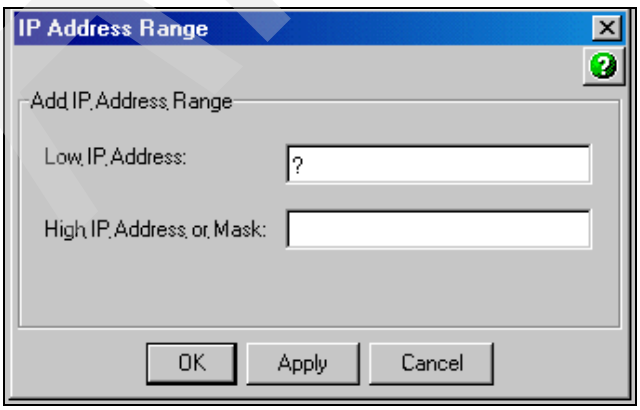


Figure 8-14 IP address range

The question mark indicates that this field requires a value. In our case, we place the value 9.9.9.9 in the Low IP Address field and click **Apply**, followed by another low IP address of 10.10.10.10 and a high IP address or number of mask bits of 24. Select **OK**. We have successfully created a reusable object. The ITSOIPADDR object set contains two objects, as shown in Figure 8-15.

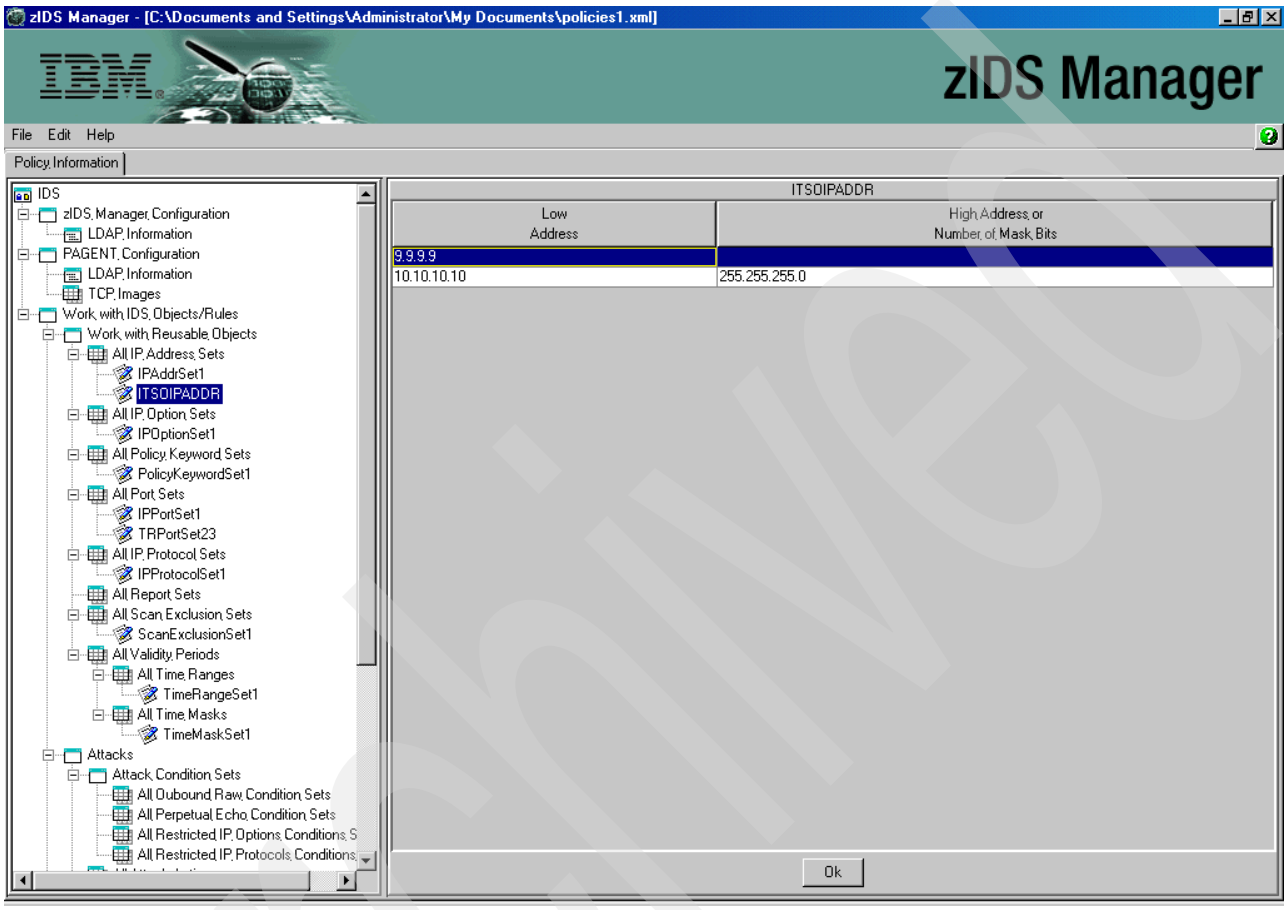


Figure 8-15 Object set summary information

The process by which we built this object set is the same for all of the reuseable objects. Of course, there are different parameters depending on the type of object set being constructed (that is, port sets require a port). We have built several other reusable objects that will be used at a later time as per Figure 8-16 on page 222. It is recommended at this time that the reader create reusable object sets with the same names (that follow) for condition, action, and policy building.

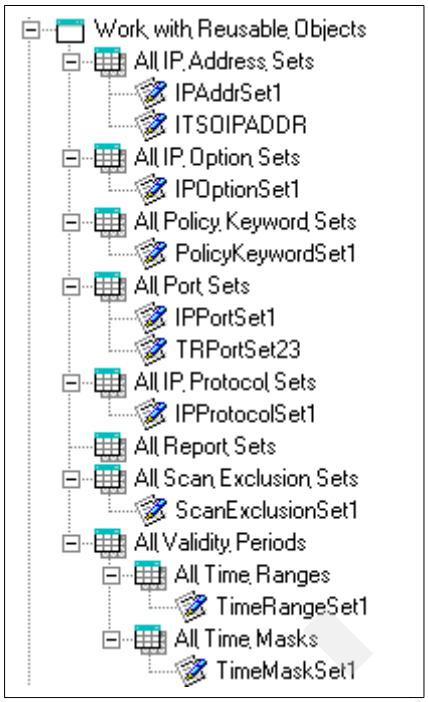


Figure 8-16 Constructed reusable objects

Attacks

An attack can be a single packet designed to crash or hang a system, or multiple packets designed to consume a limited resource causing a network, system, or application to be unavailable to its intended users (a denial of service). IDS attack policy lets you turn on attack detection for one or more categories of attacks independently of each other. In general, the types of actions that you can specify for an attack policy are event logging, statistics gathering, packet tracing, and discarding of attack packets.

The next step is for us to generate an attack condition, action, and policy. Let us start with the condition. Click **Attack Condition Set**, making this the active folder. The folder is open if the + or - next to the folder is a -. You should see the same screen shown in Figure 8-17 on page 223.

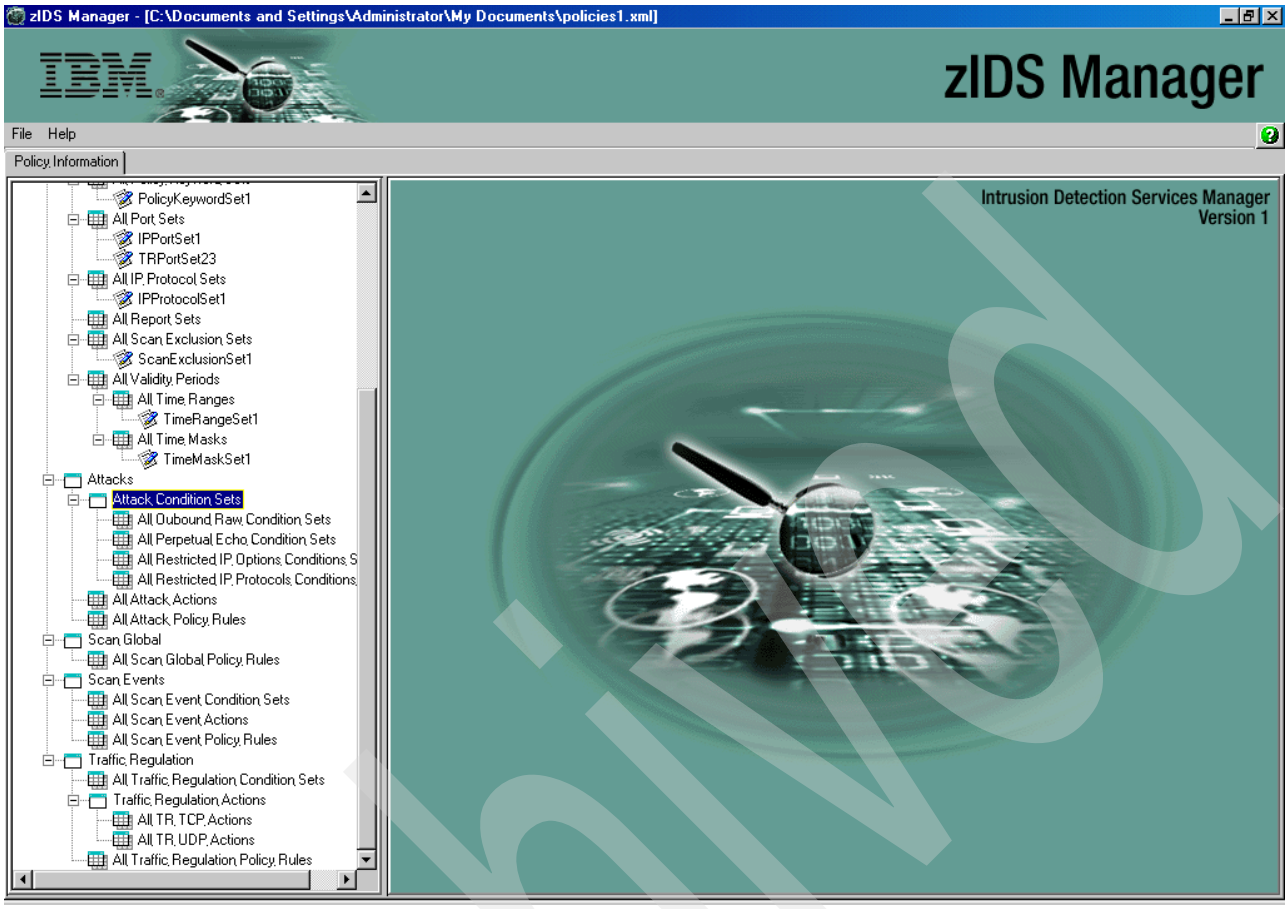


Figure 8-17 Attack Condition Sets

Only four attack types (outbound raw, perpetual echo, restricted IP options, and restricted IP protocols) have Condition Sets for which the user can specify values. Select **All Outbound Raw Condition Sets**, making it active in the left pane, followed by right-clicking to bring up the option to **Add Outbound Raw Condtion Set**. Select this option by clicking it and the menu appears as shown in Figure 8-18.

