Draft Document for Review March 29, 2010 3:35 pm

SG24-7803-00

Security on the Mainframe

IBM Mainframe Security Concepts

IBM Security Blueprint and Framework

Operating System and Application Security

Karan Singh
Thomas Castiglion
Lennie Dymoke-Bradshaw
Pekka Hanninen
Vincente Ranieri Junior
Patrick Kappeler

ibm.com/redbooks

Redbooks

International Technical Support Organization

Security on the Mainframe

December 2009

SG24-7803-00

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

First Edition (December 2009)

This edition applies to the IBM System z10 Enterprise Class server, the IBM System z10 Business Class server, and Version 1, Release 11, Modification 0 of z/OS (product number 5694-A01).

This document created or updated on March 29, 2010.

Contents

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX/ESA®
AIX®
CICS®
DB2®
DirMaint™
Domino®
DS8000®
ESCON®
FICON®
HiperSockets™
i5/OS®
IBM®
IMS™
Language Environment®
Lotus Notes®
Lotus®
Notes®
OS/390®
OS/400®
Parallel Sysplex®
POWER®
PR/SM™
Processor Resource/Systems Manager™
RACF®
Redbooks®
Redguide™
Redpaper™
Redbooks (logo) ®
System Storage™
System x®
System z10™
System z®
Tivoli®
VM/ESA®
VTAM®
WebSphere®
z/Architecture®
z/OS®
z/VM®
z/VSE™
z9®
zSeries®

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

Preface

This IBM® Redbooks® publication documents the strength and value of IBM's security strategy with System z hardware and software. In an age of increasing security consciousness, IBM's System z provides the capabilities to address the needs of today's business security challenges. This publication will explore how System z hardware is designed to provide integrity, process isolation and cryptographic capability to help address security requirements. We highlight the features of z/OS and other operating systems which offers a variety of customizable security elements within the framework of the Security Server and Communication Server components. We discuss z/OS and other operating system and additional software which leverages the building blocks of System z hardware to provide solutions to business security needs. This publication's intended audience are technical architects, planners and managers who are interested in exploring how the security design and features of System z, the z/OS operating system and associated software addresses current issues such as data encryption, authentication, authorization, network security, auditing, ease of security administration and monitoring.

The team who wrote this book

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

Karan Singh is a Project Leader at the ITSO.

Thomas Castiglion is a software engineer with the IBM Software Group.

Lennie Dymoke-Bradshawis a long-term MVS practitioner. He has worked with S/370 since 1975 when he started programming a 370/145 running DOS/VS in PL/1 and assembler. Over the years he has maintained his interest in programming while performing a variety of roles in applications programming, systems design, technical support, systems programming, software support, and whatever other technical roles customers have asked him to do. During this time he has worked with every level of RACF since RACF 1.3, and he maintains a keen interest in computer security especially as applied to System z. Lennie joined IBM in 1996 and nowadays he works more with System z cryptography as well. Lennie currently works in IBM System z software pre-sales in the UK, and is trying to publicise the enormous benefits of System z from a security standpoint.

Pekka Hanninen is working Crypto and Security specialist with C7D8 Consulting company, based in Helsinki, Finland. He was an IT Specialist over 12 years in IBM Finland until his recent retirement. He has over 35 years of experience in IBM large System software. His areas of expertise include cryptography, RACF and security administration. He holds certificates for CISSP, CISA and CISM.

Vincente Ranieri Junior is an Executive IT Specialist at the Advanced Technical Support (ATS) team reporting to the Washington Systems Center. He has 28 years of experience working with IBM customers, 23 of which providing technical support for the System z™ platform. He is the System z Security Regional Designated Specialist for Americas South Region, leading several security projects across the region. He writes extensively and has co-authored several Redbooks. He also teaches IBM classes worldwide on all areas of System z security.

Patrick Kappeler is a former IBM technical expert in the domain of IBM System z eBusiness Security. He practiced for the last decade both as an IBM Certified Consulting I/T Specialist in the Montpellier (France) European Products and Solutions Support Center (PSSC) and as an IBM Redbooks® co-author and project leader for the International Technical Support Organization (ITSO) in the IBM Poughkeepsie (USA) Laboratory. Patrick is now providing worldwide independent consultancy and education on IBM mainframes Security, covering areas such as System z hardware integrated cryptography, RACF, Public Key Infrastructure, secure protocols, IP Security and LDAP.

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

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Part 1

Introduction

In this part we introduce IBM Security Framework and IBM Security Blueprint. We also look at the history of security on the IBM Mainframe.

1

Introduction

This chapter introduces the IBM Security Blueprint and its relevance to System z®. It sets out a high-level, business-oriented view of the security landscape.

- 1.1, “IBM Security Framework”
- 1.2, “Framework and Blueprint”
- 1.3, “IBM Security Blueprint”

1.1 IBM Security Framework

At a strategic level, security for information technology is the protection of resources. These resources can be categorized in many different ways, such as by threat profile, by network layer or by product line. To make decisions about security, a consistent view of threats and solutions is needed. The IBM Security Framework is a perspective developed by IBM to describe security from a *business* viewpoint.

The business factors which drive security measure value, risk and cost. Value drivers determine the worth of assets, of the system to the business, and of the business itself. Risk drivers involve compliance, corporate structure, corporate image, and the risk tolerance of the company. Cost drivers determine productivity impact, competitive advantage, and economic feasibility.

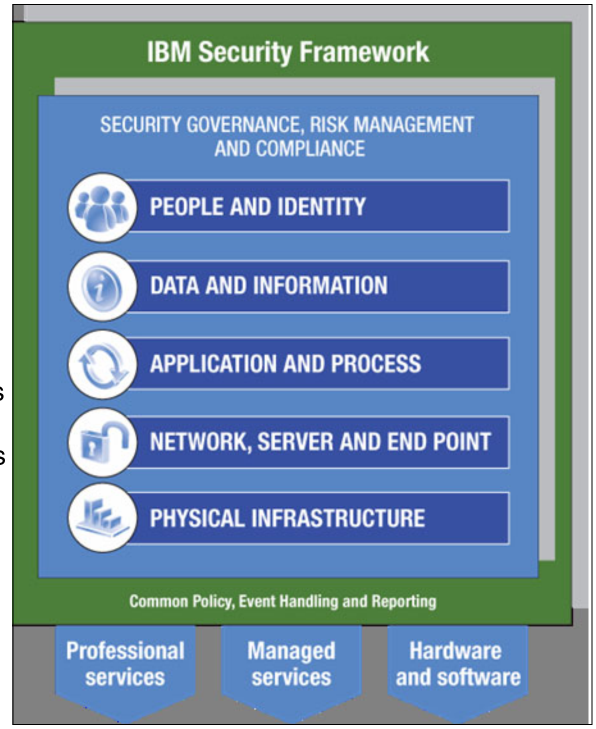


Figure 1-1 The IBM Security Framework

At an even higher level, the business reasons which influence security are correctness, reliability, and safety.

- *Correct* operation means that the information technology operations achieve the desired result.
- *Reliable* operation means that the same result occurs predictably, providing stakeholders with surety.
- *Safe* operation means that people and business resources are not harmed by the technology.

Under the IBM Security Framework, these business drivers are used to group business elements in need of securing are into *resource domains*, categories applicable to business policy decisions.

1.1.1 People and Identity

Organizations need to protect the assets and services that serve the business and support the business operation. One aspect of protection is provided by *access control*. The ability to provide effective access control services is based on the ability to manage People and Identity. Enterprises define access control models - relationships between people and identities expressed in terms of *role*, *rights*, *business policies*, and *rules*.

Operationally, people acting in authorized roles in an organization or as part of an extended relationship are granted access (or *rights*) to infrastructure, data, information, and services. At the same time, people acting in unauthorized roles are denied access if they are acting outside of the business policies and agreements.

Identity systems need to be integrated with appropriate sets of access controls. Unless multi-platform identity systems are available to manage user roles, rights, and privileges across the IT infrastructure, the presence of multiple technological architectures will require multiple identity and access control systems to ensure that users have access to the right assets and services.

1.1.2 Data and Information

Raw data needs to be available to the business, today and tomorrow, as does *contextualised information*. Data storage issues include removable media, uncontrolled or unstructured access, inconsistent policies, legal requirements, and the cost of tracking who touches what data when. Secure *Storage Management* solutions need to prevent data breaches, notify stakeholders of security events, and control access to data.

An effective plan for data and information protection includes maintaining a catalog or inventory of these assets, along with attributes, policies and enforcement mechanisms and services that govern the access, transformation, movement and disposition of data and information.

1.1.3 Application and Process

Organizations need to protect their *business-critical applications* from external and internal threats throughout their entire life cycle, from design to implementation and production. Control throughout the application life cycle implies effective control and compliance in the remaining security domains.

For example, whether an application is internally focused, such as a customer relationship management (CRM) system delivered through a service-oriented architecture (SOA), or an externally facing application, such as a new customer portal, clearly defined security policies and processes are critical to ensure the application is enabling the business rather than introducing additional risk.

Service Management for all business and business support processes, including Service Management for processes within the security domain, is a critical part of ensuring that the business is operating within the appropriate risk management and compliance guidelines. Service Management of Security would typically include a combination of capabilities, such as centralized authentication, access and audit policy management, and Web application vulnerability scanning and intrusion prevention.

1.1.4 Network, Server and Endpoint

Organizations need to *preemptively* and *proactively monitor* the operation of the business and the IT infrastructure for *threats* and *vulnerabilities* in order to avoid or reduce any breaches.

Security monitoring and management of an organization's network, server, and endpoints are critical to staying ahead of emerging threats that can adversely affect system components and the people and business processes they support. The need to identify and protect the infrastructure against emerging threats have dramatically increased with the rise in organized and financially motivated network infiltrations. While no technology is perfect, the focus and intensity of security, monitoring, and management can be affected by the type of network, server, and endpoints deployed in the IT infrastructure and how those components are built, integrated, tested, and maintained.

1.1.5 Physical Infrastructure

In order for an organization to effectively implement an enterprise security plan, the business and technical risks that are associated with the physical infrastructure must be understood and addressed. Security Governance, Risk, and Compliance provides guidance on the types of risks and the types of plans and responses for physical security.

Protecting an organization's infrastructure may mean taking precautions against a failure or loss of physical infrastructure that could impact business continuity. Protecting an organization's infrastructure may involve protection from indirect threats and vulnerabilities, such as the impact of loss of a utility service, a breach in physical access control, or loss of critical physical assets. Effective physical security requires a centralized management system that allows for correlation of inputs from various sources, including property, employees, customers, the general public, and local and regional weather.

For example, securing the perimeter of the data center with cameras and centralized monitoring devices is critical to ensure managed access to an organization's IT assets. Therefore, organizations concerned about theft and fraud, such as banks, retail stores, or public agencies, should define and implement an integrated physical security surveillance strategy that includes monitoring, analytics, and centralized control. This approach enables organizations to extract intelligent data from multiple sources and respond to threats sooner than manually monitored environments, resulting in reduced cost and risk of loss.

1.2 Framework and Blueprint

The Security Framework is technology-neutral, providing a guide to *what* solutions are needed rather than *how* they are to be delivered. Solutions for problems in a given domain tend to share characteristics, and can be evaluated using common criteria.

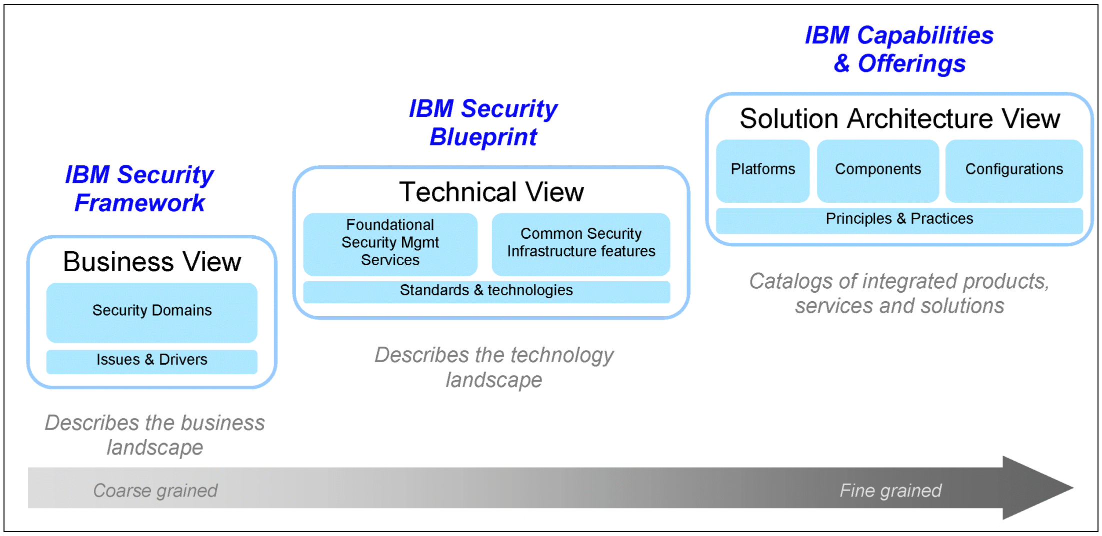


Figure 1-2

The next step after the *IBM Security Framework*'s business view is the *IBM Security Blueprint*, a map of security concerns and responses for technical architects. Designed primarily for use by technical stakeholders, the Security Blueprint goes beyond a strategy view to add a more detailed operational and technical view. While the Security Blueprint is still product- and platform-agnostic, it provides a guide to the architecture of security systems.

The Blueprint provides system views, component catalogs and solution patterns which can be used to build and analyze secure IT systems. For more information on understanding and applying the IBM Enterprise Security Architecture, see *Introducing the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*, REDP-4528

1.3 IBM Security Blueprint

The IBM Security Blueprint divides security architecture into four layers: business domains, security controls, security technologies and architectural principles. The IBM Security Framework defines the business domains, and the other three layers are defined within the Blueprint to provide an *architectural* view of IT security.