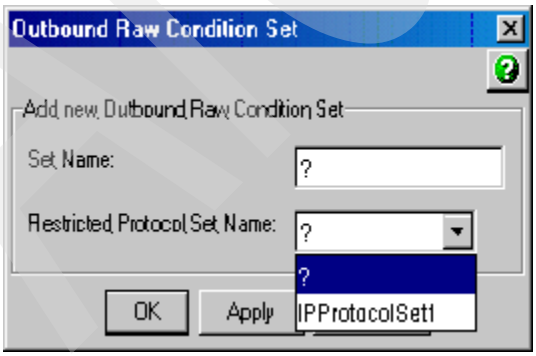


Figure 8-18 Outbound Raw Condition Set

The set name is the name associated with this condition. We chose to use the name RawCondSet1. The **Restricted Protocol Set Name** option presents us with a drop-down menu that includes all of the objects built in the **Reusable Object/All IP Protocol Sets**. Let us select the **IPProtocolSet1** object. Select **OK**. We have generated a Condition Set, as shown in Figure 8-19.

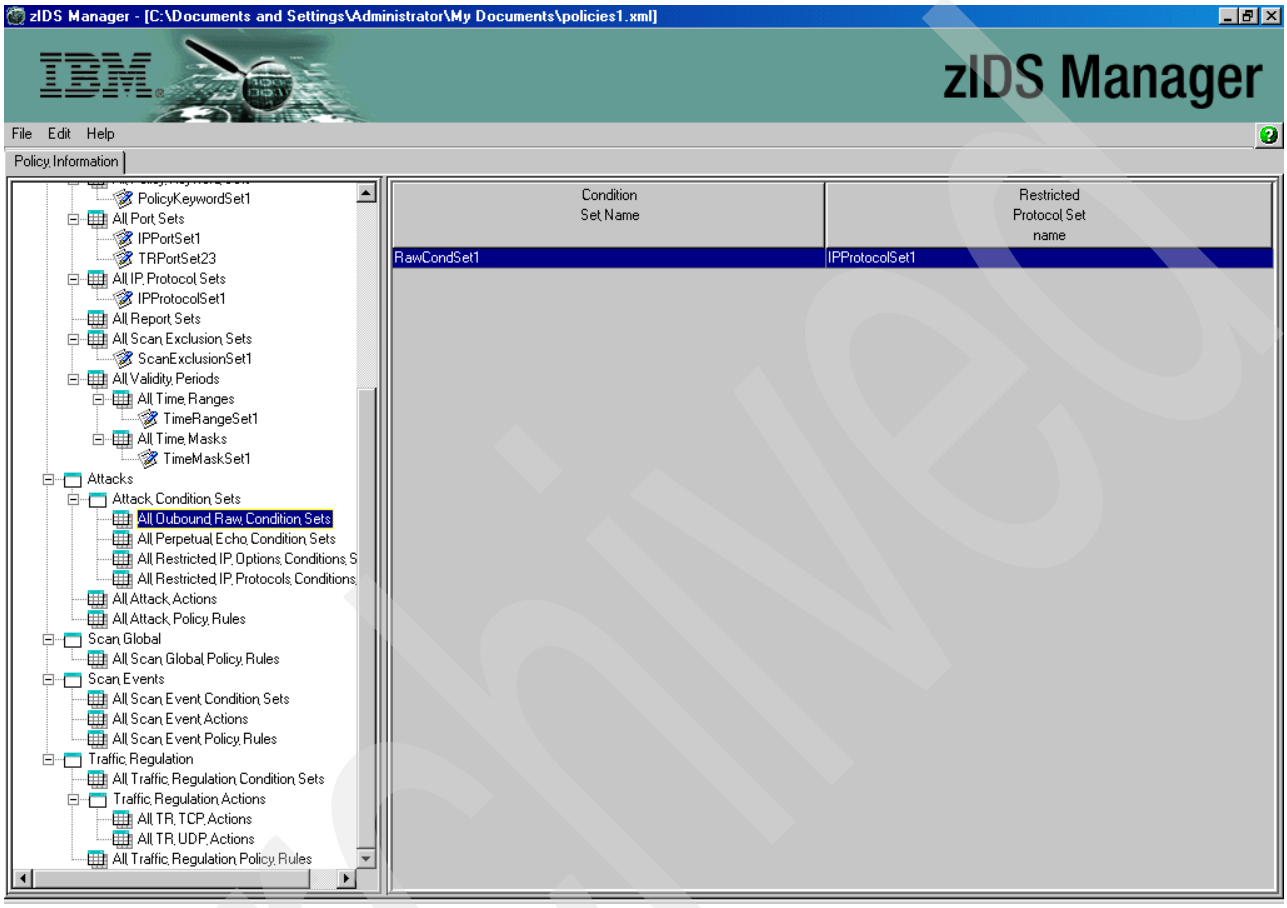


Figure 8-19 Summary of raw Condition Sets

Next select **All Attack Actions** in the left pane; then right-click and select **Add Attack Actions**. The pop-up menu appears as shown in Figure 8-20.

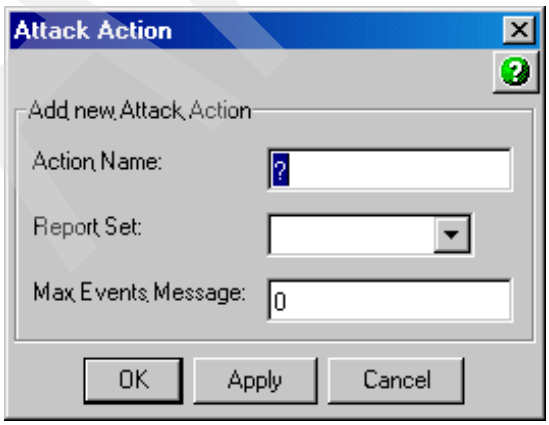


Figure 8-20 Attack action pop-up menu

Action Name is a required field and we chose the name RawAction. **Report Set** is an optional field with a drop-down menu containing the reusable report sets. In our case, we will use IPReportSet1. Select **OK**. There is now attack action built, as shown in Figure 8-21.

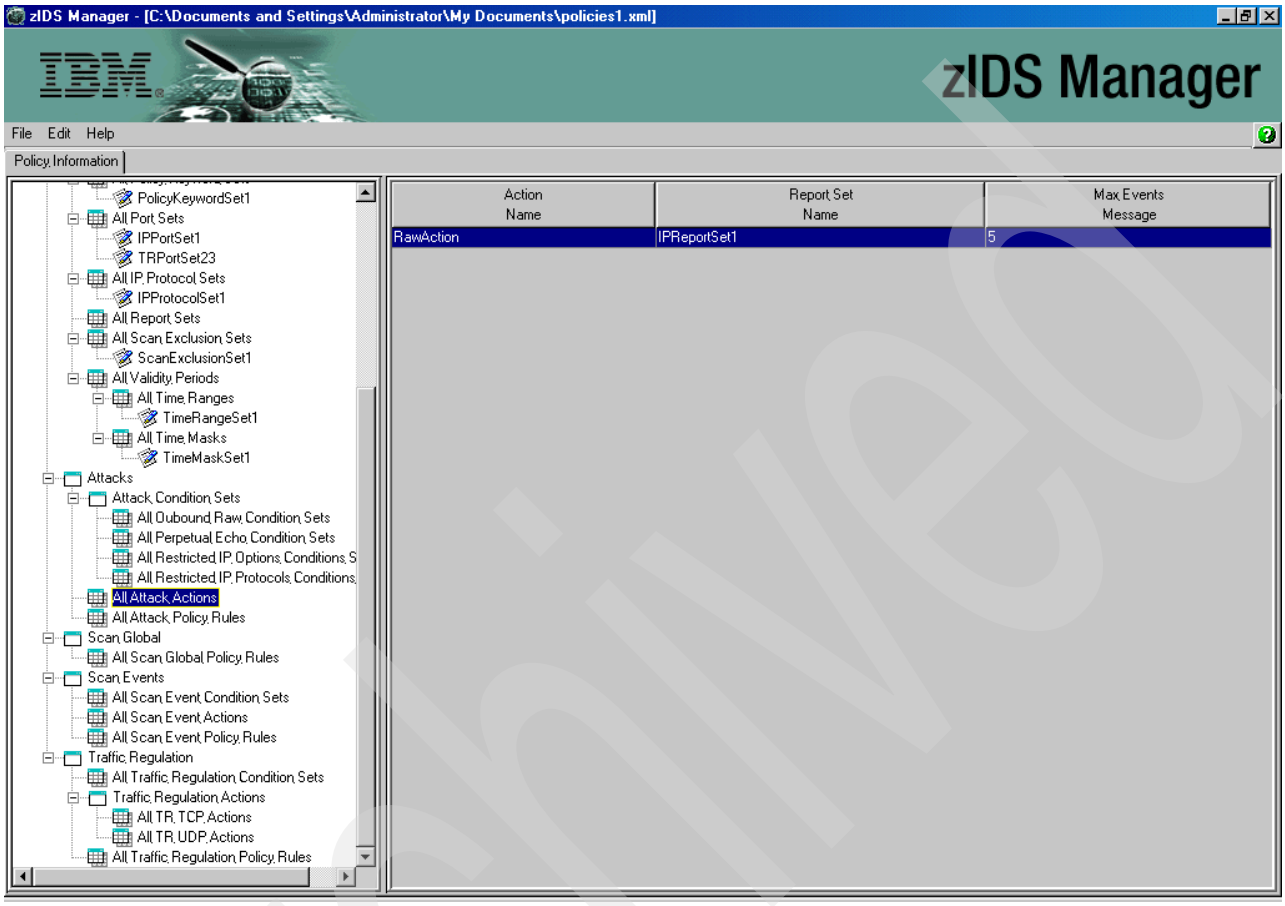


Figure 8-21 Attack action summary

There now needs to be a policy associating a condtion with an action. Select **All Attack Policy Rules** by clicking the folder. Now, right-click and select **Add Attack Policy Rule/Below Section**. The pop-up menu appears as shown in Figure 8-22 on page 226.