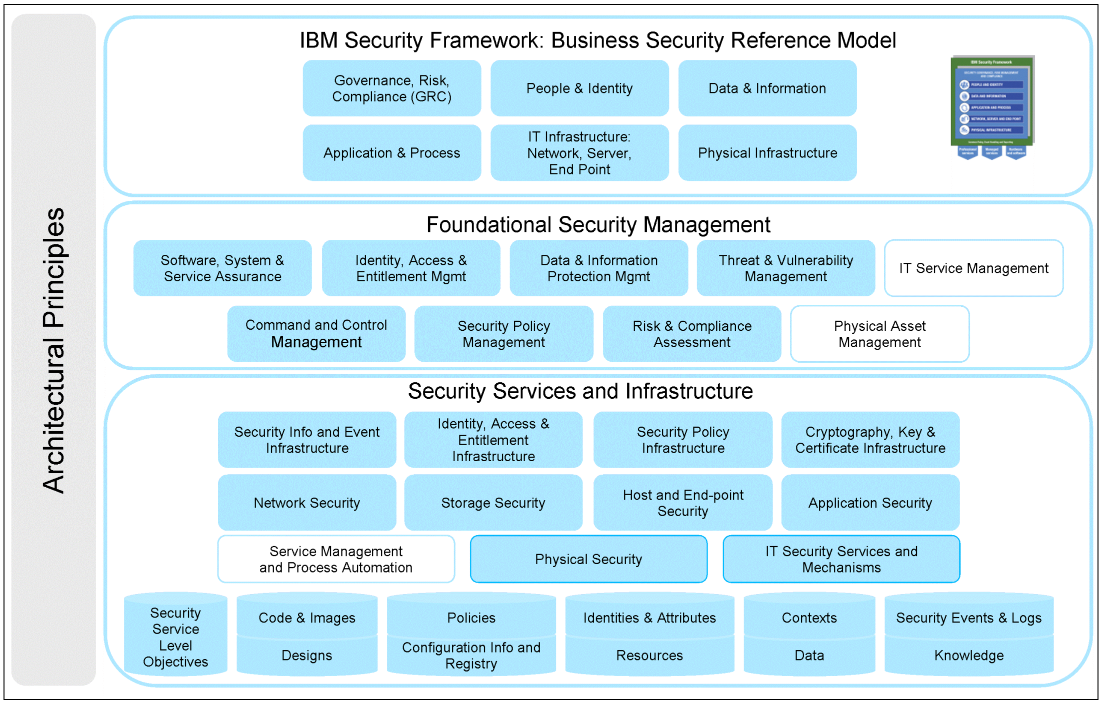


Figure 1-3

Foundational Security Management

The Foundational Security Management layer contains the top level components, which can be directly mapped to the IBM Security Framework. Each Foundational Security Management component represents business controls, rather than technology. The sublayers themselves consist of multiple individual and linked services:

- *Risk and Compliance Assessment* enables the IT organization to collect, analyze, and report security information and security events in order to identify, quantify, assess, and report on IT related risks that can contribute to the organization's operational risk. This component covers *risk aggregation and reporting*, *IT security risk processes*, *business controls management*, *resiliency and continuity management*, *compliance reporting*, and *legal discovery services*.
- *Command and Control Management* provides the command center for *security management* as well as the *operational security capabilities* for non-IT assets and services to ensure protection, response, continuity, and recovery. It covers topics such as ensuring that physical and operational security is maintained for locations, assets, humans, environment and utilities, providing surveillance and monitoring of locations, perimeters and areas, enforcing entry controls, providing for positioning, tracking, and identification of humans and assets, and providing a focal point for continuity and recovery operations.
- *Security Policy Management* provides all services and repositories to author, discover, analyze, transform, distribute, evaluate, and enforce security policies.

- *Identity, Access, and Entitlement Management* provides services related to roles and identities, access rights, and entitlements. The proper use of these services can ensure that access to resources has been given to the right identities, at the right time, and for the right purpose. These services can also address that access to resources is monitored and audited for unauthorized or unacceptable use.
- *Data and Information Protection Management* provides services that protect unstructured and structured data from unauthorized access and data loss, according to the nature and business value of information. It also provides usage and access monitoring and audit services.
- *Software, System, and Service Assurance* addresses how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software life cycle to create predictably secure software. This component covers structured design, threat modeling, software risk assessment, design reviews for security, source code reviews and analysis, dynamic application analysis, source code control and access monitoring, code/package signing and verification, quality assurance testing, and supplier and third-party code validation.
- *IT Service Management* provides the process automation and work flow foundation for security management. In particular, Change & Release Management processes play a significant role in security management.
- *Threat and Vulnerability Management* provides services that identify vulnerabilities in deployed systems and receive reports of vulnerabilities from outside sources, determine the appropriate response, and make proactive changes to deployed systems to maintain the security of the deployed system.
- *Physical Asset Management* provides awareness of the location and status of physical assets as well as awareness of physical security controls and coordinates the security information for physical systems with the IT security controls.

For more detail on the Foundational Security Management sublayers, see the following IBM Redbooks publications[1].

- *IBM Enterprise Security Architecture for People and Identity*, SG24-7751
- *IBM Enterprise Security Architecture for Governance, Risk and Compliance*, SG24-7750
- *IBM Enterprise Security Architecture for Data and Information*, SG24-7752

Security Services and Infrastructure

The Security Services and Infrastructure layer contains the top level components, which can be directly mapped to the IBM Security Framework. Each Foundational Security Management component represents business controls, rather than technology. The sublayers themselves consist of multiple individual and linked services:

- *Security Information and Event Management Infrastructure* provides the infrastructure to automate log aggregation, correlation, and analysis. It also enables an organization to recognize, investigate, and respond to incidents automatically, and streamline incident tracking and handling, with the goal of improving security operations and information risk management.
- *Identity, Access, and Entitlement Infrastructure* provides services to manage user provisioning, passwords, single sign-on, access control, and synchronization of user information across directories.
- *Security Policy Infrastructure* provides services to manage the development implementation of security policies in a consistent manner and automate the deployment of those policies to IT systems.

[1] These IBM Redbooks publications are currently in development and will be published at a later time in 2009.

- *Cryptography, Key, and Certificate Infrastructure* provides services to perform cryptographic operations efficiently and provides operational processes and capabilities to manage cryptographic keys.
- *Network Security* consists of multi-layered network security to provide defense in depth, deep inspection, and analysis of protocols, application level payloads, and user content to protect at all levels of the network stack. It extends to virtual networks for security in modern, heavily virtualized environments.
- *Storage Security* provides data centric security capabilities for protecting data in use, in transit, and at rest through isolation and encryption capabilities. It also provides services to catalog and classify storage assets and associate control policies with them.
- *Host and Endpoint Security* provides protection for servers and user devices, such as mobile phones, desktop computers, and mobile computers using both host and network based technologies. This protection integrates into the virtualization infrastructure to provide security for virtual environments. It includes hardware based attestation of host OS and system resources to protect against malicious attacks.
- *Application Security* provides the infrastructure for testing, monitoring, and auditing deployed applications.
- *Service Management and Process Automation* consists of the infrastructure services to handle service management processes, such as incident, problem, change, and configuration management. Process automation are generic framework-based services to automate IT actions, including security related activities.
- *Physical Security* are IT infrastructure services to create awareness of physical security and coordinate it with IT security. This can include employee badges, RFID readers, surveillance systems, and associated technology or assets. Physical security may include automation related to surveillance, motion detection, object and human identification and tracking, entry control, environmental system monitoring, perimeter control, and power and utility system monitoring.
- *IT Security Services and Mechanisms* provide instrumentation to IT systems for collecting security information and configuration information from IT systems.

Architectural Principles

IBM security architects have defined the following *Architectural Principles* that accompany the conceptual layers. These can be applied to all levels of the blueprint, framework or solution designs, and are also guidelines for IBM products and solutions.

- Openness.

  Openness is of primary importance in an enterprise environment. This includes support for all major platforms, runtimes, languages, support for major industry standards, published interfaces and algorithms, no security by obscurity, documented trust and threat models and support for Common Criteria, and similar formal security validation programs.

- Security by default.

  Security should not be an afterthought in IT solutions, but security policies should be secure out-of-the box. This is helped by a consistent definition and management of configurations, a consistent set of security roles and persona across products, and a consistent security management user interface.

- Design for accountability.

  In today’s environments, with many requirements in the compliance area, it is important that all security-relevant actions can be logged and audited, the audit infrastructure should be scalable to handle these events, and audit information must be immutable and non-repudiable.

- Design for regulations.

  Regulations drive many requirements in IT security projects, and regulations change over time. To handle this, it requires flexible support for the constraints set by government regulations and industry standards and traceability between regulations, standards, and business policies and the security policies used to implement them.

- Design for privacy.

  In the current age of data sharing, privacy becomes increasingly more important. Solutions should highlight the use of personally identifiable information and corresponding data protection mechanisms, and enable the principles of notice, choice, and access.

- Design for extensibility.

  Good solutions are component based and separate the management of mechanisms from the mechanisms themselves, to support a variety of mechanisms under the same framework. Already deployed systems must allow for the addition and extension of new mechanisms within the existing management framework.

- Design for sharing.

  Multiple solutions can share a single IT environment, such as in a shared service center. To achieve this goal, security services and management must be able to span multiple domains, each domain potentially providing its own and independently set security policy, identity, models, and so on. Architectures must explicitly document the assumptions and limitations made in terms of span of control.

- Design for consumability.

  All security services must be easily used by a variety of audiences. This includes programmers who develop and integrate applications with the security services, management systems that create, update and manage security policies and other security artifacts, and people who manage security activities, audit security activities, and request access to protected resources.

- Multiple levels of protection.

  Defense in depth is a general principle, which can be achieved by multiple levels of enforcement and detection. Resources must be designed to protect themselves as a first layer of defense. Intrusions can be contained through *isolation* and *zoning*. Multiple levels also minimize the attack surface to the outer-most accessible layer. *Least privilege* is a similar fundamental principle. Finally, the system should incorporate fail-safe principles.

- Separation of security management, enforcement, and accountability.

  Security management services (identity, authorization, audit, and so on) shall be provided through a dedicated and shared security infrastructure, enabling consistent monitoring and enforcement. The enforcement itself (through cryptography, through policy enforcement, or through physical isolation) is typically distributed and kept close to the resources.

- Security-critical resources must be aware of their security context.

  Resources and actors are kept aware of their environment (including physical location and logical co-location), and their security status and context.

- Security is model-driven.

  Models are reflective of the operating environment, common models, and consistent formats for identity and trust, data, policy, applications, security information and events, and cryptographic keys. Models are consistently interpreted across the stack (for example, network identities are linked to application-level identities) and across units (for example, policies and trust are negotiated and understood within a federation). Models are consistently validated against reality (feedback from policy and model discovery).

- Consistency in approaches, mechanisms, and software components

  Two independent layers of protection for one resource may improve security. But using two different mechanisms for the same purpose for two resources increases the chances that at least one of them gets broken (plus, they increase management impact).

  The IBM Security Blueprint lists the preferred standards and mechanisms.

2

Security of the IBM Mainframe: Yesterday and Today

In this chapter we introduce the IBM mainframe in a historical context. We show the operating systems that it can run, and give some details of each of their heritage. We then move on to give a historical perspective of the IBM mainframe from the late 1960s to the present day with an emphasis on the security issues that were faced at each time. It should be plain that security issues faced in today's connected world and very different from those faced by the early S/360 systems.

We then attempt to show the nature of present day IBM mainframe installations, and give some perspective on how they are used and run.

We give details of the qualities of service of the mainframe, and then move on to demonstrate how the confidence that many have in the IBM mainframe is borne out by,

- IBM's own confidence over nearly 40 years
- The longevity of the platform
- The concepts of certification achieved as achieved by various mainframe components.

But we start with some description of IBM mainframe operating systems.

2.1 Operating Systems

While some hardware platforms are designed to run a single operating systems this is not true for all. Intel® platforms can run several different operating systems, from the early version of PC-DOS to OS/2 to multiple versions of Windows® and lately, Linux® and Apple Macintosh OS.

For System z there have been many versions of various operating systems from the early days of S/360 to the recent time of the System z hardware.

Some operating systems have been used, but are no longer in use. Many of these early operating systems were part of the intense development which was undertaken in the early years of mainframe development. Some of these were early predecessors of systems we have now. Others were development paths that were investigated but which proved less suitable for the restrictions which business systems needed. So names as,

- TPS
- DPS
- OS/PCP
- BOS/360
- TOS/360
- TSS/360
- CP-40
- CP-67
- AIX/ESA®

represent operating systems which once ran on S/360 or S/370 or the successor hardware but which have not been used for many years.

However there are some very important families of operating system some of which have been around for many years.

2.1.1 z/OS Operating System family

When we get to describing the security characteristics of the z/OS® operating system it will be distinguished from z/OS, which is really a bundle of the z/OS Operating System and many other components. However let’s just call it z/OS here.

In the 1960s IBM developed the S/360 hardware and had several operating systems which could run on it. Two of them were very close in nature, and applications could be simply ported between the two. These were OS/MFT and OS/MVT.

These later became OS/VS1 and OS/VS2. OS/VS1 was allowed to die in the 1970s. OS/VS2 became MVS. MVS went through many incarnations, such as MVS/SE, MVS/SP, MVS/XA, MVS/ESA, OS/390® and eventually z/OS.

z/OS is the most sophisticated of IBM’s mainframe operating systems and when used in a clustered manner (i.e. in a sysplex) it can supply the highest levels of availability, and scalability.

However, z/OS does not have some of the characteristics and capabilities of other operating systems on System z. For example it cannot run other operating systems as guests.

2.1.2 z/VM Hypervisor family

The origins of z/VM® lie in a system known as CP-40 then CP-67 which was developed at IBM's Cambridge Scientific Center. This was developed as an operating system hypervisor and released in 1972 as VM/370 and later VM/CMS.

The VM family of systems are characterized by being able to run other operating systems as guests. VM was capable of running multiple copies of MVS as a guest.

VM progressed as VM/SP, VM/SP HPO, VM/SF and VM/XA.

VM was enhanced greatly when PR/SM™ became available due to the presence of the SIE instruction.

VM/ESA® was introduced in the 1990s and z/VM in October 2000.

z/VM is capable of hosting z/OS guests as well as the other operating systems described in this section. z/VM can also host itself if required. Nowadays it is frequently used to host multiple Linux systems which can take over the roles of decentralized servers in an enterprise.

IBM is so serious about this capability that it has produced a special type of PU (a specialty engine, called an IFL) which will run z/VM and Linux only (i.e. it will not run z/OS or z/VSE™). This can provide some pricing advantages.

A short note about CMS

CMS or the Conversational Monitor System is an operating environment which runs only on the VM family of operating systems. It runs as a guest and is designed for single user use. It has only the security features needed for a single-user system and for resource access control relies on the control of the VM operating system. CMS is distributed with each VM operating system and is an important part of the means to maintain the operating system, as well as being a potential single user production environment in its own right.

2.1.3 z/VSE family

DOS/360 was one of the early operating systems available in the 1960s. DOS/360 became DOS/VS then DOS/VSE, then VSE/SP and finally z/VSE.

z/VSE has a Job Control Language but this is different to that of the z/OS family of operating systems.

z/VSE is frequently run under z/VM.

2.1.4 z/TPF family

In the 1960s IBM developed a very restricted function high performance operating system for the Airline Industry, called ACP (Airlines Control Program). This was available as a free product.

In the late 1970s this was transformed into TPF (Transaction Processing Facility) and was marketed by IBM.

Latterly this has morphed again into z/TPF and has a full 64-bit architecture.