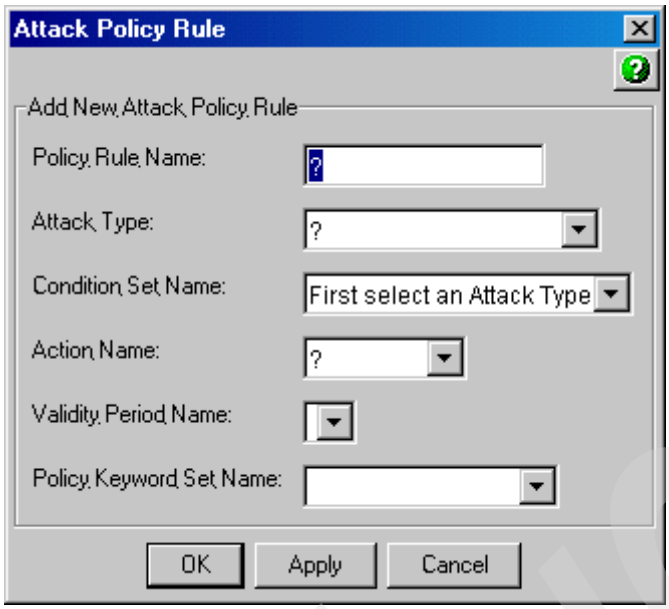


Figure 8-22 Attack policy rule pop-up menu

Policy Rule Name, **Attack Type**, **Condition Set Name**, and **Action Name** are all required fields. We will use RawPolicy, Outbound Raw, RawCondSet1, and RawAction, respectively. **Attack Type** has a drop-down menu. This contains the different attack categories; for information on the attack types or place the cursor in this field and press the F1 key. The **Validity Period Name** and **Policy Keyword Set Name** fields are optional. Other fields and are not used in this example. Select **OK**. An attack policy has been built for outbound raw sockets, as shown in Figure 8-23 on page 227.

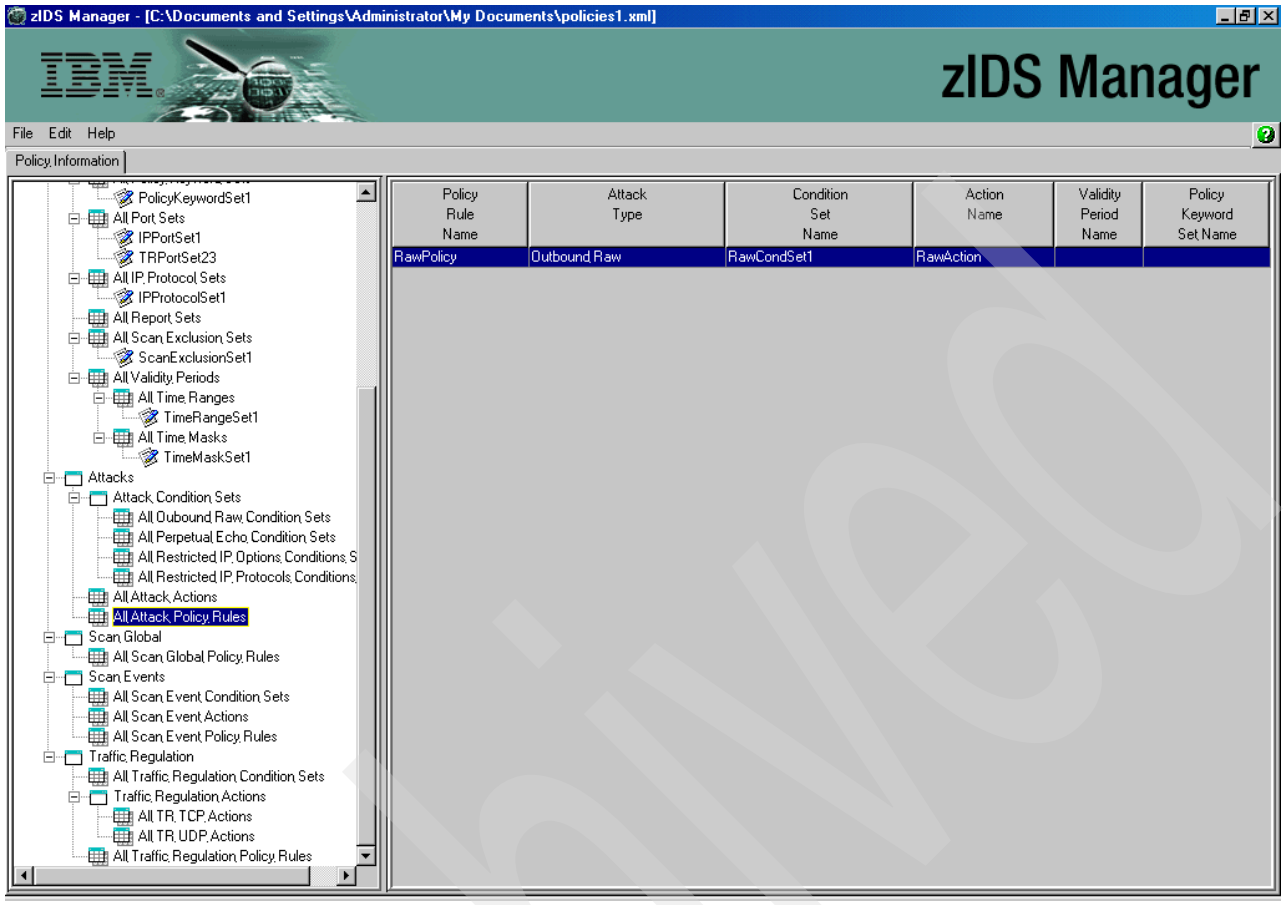


Figure 8-23 Attack policy

Scan global

You can specify sets of global scan detection parameters (threshold and interval for fast and slow scans). These attributes apply to all scan events. If you configure a certain category of scan event, the action will be triggered if the number of those events received from one IP address exceeds the slow scan threshold during the slow scan interval. Similarly, if you configure a certain category of scan event, the action will be triggered if the number of those events received from one IP address exceeds the fast scan threshold during the fast scan interval. The slow scan threshold must be greater than the fast scan threshold. The slow scan interval must be greater than the fast scan interval.

Open the **Scan Global** folder in the left pane. Click **All Scan Global Policy Rules**. Next, right-click and select **Add Scan Global Policy Rule/Below Section**. The menu appears as shown in Figure 8-24 on page 228.

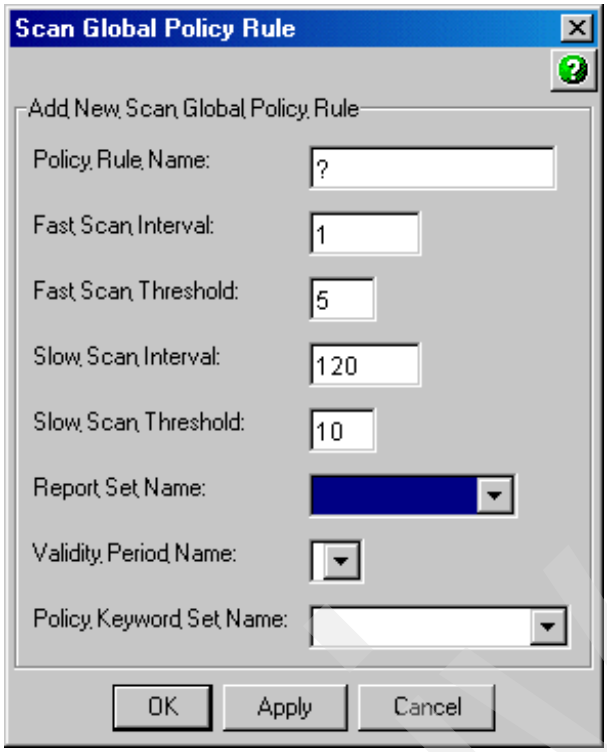


Figure 8-24 Scan global policy rule pop-up menu

As we have already seen, there needs to be a policy name. For the policy rule name we used the name ScanGlobalPolicy1. Notice that there are default values for **Fast Scan Interval**, **Fast Scan Threshold**, **Slow Scan Interval**, and **Slow Scan Threshold** fields. We will accept all of the default values. The **Report Set Name** and **Policy Keyword Set Name** fields have drop-down menus indicating that the choices are defined as reusable objects. We chose IPReportSet1 and PolicyKeywordSet1, respectively. Select **OK**. A scan global policy is now created, as shown in Figure 8-25 on page 229.