2.1.5 Linux

Linux was developed originally by Linus Torvalds. It is an open source operating system, but is partly supported by many commercial organizations including IBM.

Linux on System z was developed in the late 1990s, and in recent times has made great advances as a free operating system, frequently run under z/VM, and able to provide a wealth of function at low cost.

The Linux distributions for System z include one from RedHat, one from Suse. The distributions include specialized drivers for System z hardware, all distributed in open-source form.

2.2 History of the Mainframe

Many histories of the Mainframe have been prepared over its long history. Each one has given a slightly different definition of what is meant by the term “mainframe” and each has presented a different view.

This history is here to serve a specific purpose. This purpose is to show that IBM mainframe hardware *and* software has grown in terms of its security capability from a relatively simple batch processing machine to one capable of handling vast numbers of data items and transactions, each owned and run by different identities, all run within an environment which can guarantee separation, can produce predictable results, reliably, and most importantly, that exhibits an unprecedented level of operational integrity that became the foundation for building very strong security features.

However, much of the security capability was built into the hardware in the early systems. The concepts of storage key separation and privileged operational states existing from the very first S/360 machines. The advances in programming capabilities over this period have made extensive use of those hardware capabilities to provide the highly configurable security controls which we use today. This demonstrates a remarkable level of foresight by the early hardware designers.

For our purposes we will be addressing ourselves only to the IBM mainframes, and only to the series which commenced with the S/360 series which was first available in 1964.

We shall be looking at the mainframe in five yearly chunks, and describing what might be a typical system in each era. We show the hardware and software uses and the way in which the systems were being used. In each case we attempt to highlight the security landscape and threats of the era.

So lets start our journey with the S/360 mainframe of the late 1960s.

2.2.1 Late 1960s

S/360 mainframes were becoming popular with many clients. These computers operating systems such as TOS, DOS and OS (which came in several flavours). They used punched card for input and controls and printers for output. Each system had a console, with an operator. The only type of processing was batch, and these systems used the language known as JCL.

There would have been no networking capability, other than locally attached screens.

However, in the IBM laboratories intense development work was taking place which was to produce the next generation of mainframes. Time sharing systems were being developed which would free the mainframe from its batch only status. Virtual storage systems were under development which would free the machine from the limitations of small memory sizes and enable the extensive resource sharing which became the mark of the mainframe. Even in those early days extended addressing capabilities of the hardware and software were under investigation. Some of this early work (such as addressing capabilities beyond the original 24-bit limitations) would only come to fruition nearly two decades later.[1]

[1] Interested readers may like to investigate the 32-bit addressing capabilities of the S/360 Model 67 which was made available in 1966.

Security landscape and threats

What threats? This system has a single user (the operator). All access controls would be related to physical access.

2.2.2 Early 1970s

S/370 models have largely replaced S/360. These now have virtual storage capabilities although true virtual storage memory separation capabilities were still in their infancy if compared to today's commercial systems.

Operating systems were DOS/VS, OS/VS1 and OS/VS2. OS/VS2 later became MVS, which introduced address space memory separation.

CICS® was available on DOS/VS, OS/VS1 and OS/VS2. DOS/VS used POWER® as a spooling system, while OS/VS1 used JES. OS/VS2 used HASP and ASP.

As OS/VS2 Release 3 became available it was named MVS and introduced address spaces. IBM issues "Statement of Integrity" for MVS.

VM/370 was available for those customers wanting a virtual machine environment for running multiple levels of operating systems.

TCAM and BTAM networking systems were used in some installations. BTAM was used for IMS™ which was available on OS/VS2. TCAM was available to support TSO.

Security landscape and threats

Little software is available to provide any concept of user identification.

TSO is available on a few machines. This provided an early registry of users called UADS, but applied only to TSO and had rather simple password controls. The concepts of access control to resources were still under development.

Some data sets are password protected using the PASSWORD data set, but this is rare. As processing is still predominantly batch processing this is not yet a major issue.

2.2.3 Late 1970s

S/370/168 is available and has about 2.7 mips of computing power. Attached processor versions of 158 and 168 are available as well. 303x series of processors available later.

DOS/VS becomes DOS/VSE.

MVS ran JES2 and JES3 spooling systems. MVS 3.7 and 3.8 are available and prove stable versions of MVS, and are widely adopted. MVS then changes to MVS/SE and thence to MVS/SP.

Growth in use of CICS and IMS for transaction based systems. TSO (and later ISPF) used greatly for systems programming, and application development.

SNA architecture growing in popularity along with 3705 communications processors. VTAM® becoming a very important MVS component.

Security landscape and threats

RACF® is introduced in 1976, along with some competitors products. This early RACF provides optional user authentication via a changeable 8-byte password together with controls on access to individual selected data sets. Password can be supplied at TSO logon and on a PASSWORD statement on a batch job. TSO submit process can be made to store the password in read-protected storage, for insertion into submitted jobs using an exit. If this was not done, JCL had passwords hard-coded.

Threats were related to access within companies, and the need to protect computer resources from unauthorized access in each enterprise over the networks.

2.2.4 Early 1980s

MVS/SP installed on 308x processors in growing number of installations. These were single, dyadic or quadratic processors having a possible real storage allocation of up to 64 megabytes.

MVS morphs into MVS/XA with vastly extended memory addressability and new I/O instructions and capabilities.

DOS/VSE becomes VSE/SP but still with only 24-bit addressability.

DB2® is introduced as IBM's relational database.

CICS gets Multi Region Option (MRO).

3725 processors replace 3705 processors, larger SNA networks created. Many enterprises give each developer their own screen, rather than having to share.

Security landscape and threats

RACF rapidly changes from a means to protect a selected set of critical data sets to a means to protect all data sets and multiple types of other resources. List of groups processing makes resource access control easier. Generic profiles make problem management of discrete profiles a thing of the past, and provides performance advantages too.

Threat profile starts to grow, with governments starting to face security and privacy issues.

2.2.5 Late 1980s

309x processors available. PR/SM adds multi-system capability on single processors. MVS/ESA introduced providing access to data spaces and larger amounts of memory. Advanced Address Space Facility (AASF) provides simplified cross memory services capability via access registers.

VSE still uses 24-bit memory.

CICS gains new architecture with multiple TCBs.

SNA extends to SNA interconnection capability providing links between and among multiple enterprises using linked 3745 processors.

Security landscape and threats

Security identification (userid and password) used for all mainframe access. Access to other enterprises applications secured via RACF.

RACF extends its control over multiple classes of resource. Most installations have protect-all for data sets, so RACF logon becomes mandatory. Rudimentary capability for security propagation available.

2.2.6 Early 1990s

Typical system might include MVS/ESA running on ES9000 processors with 128MB storage.

MVS/ESA OpenEdition becomes available, providing UNIX® capabilities. Little take up of this at first.

VSE/SP becomes VSE/ESA with both 31-bit real and virtual addressing.

CICS starts using RACF for transaction security.

MVS/ESA 3.1.3 provides massive leap in MVS security capabilities, with consoles secured, MVS commands secured, both JES's making use of RACF, security propagation from one address space to another and much else besides.

Most large enterprises have SNA network interconnections. Growing use of smaller systems and networks using TCP/IP and other networking protocols, such as Novell Netware and Netbeui.

Growth of smaller decentralized computing systems on the rise, with multiple logons required, confuses the computing identity landscape.

Security landscape and threats

RACF 1.9.2 provides large leap in security capabilities when operating together with MVS/ESA 3.1.3. Labelled security becomes possible with Security Labels and Multi-Level Security support. Whole system is built to achieve the B1 accreditation as measured against the US Department of Defense Orange book. This is subsequently achieved.

Large corporations starting to need identity programs for staff, which is proving more difficult in the face of a growing number of computing systems and platforms in many enterprises.

2.2.7 Late 1990s

IBM drop bipolar processing technology in favour of cheaper CMOS based processors. These processors rapidly progress from G3 to G6, with large performance gains. Sysplex capabilities available providing clustering for MVS systems.

Hardware encryption capabilities introduced into processors, with ICSF to manage them and provide CCA based API's for programming[2].

MVS morphs into OS/390 which solves some of the software licensing problems many organizations face.

Sysplexes grow in use providing better RAS characteristics for enterprises. CICS and DB2 make extensive use of sysplex functions.

Security landscape and threats

USS file systems become more standard as product exploit OS/390's UNIX capabilities. Websphere products make use of UNIX environment. IBM introduces “security services” as an offering.

Distributed systems and desktop systems start to merge into the computing landscape, as companies standardize platform choice.

[2] Some cryptographic capabilities had been introduced as early as 1990. However, they were atypical.

Connections to the internet provide a confusing array of capabilities for companies who are caught between wanting to make use of the available methods of marketing and sales, and fear over growing number of viruses and hacking attempts.

2.2.8 Early 2000s

OS/390 morphs once more, this time into z/OS. z/OS has 64-bit capability and runs on zSeries®. Potential virtual memory sizes are immense (16 exa bytes).

zSeries processors take large leaps in processing capability, with faster engines and more of them. PR/SM capabilities continue to grow.

Hardware encryption capabilities becomes more pervasive.

Explosion in TCP/IP growth as it overwhelms other networking protocols. Enterprises withdraw from using SNA except on internal networks.

Firewalls used extensively to protect companies from the hosts of malware in the internet.

Security landscape and threats

RACF grows into major identity management for some corporations, fits in and works with other identity management at others. RACF now works with multiple sysplexes and can propagate updates from sysplex to sysplex. It can also work with digital certificates which are becoming prevalent in the internet as a mechanism for identity management, and also a mechanism for client server authentication via encrypting protocols such as SSL.

2.2.9 Late 2000s

zSeries become System z. Mainframe hardware changes from z900 to z990 to z9® to z10 growing in power and capabilities at each step. Hardware cryptographic capabilities increase and are used by financial organizations for PIN management, ATM management and other financial transactions.

Cryptography is built into tape and disk systems with key management supplied TKLM.

TCP/IP is everywhere, with a need for enterprises to move to IPv6. SNA is shrinking in use.

Security landscape and threats

Public views of security become a major issue in corporations choice of policy, as multiple data loss incidents occur. “Encryption” becomes a buzzword representing a capability that all systems must make available.

Identity management is a reality in most large corporations. Methods and processes governing those systems still in need of work.

Companies have security policies mapping all their IT systems and other data as well.

2.3 The Mainframe today

Well the journey above may be something you recognize. Lets move on to describe the characteristics of IBM mainframe use in many of today’s corporations. For it is the qualities of service that the mainframe can provide which makes it such an attractive proposition for many enterprises. First, however, let us have a look at the personnel and roles that a typical installation might have.

2.3.1 Personnel and Roles

Unlike some computer platforms, the administration of the IBM mainframe is frequently distributed in nature. This has advantages, as more rigor is needed when planning cooperative events. The typical roles that might be used are as follows.

Table 2-1

Role	Responsibilities
Hardware planning	Responsible for new hardware, maintenance of exiting hardware, positioning of processors and so on.
Systems programming	Operating systems software installation, maintenance, debugging of software problems, configuration of LPARs and operating systems.
Middleware management	Middleware software installation, maintenance, debugging of software problems and configuration.
Security Administration	Maintenance of identity management, security policy and implementation, access controls.
Storage administration	Provision of storage and data availability on both DASD and tape based storage.
Capacity planning	Performance monitoring, capacity monitoring, prediction of capacity growth.
System operator	Day to day, hour to hour operation of system, with problem management and recording.
Change management	Management of problems and all potential and real changes to the entire System z environment. Planning for large scale changes and micro-management of small changes, to ensure minimized impact.

Note: The above list is neither expected to be treated as exhaustive, nor to define what is necessary in every installation. Each installation will have its own set of standards and responsibilities. This is merely an example.

Perhaps the most important of these is the last. System z environments are frequently heavily change-managed. It is in the nature of the disciplines which have grown up around the mainframe that change management is treated so seriously. The mainframe is frequently the security hub and the central data server in many large organizations, and the impact of “down time” can simply become unacceptable.

2.3.2 Role of Mainframe

In today’s environment the mainframe is likely to be at the centre of the operation of large organizations. It can fulfill a multitude of roles,

– As a central data server hosting DB2 tables of enormous size, and providing back-end data processing for data requests. DB2 is one of the world’s most successful database systems and when running on z/OS is capable of providing high availability, high throughput and high performance. These capabilities are enhanced when running in a z/OS sysplex. DB2 is a world leader in scalable database solutions on z/OS.

DB2 also runs on Linux on System z.

- As a central management point for identity, hosting RACF identity management, but also providing distributed access via LDAP to other servers and platforms for authentication.

  RACF can be used as the central management point for identity management within an enterprise. It is capable of having information stored which relates to multiple other systems by using the custom segments facility. LDAP capabilities will allow RACF to be used for authentication over secured network links.

  RACF can also be used for central security access control by making use of a Pluggable Authentication Module (PAM) for Linux. A single user can have multiple identities on multiple Linux servers.

- As a Central network Policy Server for the enterprise's network management. The z/OS Commserver policy agent provides this capability. The centralized or distributed policy services are provided over connections between server and clients that are secured with AT-TLS. The distributed policy services provide a security mechanism based upon client login passwords or pass tickets.

- As a host for WEB services the mainframe can provide access to heritage services.

  SOA services within Websphere can be used to provide access to services developed over previous years which encapsulate business rules in various forms, such as Cobol routines, CICS transactions and so forth.

- The mainframe is still a very large host for volume transactions frequently run through CICS or IMS or one of the third party products which run on the mainframe. Typically an enterprise's online day may commence in the early hours of the morning, and finish in the early evening when the available processing capacity is turned over to batch processing.

- The mainframe still holds its place as the main processor of large scale batch processing. The capability to process large volumes of data at high speed is one of the mainframe strengths that has never changed. Many enterprises using IBM mainframes have significant volumes of batch processing every evening, when online transactions are at a low ebb. It is this ability to be fully utilized over the entire 24 hour day, which make the mainframe computing model so efficient and successful.

Of course in most situations the mainframe will fulfill multiple of the above roles. With z/OS sysplex clustering capabilities this can provide very high levels of availability and scalability to each enterprise.

2.3.3 Maintenance and history

z/OS systems have become complex in terms of the relationships between software elements and components. In consequence the z/OS has a sophisticated mechanism for managing the updating of software elements, known as SMP/E (System Modification Program / Extended). This program can be used during software installation, for maintenance and for software backout. It will handle the complex dependencies of software so that incompatible software is not concurrently installed.

Software fixes or changes are built by IBM and can be installed by the customer using SMP/E, so that the particular configuration of the customers system is managed. Thus changes made by the customer are always taken in to account when the fix or change is installed.

While providing fixes to problem and making them easily available and capable of being reliably installed is important, it is also very important that problems that have been discovered are tackled early and the knowledge gained in their analysis is not lost.