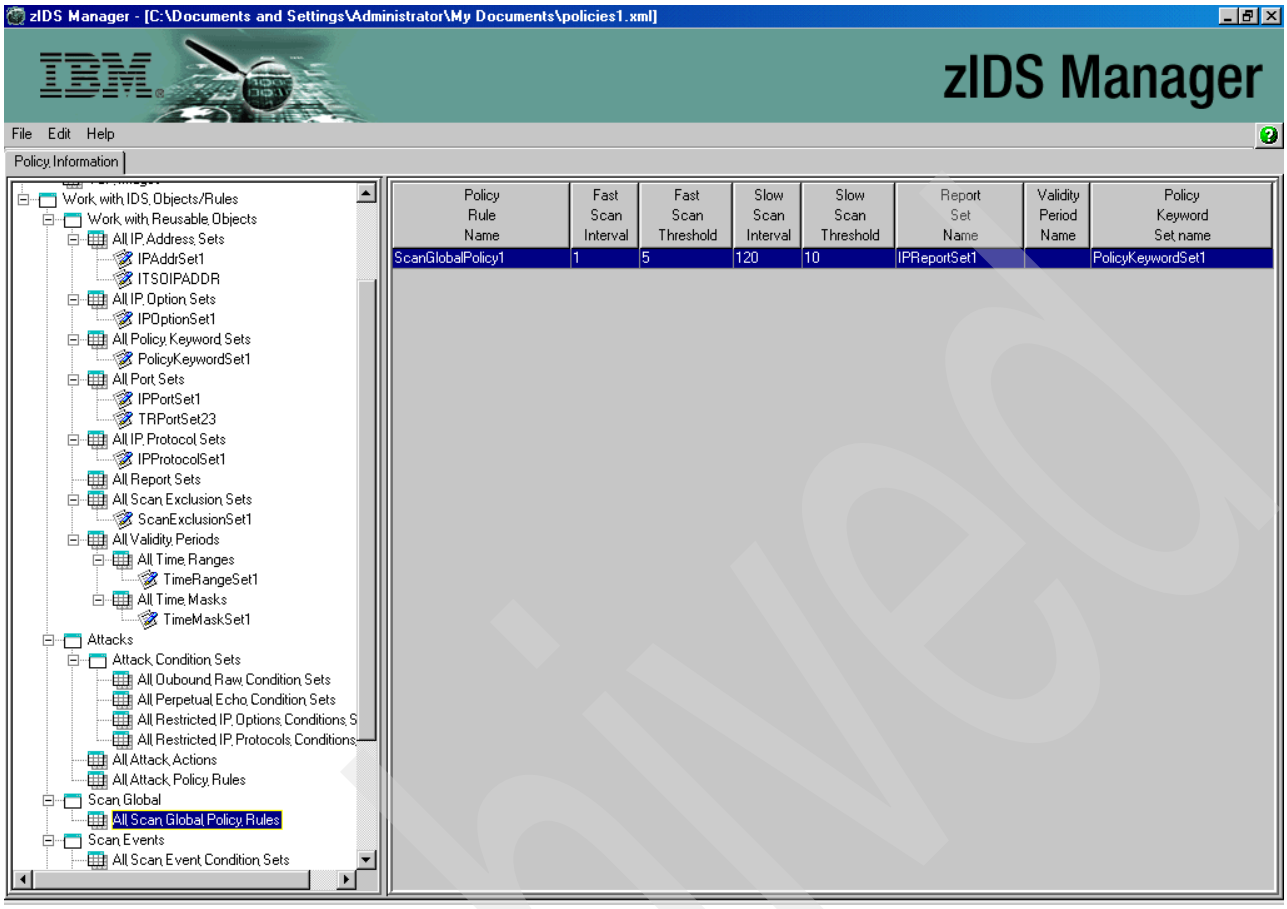


Figure 8-25 Scan global policy

Scan events

Scan events come from the following categories:

- ICMP scans: ICMP requests (echo, information, timestamp, and subnet mask) are used to map network topology. Any request sent to a subnet base or broadcast address will be treated as very suspicious. Echo requests (PING) and timestamp requests are normal unless they include the Record Route or Record Timestamp option, in which case they are possibly suspicious.
- TCP port scans: Because TCP is a stateful protocol, many different events may be classified as normal, suspicious, or highly suspicious. For more details, please see the section "Scan policies" of *OS/390 V2R10.0 IBM Communications Server IP Configuration Guide,* SC-31-8725.
- UDP port scans: A datagram received for a restricted port is very suspicious, one received for an unreserved but unbound port is possibly suspicious, and one received for a bound port is normal.

The individual packets used in a scan can be categorized as normal, possibly suspicious, or very suspicious. To control the performance impact and analysis load of scan monitoring, you can adjust your interest level in potential scan events. If you set the sensitivity level to high, then normal, possibly suspicious, and very suspicious events will be counted. If you set the sensitivity level to medium, then possibly suspicious and very suspicious events will be counted. If you set the sensitivity level to low, then only very suspicious events will be counted. If you set the sensitivity level to none, then no events will be counted.

First open the **Scan Events** folder. Click **All Scan Event Conditions Sets** to activate. Right-click and select **Add Scan Event Condtion Set** (see Figure 8-26).

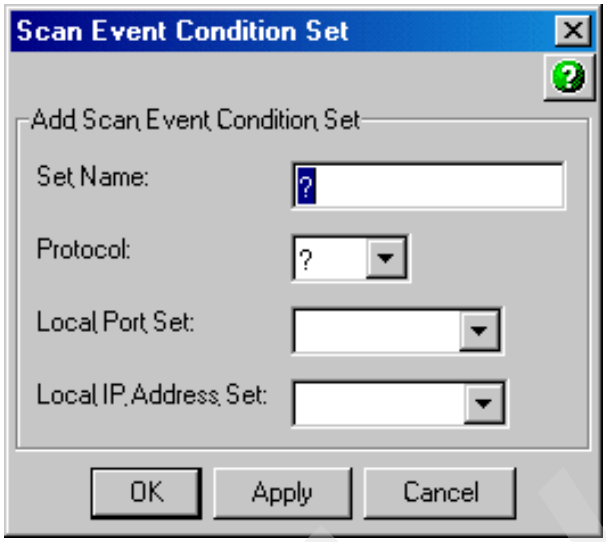


Figure 8-26 Scan event Condition Set pop-up menu

The required fields are **Set Name** and **Protocol**. We will use ScanEventCondition1 and TCP, respectively. The **Local Port Set** and **Local IP Address Set** fields have drop-down menus that contain reusable objects. For these fields we will use the reusable objects IPPortSet1 and IPAddrSet1. Next select **OK** and we have created a condtion set. Now we need an action and, as with the Attack section, a policy to relate the condtion to an action. Click **All Scan Event Actions** in the left pane, making it the active element. Right-click and select **Add Scan Event Action**. The pop-up menu appears, as shown in Figure 8-27.

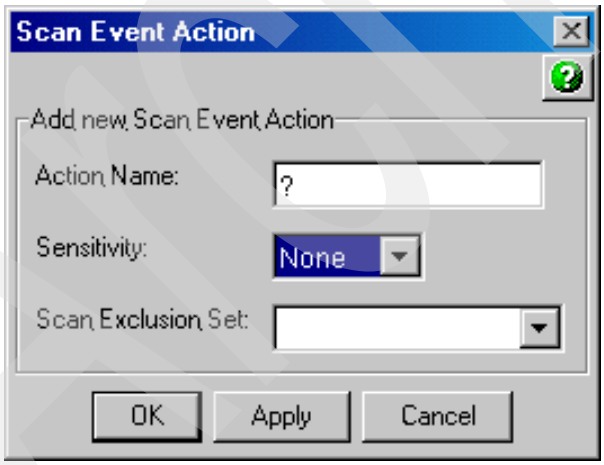


Figure 8-27 Scan event action pop-up menu

Action Name is a required field and our action is called ScanEventAction1. The sensitivity defaults to None. This will cause no events to be counted toward the scan event. We chose a low sensitivity. For more information on the sensitivity, place your cursor in the **Sensitivity** field and press the F1 key. **Scan Exclusion Set** is optional, and for this action we will not code a value. Select **OK** to complete the action.

Next click **All Scan Event Policy Rules**, making this the active folder. Right-click and choose **Add Scan Event Policy Rule -> Below Section**. The pop-up menu appears as shown in Figure 8-28.

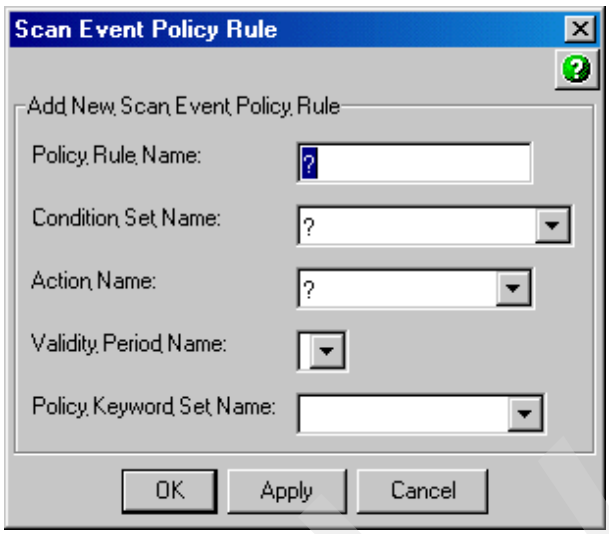


Figure 8-28 Scan event policy rule pop-up menu

The required fields are **Policy Rule Name**, **Condition Set Name**, and **Action Name**. We will use ScanEventPolicy1, ScanEventCondition1, and ScanEventAction1, respectively. **Validty Period Name** and **Policy Keyword Name** are optional fields that will be left blank. Select **OK** and our policy is now complete, as shown in Figure 8-29 on page 232. To modify the policy, you can double-click the policy in the right pane.