Since the 1970s IBM have had a large database in operation called Retain. Retain holds information on mainframe systems going back to the 1970s. This enables service personnel to track problems and understand usage and development issues over all the incarnations of the various operating systems. It is an invaluable source of information for both problem solvers and developers.

Early foresight

The casual observer of Retain, might see this as just a bunch of problem reports and tracking information.

However Retain shows how the early developers of the S/360, S370 hardware and the IBM Operating Systems had great insight and forethought about the problems that would be faced by future computing systems. Concepts related to multi-processing were tackled early; serialization issues like those tackled by the GRS component of z/OS are available in very early releases of those operating systems.

Similarly the concepts of security and integrity were recognized as distinct and highly important in the early 1970s. Many of the security issues that have plagued us in recent times on other computing platforms were addressed and resolved for System z by those early developers. The clear thinking of those IBM designers has stood the test of time, as IBM is still able to display its statements of integrity (see 2.4, “Statements of Integrity”) with a great deal of pride.

2.3.4 Change Control and continuous availability

Due to the sheer size of operations of many mainframe installations, the role of change control has become a major factor in its running. However, this introduces disciplines which provide their own benefit.

z/OS and z/VM in particular lend themselves well to the type of controlled operation which can be change managed. Backout capabilities for changes are built into the structure of the configuration files.

However, also built into the System z software and hardware are a multitude of capabilities of making changes in a non-disruptive manner. These capabilities are part of a thorough design aim which enables changes to be made in such a way as to be non-disruptive to applications. Frequently these capabilities require planning in the configuration of processors, LPARs, and applications, but the aim is to enable applications to continue running while changes are made.

As applications can be configured to run on z/OS sysplexes which straddle LPARs and processors, and can be configured to allow work to flow freely among those LPARs, it becomes possible to remove LPARs and processors from sysplex configurations and still allow applications to continue processing.

It is this property of continuous availability while change continues which characterizes System z and distinguishes it from other computing platforms.

Some examples of the continuous availability concurrent change features are,

- Ability to add or remove hardware in flight. System z makes advances in this area at each new hardware announcement. Even processor books, containing processors and memory can be added and removed while other work continues on the same System z mainframe.

- LPAR and PR/SM reconfigurations can be made by dynamic reconfigurations, while some LPARs remain active. Reconfigurations can include changing channel path definitions and device definitions.
- z/OS Sysplex reconfiguration allows moving of coupling facility structures while applications continue. Allows moving of coupling facilities to new processors.
- z/OS Sysplex reconfigurations will allow the seamless moving of work from one LPAR to the remaining members of the sysplex.
- z/OS allows the dynamic addition and removal of system libraries, and system exits and control points. This allow new releases of applications to be installed while the operating system remains active, even if new APF and other system libraries are required.
- z/VM recognizes new hardware in-flight and can make use of it for new and existing guests.

With this type of capability it is not practical to simply assume these thing are always possible. Careful planning is always required, and planning for future continuous operations and availability in disaster situations is not optional.

Nevertheless it is quite feasible for applications to stay active for years without outage and during that time,

- introduce new storage and move databases to it,
- decommission old storage,
- introduce new processors and move processing from LPARs on existing processors to it,
- Replace operating systems versions in a staggered fashion,
- Replace applications and middleware in a staggered fashion.

It is perfectly feasible to have an end-user application running on ***a z/OS sysplex on a particular set*** of System z hardware, and then to implement a series of changes, to new processing hardware, new DASD storage, new operating system level, new middleware version and even new end-user application version, and achieve all of the above without any outage. At the end of the exercise the entire hardware and software has been replaced, but the application has suffered no outage.

2.4 Statements of Integrity

The z/OS operating system, from its beginnings as OS/MVT and later as MVS in the 1970s, was built on S/360, then S/370 and now System z, a robust hardware architecture that provides for the safe execution of multiple applications for multiple users on one single platform.

A foundational design point of MVS was its very tight interaction with the S/370 hardware so that, for instance, the working storage for each application is separated from the storage of other applications, as well as from the storage used by the supervisor functions of the operating system. In addition, applications run in a different hardware “state” than supervisor functions and therefore do not have direct access to privileged hardware instructions.

Because of the experience that IBM gained from designing operating systems in the 1960s, it was able to understand very early on, the distinct concepts of *integrity* and *security*, and how important they would be to the future of computing. Consequently operating systems (in particular MVS and VM) were designed so that formal integrity statements could be made about them.

These integrity statement drew boundaries around the capabilities of the operating system and around the capabilities of application programs running on those operating systems. It was recognized that there had to be a class of programs which were able to function and interact at the level of the operating system, but also there had be fixed restraints on the abilities of “normal” application programmers, so that they did not disrupt the ability of the operating system to control the processor, and respond and manage the hardware; and also of course so that they could not interfere with the ability of the operating system to maintain secure definitions of identity.

The ability to strictly classify these two sets of programs was defined and implemented in the first releases of MVS and subsequently a similar concept was applied to VM systems. This was formalized as the “Statement of Integrity” which was issued publicly. It included a commitment from IBM to fix any issue which allowed those strict controls to be circumvented.

So IBM in the 1970s had already initiated its long-term commitment to System Integrity within its operating system designs and development practices.

The 1973 MVS “Statement of Integrity” has formed the basis for more than three decades of the MVS successors’ industry leadership in system security. The fact that IBM was able to make such emphatic claims so early in the life of the MVS family of operating systems, and has been able to maintain that claim for all the years since then, speaks volumes for the stability and reliability of the hardware and software.

The current z/OS statement of Integrity can be found in Chapter 7.2.5, “Statement of Integrity” on page 154.

Important: It is valid to assume that the programs IBM provides that need to operate in an authorized state on z/OS have been scrutinized during their development phase to ensure they do not expose the integrity of system operations. However, it might not be the same for programs from other vendors. Consequently it is highly recommended that software vendors should provide certification or assurance that their authorized programs will behave as specified when executing in z/OS.

At worst, these programs should be examined by the installation for proper design and coding. This will require access to the source code.

RACF

The z/OS “Statement of Integrity” also addresses the RACF External Security Manager.

The Statement of Integrity also asserts that only users with RACF administration privileges can provide these authorizations, and any other detected way of improperly getting these authorizations would be fixed by IBM.

As is highlighted above, if rogue authorized programs gain RACF privileges, this has to be investigated. However, if these programs misuse the privileges that the RACF administrators have granted them, then it is the installation’s responsibility and is not covered by the z/OS Integrity Statement.

z/VM

Following on from the MVS statement of integrity, IBM wished to ensure that the same levels of assurance could be given to applications running on VM, and also to operating system guests running on VM, such as MVS.

Hence it developed a statement of integrity for VM/SP which was specific to VM. This statement was significantly different insofar as it relates to the ability of one guest operating system to be prevented from interfering with the operation of another guest or of the VM hypervisor.

However, the promise made by the VM "Statement of Integrity" is as important as the one made for MVS.

The current z/VM statement can be found in Chapter 5.3.2, "z/VM system integrity definition" on page 112.

2.5 Certification

We have discussed the level of trust that IBM places in its own products and the extent to which it stands behind the reliability of the integrity statements with promises to fix problems if they occur.

Another justification for the trust placed in System z by so many enterprises, is the levels of certification that it has achieved over its lifetime. Lets spend some time now, looking at the concepts of certification.

2.5.1 Some history

The need for certifying security products or security functions in products arises from the fact that many consumers of IT lack the knowledge, expertise, or resources necessary to judge whether their confidence in the security of their IT products or systems is appropriate. After all, they might not wish to rely solely on the assertions of the developers of the product.