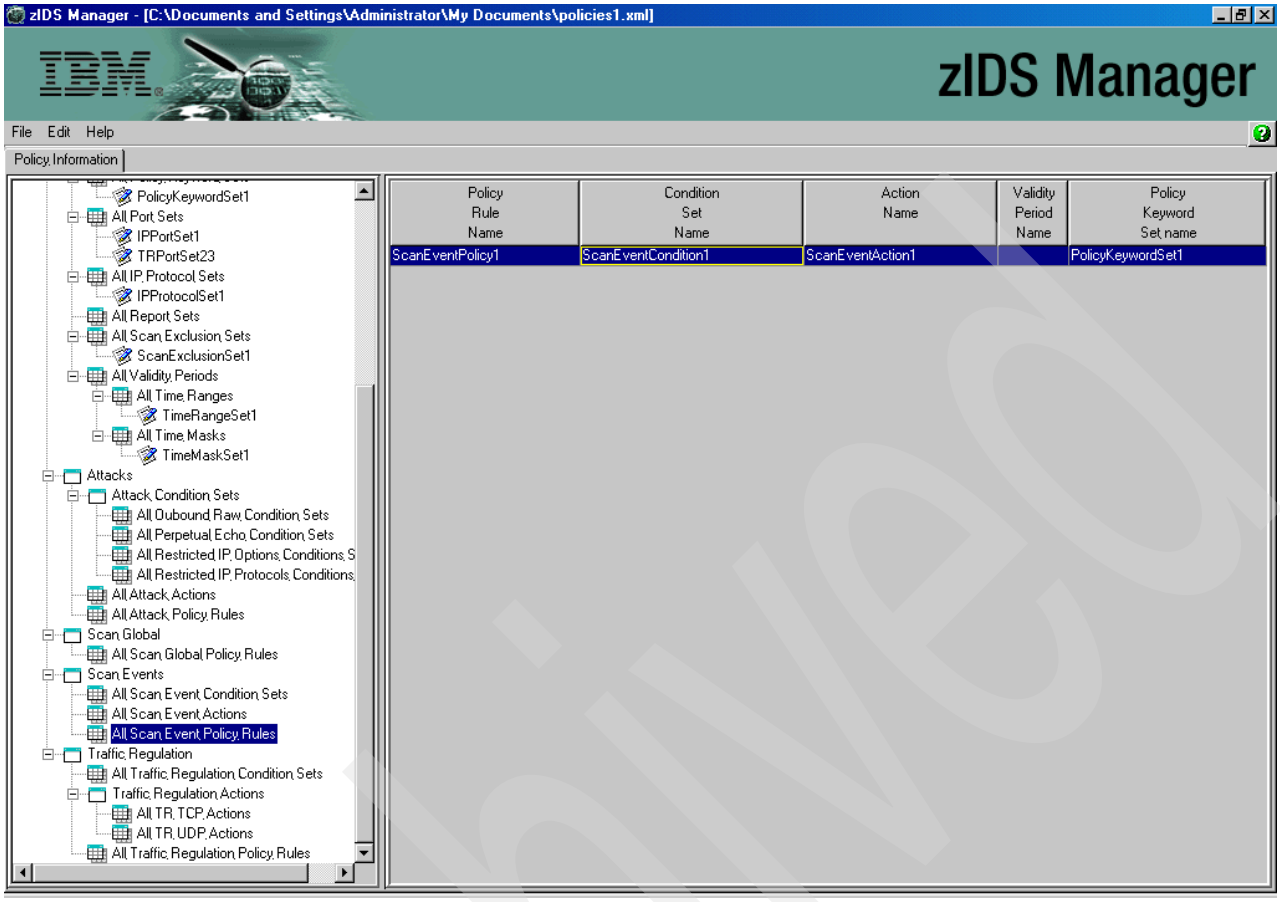


Figure 8-29 Scan event policy rule

Traffic Regulation

Traffic Regulation (TR) policies are used to limit memory resource consumption and queue delay during peak loads. TR policies for TCP ports can limit the total number of connections an application has active at one time. This can be used to limit the number of address spaces created by forking applications such as FTPD and otelnetd. A fair share algorithm is also provided based on the percentage of remaining available connections held by a source IP address. IDS policies for UDP ports specify a queue length. Longer queues let applications with higher processing rate capacity absorb higher bursts of traffic. Shorter queues let applications with lower processing rate capacity reduce the queue delay time of packets that they accept.

First open the **Traffic Regulation** folder. Click **All Traffic Regulation Condition Sets** to activate it. Right-click and select **Add Traffic Regulation Condition Set** (see Figure 8-30 on page 233).

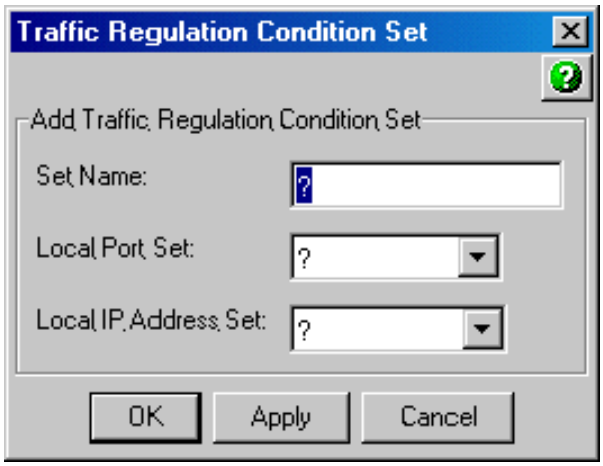


Figure 8-30 Traffic Regulation Condition Set pop-up menu

All of the fields are required as per the question marks. The set name is TRREGCondSet1. **Local Port Set** and **Local IP Address Set** have drop-down menus and we will use the TRPPortSet23 and IPAddrSet1 reuseable objects, respectively. Select **OK** and the conditions set is complete. Now click **All TR TCP Actions** (we will generate a TCP action) in the left pane, making it the active element. Right-click and select **Add TR TCP Action**. The pop-up menu appears as shown in Figure 8-31.

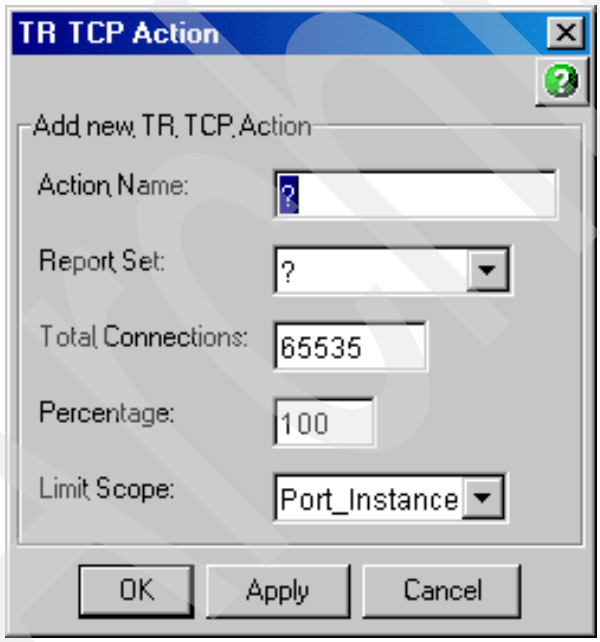


Figure 8-31 TR TCP action pop-up menu

Action Name is a required field and our action is called TRTCPAction1. **Report Set** is also required and we will use our reusable object report set TRReportSet1. We accept the default values of 65535, 100, and Port_Instance for **Total Connections, Percentage**, and **Limit Scope**, respectively. Finally, we must correlate the condtion with the action through a policy.

Next click **All Traffic Regulation Policy Rules** followed by a right-click. Now choose **Add Traffic Regulation Policy Rule -> Below Selection**. The pop-up menu appears as shown in Figure 8-32 on page 234.

Figure 8-32 Traffic Regulation policy rule pop-up menu

The required fields are **Policy Rule Name**, **Conditions Set Name**, and **Action Name**. For **Policy Rule Name** we will use TRPolicyTelnet. **Condition Set Name** and **Action Name** are drop-down menus and we will use our reuseable objects, TRRegCondSet1 and TRTCPAction1, respectively. **Validty Period Name** and **Policy Keyword Name** are optional fields that will be left blank. Select **OK** and our policy is now complete, as shown in Figure 8-33 on page 235. To modify the policy, you can double-click the policy in the right pane.

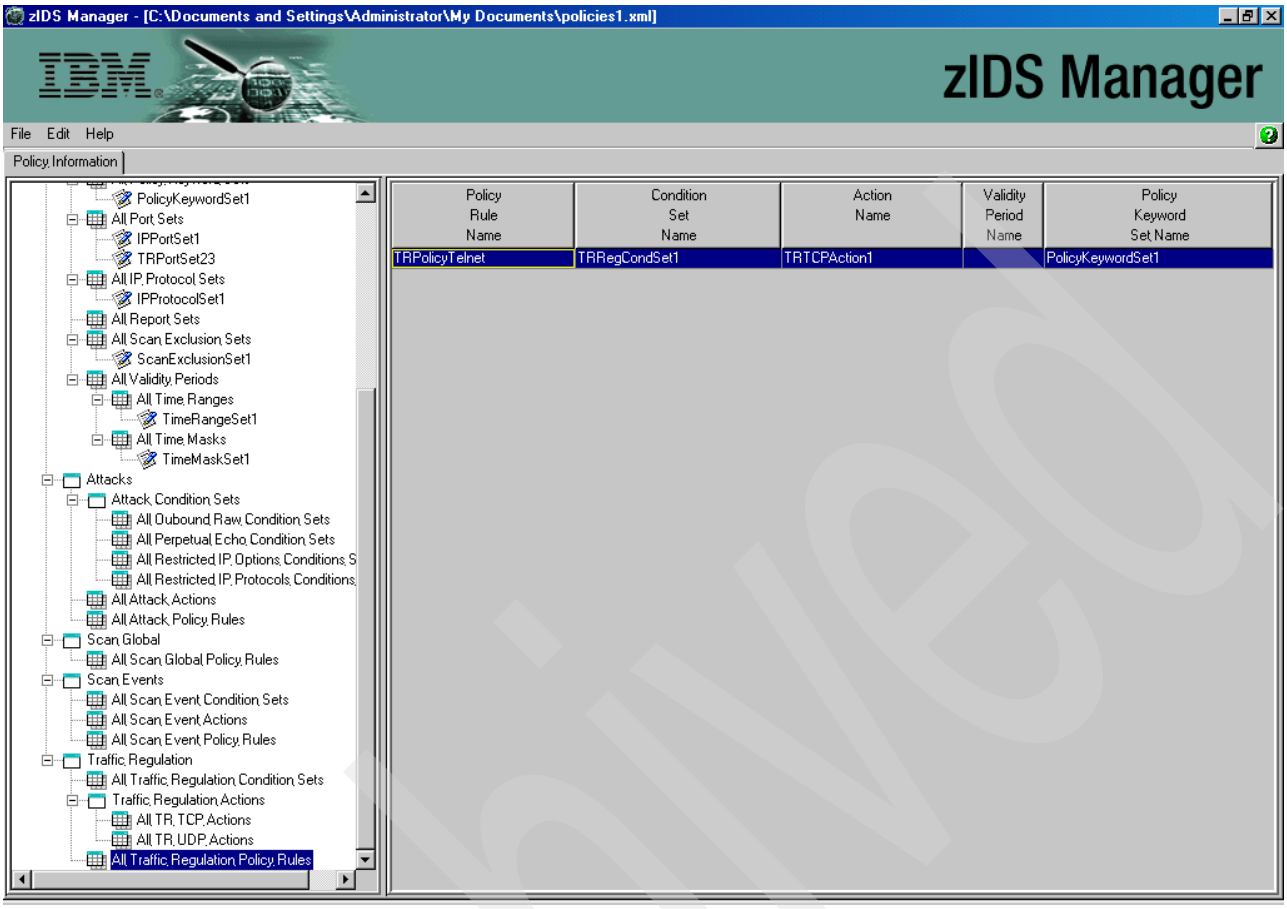


Figure 8-33 TR TCP policy

8.5 Policy priorities

Policies consist of several related objects. The main object is the policy rule. A policy rule object refers to one policy condition and one policy action with optional validity periods and policy keywords. Policy objects are analogous to an IF/THEN statement in a program. For example:

```
IF condition THEN action
```

In other words, when the set of conditions referred to by a policy rule is TRUE, then the policy actions associated with the policy rule are executed. Only one policy rule and associated action can be applied to a particular packet. The priortization of the policy can be seen when you add a policy and receive the **Above Selection/Below Selection** pop-up menu, as shown in Figure 8-34 on page 236.

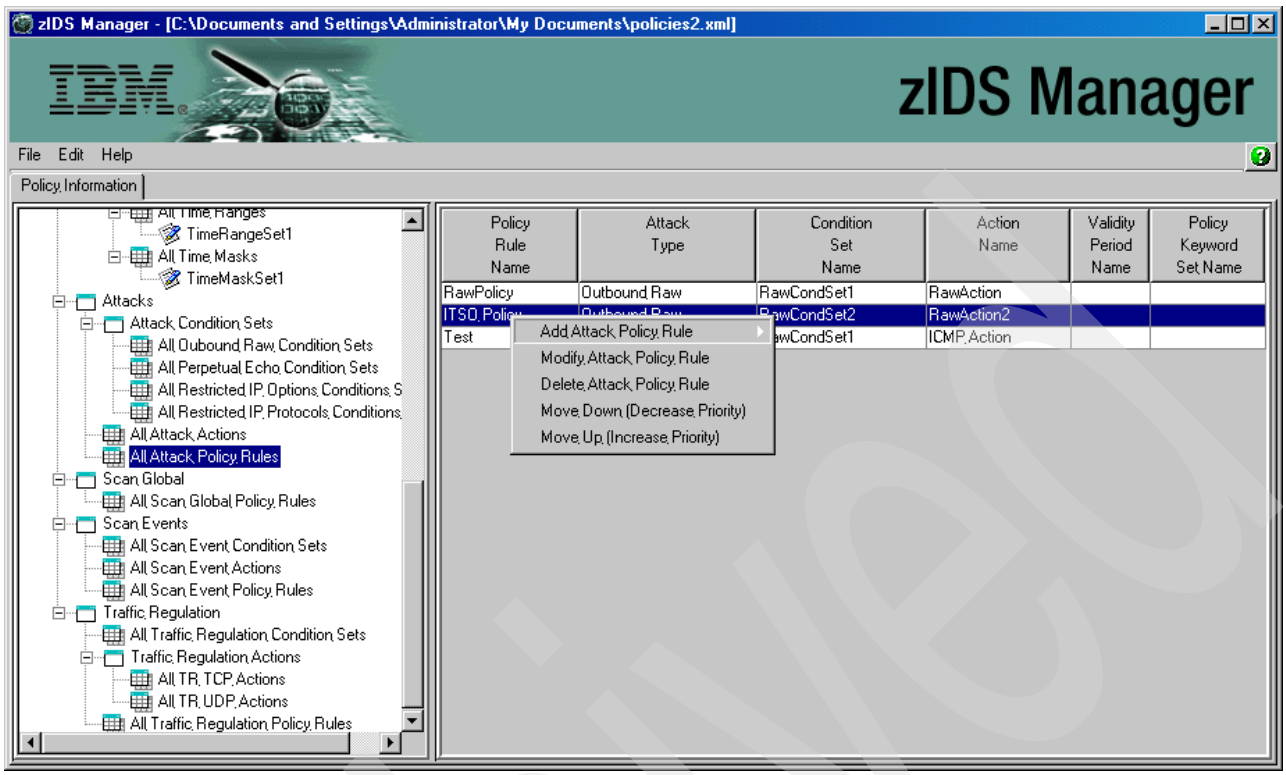


Figure 8-34 Policy priortizaiton

The first policy in the policy rule table with a TRUE condition will be executed. Thus, the prioritization of policies must be evaluated prior to implementation. One can easily prioritize a policy by clicking a policy in the right-hand pane and when the user chooses to add a policy, they are prompted with a specification for above/below the current policy. Once a policy is placed in the table of policy of rule, the user has the ability to move the policy based on the priortization. This is done by highlighting an existing policy in the right-hand pane and then right-clicking the policy. Figure 8-35 on page 237 illustrates the available options for an existing policy in policy rule table.

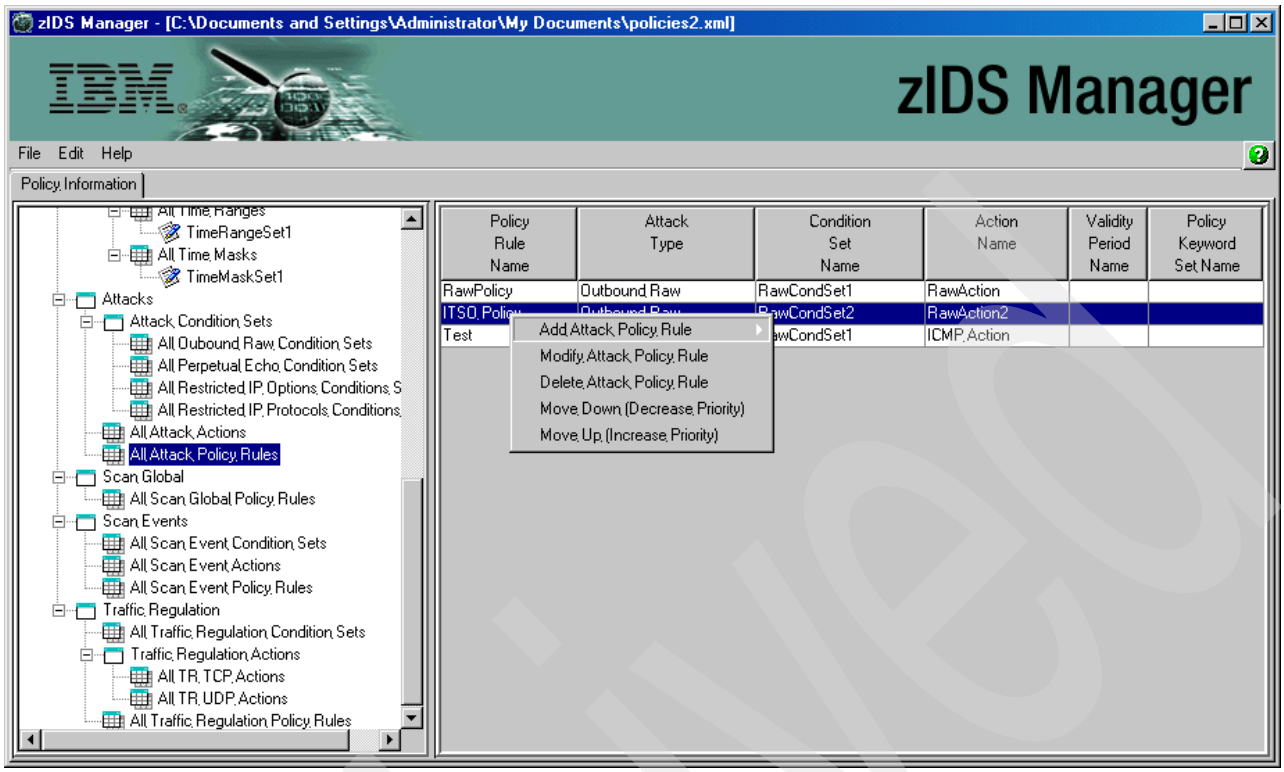


Figure 8-35 Policy options within the policy rule table

8.5.1 Conjunctive Normal Form (CNF) policies

The zIDS Manager only supports the Conjunctive Normal Form, which means an ANDed (different condition levels) set of ORed conditions (same condition level). This ORing of the same level conditions can be seen in Figure 8-36 on page 238.

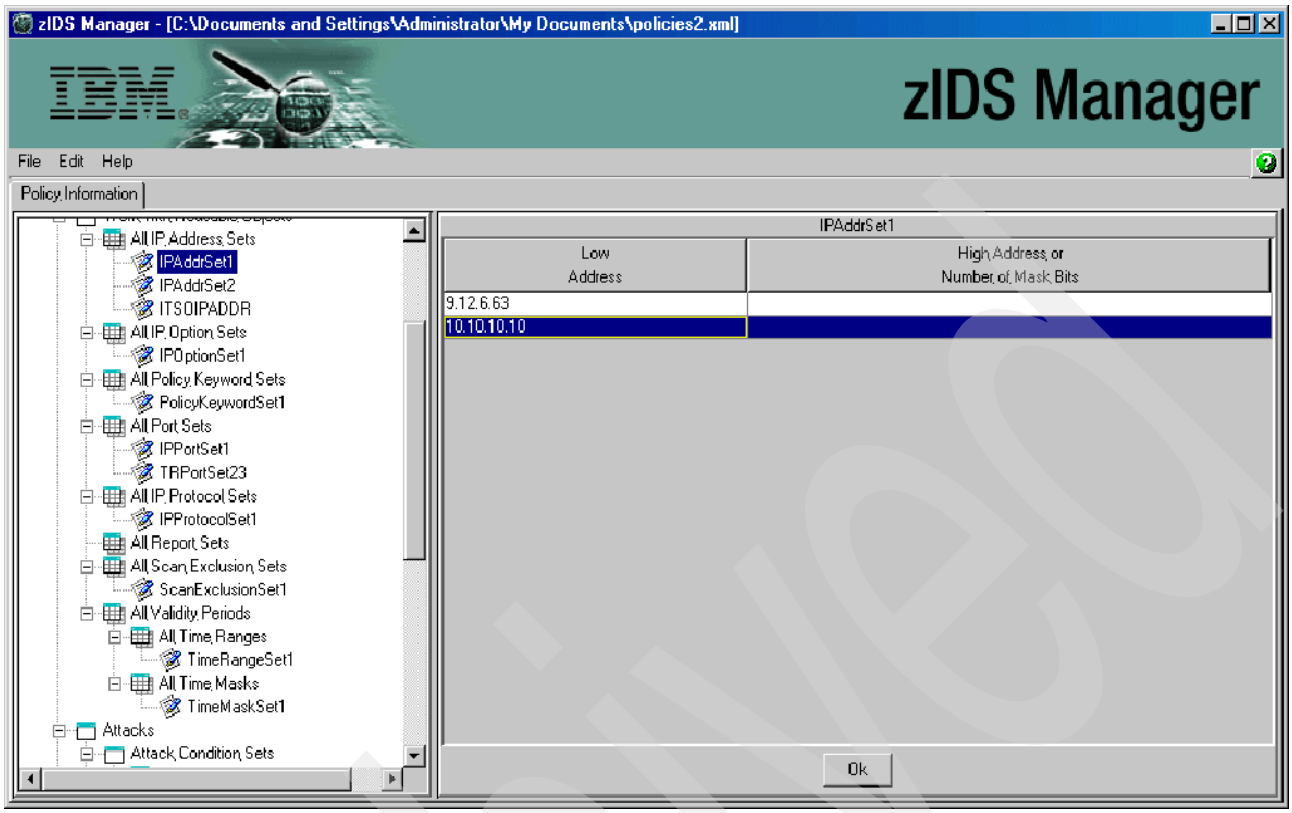


Figure 8-36 CNF ORed condition

The information in IPAddrSet1 is all at the same level. Thus, when evaluated in a packet, this information will be ORed. This can be viewed as:

```
IF IP Address 9.12.6.63 OR IP Address 10.10.10.10 THEN Action
```

Now let us look at a Condition Set with multiple condition levels, which requires the AND function (see Figure 8-37 on page 239).

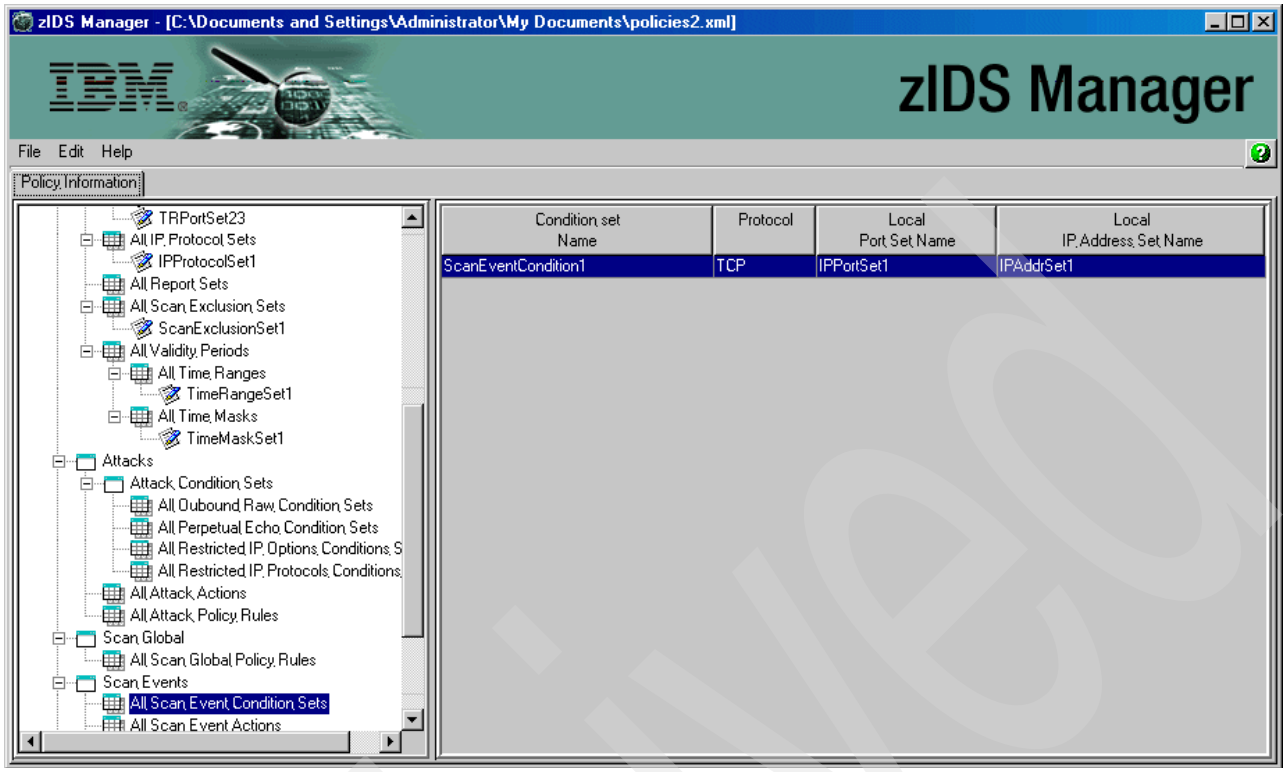


Figure 8-37 zIDS ANDed condition

Using CNF, ScanEventConditionSet1 reads as:

```
IF TCP AND IPPortSet1 AND IPAddrSet1 THEN Action
```

Where:

- TCP = TCP Protocol
- IPPortSet1 = Port 20 OR Port 21
- IPAddrSet1 = 9.12.6.63 OR 10.10.10.10

Thus making the appropriate substitutions we have:

```
If (TCP Protocol) AND (Port 20 OR Port 21) AND (9.12.6.63 OR 10.10.10.10) THEN Action
```

Now we create the action ScanEventAction1 and build the policy ScanEventPolicy1 that associates the two:

```
ScanEventPolicy1:
If (TCP Protocol) AND (Port 20 OR Port 21) AND (9.12.6.63 OR 10.10.10.10) THEN
ScanEventAction1
```

This is shown in graphical form in Figure 8-38 on page 240.

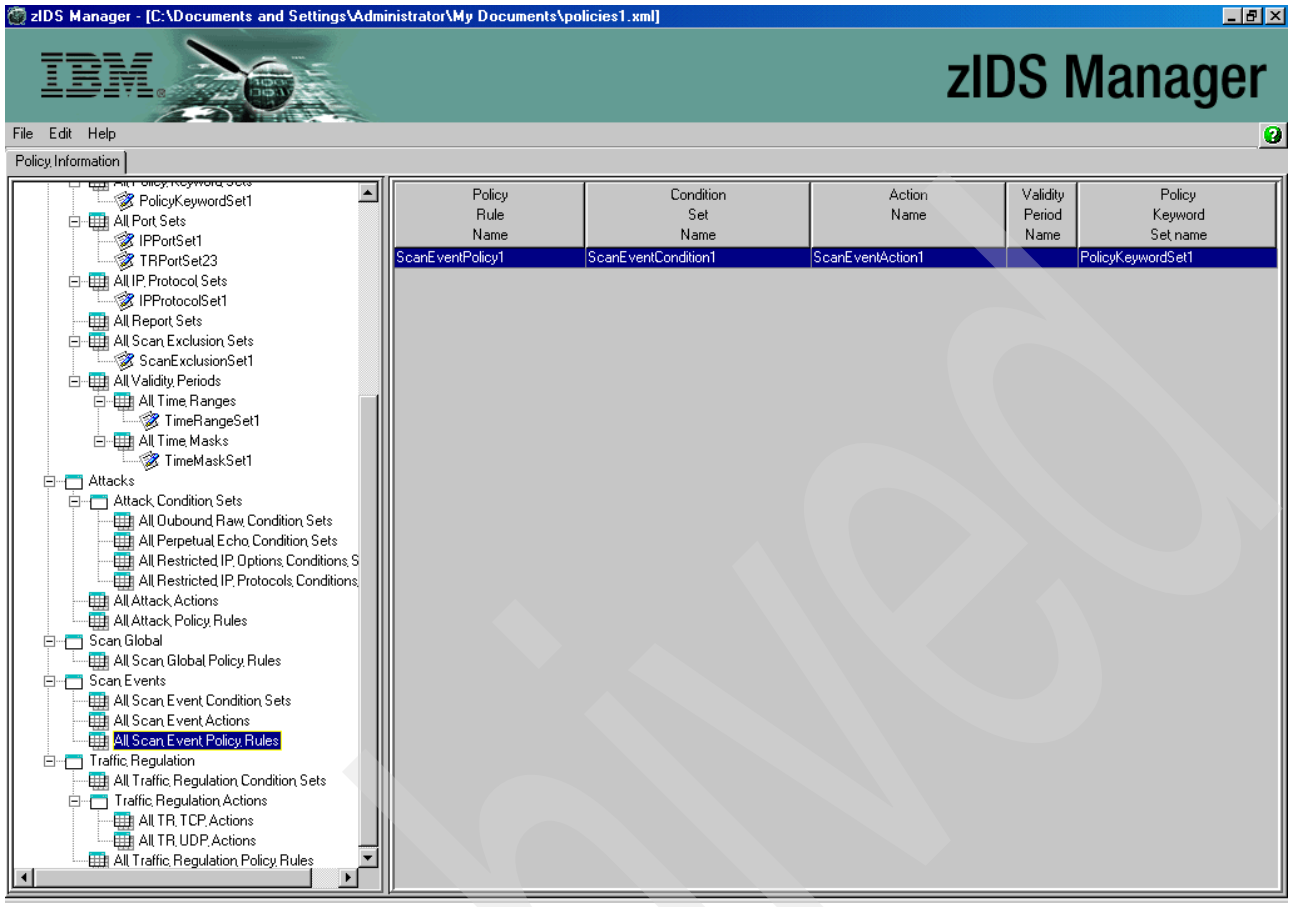


Figure 8-38 zIDS graphical representation of the policy

8.6 Additional information

In this section we provide some additional information including a summary of limitations, common mistakes, and logging.

8.6.1 Limitations

The zIDS Manager TCP/IP connection with the LDAP Server cannot be a secure connection, whereas the connection between PAGENT and the LDAP Server can be secure. IBM has been notified of this limitation via the newsgroup.

8.6.2 Common mistakes

While incorporating our policies into the zIDS Manager, we experienced problems that may be common to others.

- All Traffic Regulation Condition Sets - local values. We coded this value as the local adapter or DVIPA address we wanted to regulate the traffic on. This is not the intent of this field. The intent is discussed in “All Traffic Regulation conditions sets - local values” on page 241.
- All report sets - logging level. We did not know what to code for this field and the Help menus did not provide the information we needed to set this value.

All Traffic Regulation conditions sets - local values

Figure 8-39 is a TR Condition Set; notice the **Local Port Set Name** and **Local IP Address Set Name**. These represent reuseable objects that are based on the port and IP address that were passed in the bind() socket call. We encountered a problem when we used the IP address of the local interface we were using. We coded this in our IP address set name data set and the TR policy was not rejecting the connections. We then realized that the TN3270 listener had bound to port 23 with the IP address of INADDRANY (0.0.0.0). Once the change was made to the object set, the policy was effective.

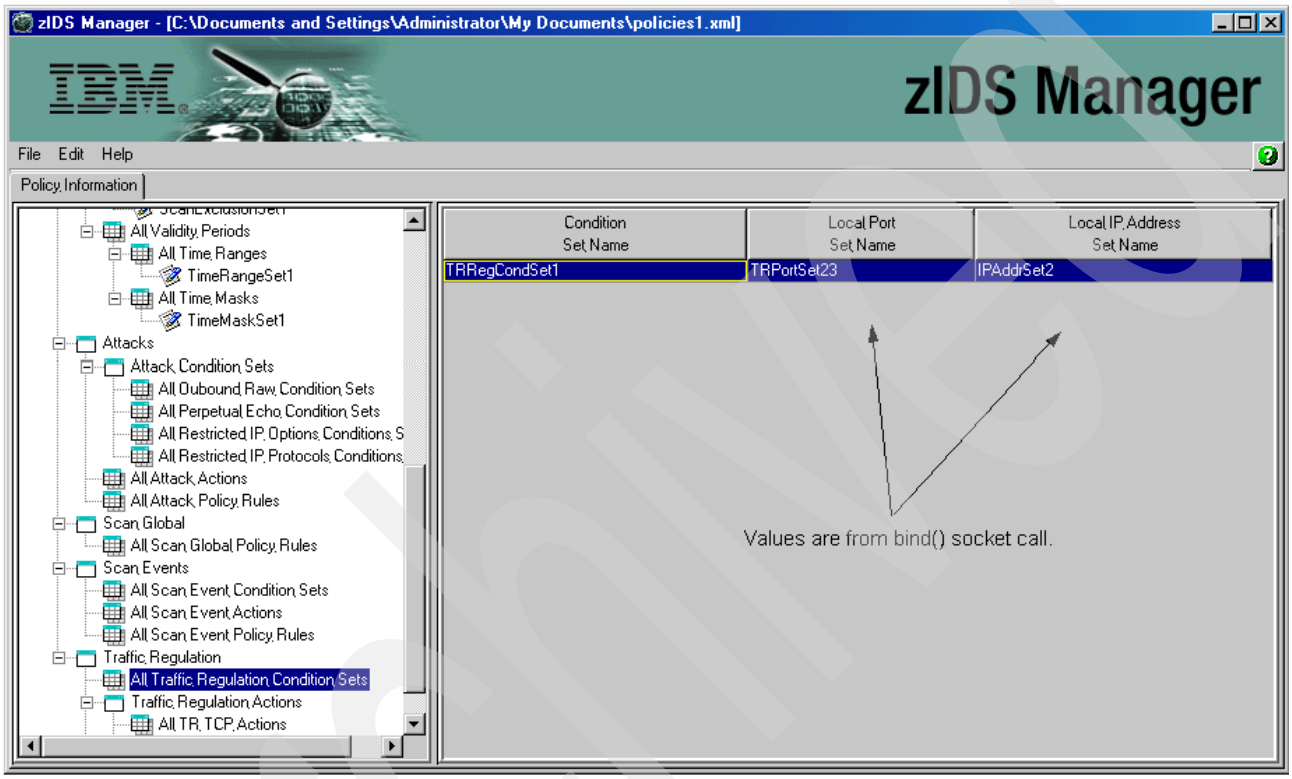


Figure 8-39 TR Condition Set values

All report sets - logging level

The **Logging Level** field represents the syslogd value for the *priority code*. Syslogd receives the information from TRMD and processes the information based on the *facility name*, which is daemon, and the priority code specified on the call. The help screen refers you to *OS/390 V2R10.0 C/C++ Run-Time Library Reference,* SC28-1663, however, no numeric values are supplied for the priority code argument. The definitions for priority code values are as follows:

LOG_EMERG	A Panic condition. This is normally broadcast to all processes.
LOG_ALERT	A condition that should be corrected immediately, such as a corrupted system database.
LOG_CRIT	Critical conditions, such as hard device errors.
LOG_ERR	Errors.
LOG_WARNING	Warning messages.
LOG_NOTICE	Conditions that are not error conditions, but that may require special handling.
LOG_INFO	Informational messages.

LOG_DEBUG Messages that contain information normally of use only when debugging a program.

Table 8-1 lists the corresponding numeric values needed to place in the **Logging Level** field.

Table 8-1 Syslogd priority table

Priority	Value
LOG_EMERG	0
LOG_ALERT	1
LOG_CRIT	2
LOG_ERR	3
LOG_WARNING	4
LOG_NOTICE	5
LOG_INFO	6
LOG_DEBUG	7

Note: The STATISTICS reporting option uses priority code, 6, LOG_INFO.

This field is not the same as the LogLevel options coded in the PAGENT configuration file.

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 on ordering these publications, see “How to get IBM Redbooks” on page 244.

- *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383
- *Understanding LDAP,* SG24-4986
- *OS/390 Security Server, Ready for e-business,* SG24-5158
- *LDAP Implementation Cookbook*, SG24-5110
- *OS/390 Security Server 1999 Updates, Implementation Guide*, SG24-4629
- *Stay Cool on OS/390: Installing Firewall Technologies* SG24-2046
- *Implementing VPNs in a z/OS Environment* SG24-6530
- *Putting the Latest z/OS Security to Work*, SG24-6540

Other resources

These publications are also relevant as further information sources:

- *z/OS V1R3.0 Security Server RACF Security Administrator's Guide,* SA22-7683
- *z/OS Parallel Sysplex Overview: Introducing Data Sharing and Parallelism in a Sysplex,* SA22-7661
- *z/OS MVS System Commands,* SA22-7627
- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS Planning: Workload Management,* SA22-7602
- *MVS Planning: Global Resource Serialization,* SA22-7600
- *z/OS UNIX System Services Planning,* GA22-7800
- *HMC Operations Guide*, SC33-7988
- *PR/SM Planning Guide*, SB10-7033
- *IOCP User's Guide*, GC38-0097
- *TKE Workstation User’s Guide 2000*, GA22-7430
- *z/OS Communications Server IP Configuration Reference*, SC31-8776
- *z/OS Communications Server IP Configuration Guide*, SC31-8775
- *z/OS Security Server LDAP Server Administration and Usage Guide*, SC24-5923
- *z/OS LDAP Client Application Development Guide and Reference*, SC28-5924
- z/OS Communications Server: IP System Administrator’s Commands, SC31-8781
- *OS/390 V2R10.0 C/C++ Run-Time Library Reference,* SC28-1663

Referenced Web sites

These Web sites are also relevant as further information sources:

- A complete and up-to-date list of coupling facility details and enhancements introduced with each CFLEVEL

  http://www.ibm.com/servers/eserver/zseries/pso/cftable.html

- you can use the coupling facility structure sizer tool, which can be found at the Web site

  http://www.ibm.com/servers/eserver/zseries/cfsizer

- Java executable can be obtained at the following URL

  http://java.sun.com/

- The download and installation instructions are written for Windows 2000 and Linux. The following information and executables are located at:

  http://www-3.ibm.com/software/network/commserver/downloads/zidsmanager.htm

How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

S

T

U

V

W

X

Z

Security Configuration in a TCP/IP Sysplex Environment

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Security Configuration in a TCP/IP Sysplex Environment

Security perspective on Parallel Sysplex physical configuration

Topography of an Internet connection and security measures

Utilization of TCP/IP in a sysplex

This IBM Redbook will help those Parallel Sysplex installations that are considering serving users over non-secure TCP/IP networks, such as the Internet, to achieve a broad understanding of the threats to their security. It offers configuration design considerations to keep the related risks at a minimum. The following areas are discussed:

- Utilization of z/OS® UNIX System Services and the relevant security setups
- The Parallel Sysplex physical configurations as scrutinized from the security aspect
- Exploitation and protection of the sysplex coupling mechanisms
- Utilization of TCP/IP in a sysplex and the associated security exposures
- Topography of an Internet connection and security measures

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6527-00 ISBN 0738423939