Over the years this has become quite an important issue as information held within computer systems is a critical resource that enables organizations to succeed in their mission. Enterprises therefore have a reasonable requirement that personal information contained in computer systems remains private, while still being available as needed, and while ensuring that it is not subject to unauthorized modification.

IT products or systems should perform their functions while exercising proper control of the information to ensure it is protected against hazards such as unwanted or unwarranted dissemination, alteration, or loss. The term IT security is used to cover prevention and mitigation of these and similar hazards. Consumers might therefore choose to increase their confidence in the security measures of an IT product or system by ordering an analysis of its security (in other words, a security evaluation)

Evaluation standards and criteria were put in place in the early 80's in different countries, such as the TCSEC (Trusted Computer System Evaluation Criteria) standard issued by the United States Government Department of Defense (DoD). Europe implemented its own evaluation standard, ITSEC (Information Technology Security Evaluation Criteria) in 1990, and so did Canada with the CTCPEC (Canadian Trusted Computer Product Evaluation Criteria).

Although aimed at providing users of IT products with a fair and independent expert evaluation of the products' security related functions, these standards diverged in the evaluation criteria that were applied and their rating systems. Furthermore, the resulting evaluations were not internationally recognized.

In October 1998, after two years of intense negotiations, government organizations from the United States, Canada, France, Germany, and the United Kingdom signed a historic mutual recognition arrangement for Common Criteria-based evaluations.

This arrangement, officially known as the Arrangement of the Recognition of Common Criteria Certificates in the field of IT Security, was a significant step forward for government and industry in the area of IT product and protection profile security evaluations. The partners in the arrangement share the following objectives in the area of Common Criteria-based evaluation of IT products and protection profiles:

- To help ensure that evaluations of IT products and protection profiles are performed to high and consistent standards and are seen to contribute significantly to confidence in the security of those products and profiles
- To increase the availability of evaluated, security-enhanced IT products and protection profiles for national use
- To eliminate duplicate evaluations of IT products and protection profiles, and
- To continuously improve the efficiency and cost-effectiveness of security evaluations and the certification/validation process for IT products and protection profiles.

The purpose of this arrangement is to advance those objectives by bringing about a situation in which IT products and protection profiles which earn a Common Criteria certificate can be procured or used without the need for them to be evaluated and certified/validated again.

It seeks to provide grounds for confidence in the reliability of the judgement on which the original certificate was based by declaring that the Certification/Validation Body associated with a Participant to the Arrangement shall meet high and consistent standards. The Arrangement specifies the conditions by which each Participant will accept or recognize the results of IT security evaluations and the associated certifications/validations conducted by other Participants and to provide for other related cooperative activities.

The Common Criteria standards are today under the ISO 15408 set of standards.

Further information on the Common Criteria including access to the reports for each level of certification achieved for a multitude of IBM products can be found at http://www.commoncriteriaportal.org/

2.5.2 Practical purpose for a Common Criteria evaluation

Evaluation of the security products first objective is to support the procurement of IT products with security features that meet in a consistent and stable way the consumers expectation.

As a by-product of this objective it fosters development practices of these features so that they can pass the evaluation, and provides the structural base for any other specific evaluation, such as an auditor might have.

2.5.3 The Common Criteria evaluation model

The scope of the common criteria evaluation addresses the implementation of security related functions in a product, that is the functions supporting data integrity and confidentiality, access control, etc. The evaluation in itself is independent of the level of implementation in the product, that is it could be hardware, firmware or software functions. The evaluation is also technology-agnostic.

The protection profiles

The Common Criteria model aims at evaluating the product using predefined sets of user requirements which constitute standardized test cases. They are used for rating the security properties of the product. Many Common Criteria protection profiles have been registered today that serve many purposes. Among the prevailing purposes with relation to testing the security functions imbedded in a software product or an operating system are:

- The CAPP (Controlled Access Protection Profile) which addresses the functions supporting the implementation of discretionary access controls.
- The LSPP (Labelled Security Protection Profile) focuses on the implementation of mandatory access controls using the security levels and security labels approach.

> **Note:** The Common Criteria LSPP profile was retired before the evaluation of z/OS V1R10. Requirements that are unique to a multilevel-secure environment, that were previously covered by the LSPP profile, are identified as applying to “Labeled Security Mode” in the Security Target and this chapter.

> **Important:** It is important for consistently assessing the results of the Common Criteria evaluation of a product to know exactly what protection profiles were used for this evaluation.

Some Common Criteria terminology

A very specific terminology is used to describe the entities comprised in the Common Criteria evaluation model. Here are a few terms often found, in addition to the protection profiles above, in the certification related documents:

- The Target of Evaluation (TOE) which is the product to be evaluated using the protection profiles
- The Security Target (ST) is a document that identifies the security properties of the TOE using the Common Criteria standard catalog of security functions. This will be used to rate the product against the standardized security functions.
- Evaluation Assurance levels (EAL) this is the rating given to the evaluated product using a “package” of criteria, that includes the protection profiles but also considerations on the context of use of the product and the depth and rigor of the development and maintenance processes. There are today 7 levels of EAL that are explained in, “Evaluation Assurance Levels (EAL)” on page 30.

2.5.4 The evaluation process

The Common Criteria evaluation process addresses the following items:

- The analysis of the manufacturer’s processes and procedures with evidence that the procedures are applied during the product design, build and release cycles.
- Theoretical analysis of the TOE through its design and how the design matches requirements.
- Verification of mathematical proofs, if any, produced during the design stage
- Theoretical vulnerability analysis

- Analysis of the product's user documentation
- Independent functional testing that may include penetration testing

The functions being tested

The Common Criteria model uses a standardized set of security functions, the Security Functional Classes, that we list below with the acronyms used in the certification documents. Not all the classes might be used for evaluating a specific TOE.

- Security Audit (FAU
- Communications (FCO)
- Cryptographic Support (FCS)
- User Data Protection (FDP)
- Identification & Authentication (FIA)
- Security Management (FMT)
- Privacy (FPR)
- Protection of the Trusted Security Functions (FPT)
- Resource Utilization (FRU)
- TOE Access (FTA)
- Trusted Path (FTP)

Evaluation Assurance Levels (EAL)

There are currently 7 levels of assurance, from 1 (the lowest) to 7.

Apart from the pure results of the evaluation process, an EAL assessment examines contextual factors in the normal use of the product such as a risk assessment regarding the assets it should protect and the threats that one can expect in this context of use.

We briefly describe here the 7 levels of evaluation assurance:

- EAL 1 - Functional test - The testing is performed without assistance from the product's development team
- EAL 2 - Structural Test - more aspects of the product and its development and manufacturing processes are looked at, with the help of the product's developers.
- EAL 3 - Methodical Test and Check - The design of the product is looked at for appropriate security considerations - The depth of functional testing and examination of the processes is increased with respect to EAL 2.
- EAL 4 - Methodical Design, Test and Review - The analysis goes deeper than for EAL 3, an informal security policy model of the product is also requested.
- EAL 5 - Semiformal, Design and Test - At this level more stress is put on vulnerability analysis and testing, along with an assessment of the rigor of development practices.
- EAL 6 - Semiformally Verified Design and Testing - Even more vulnerability analysis and testing; The development process goes under a semi-formal examination.
- EAL 7 - Formally Verified Design and Testing - This is the highest assurance level that can be achieved. High resistance to penetration is required from the product. There is also a requirement for extended test results, both by the product developers and by the independent organization.

2.6 Trusted programs

Some programs executing in z/OS or z/VM require for their operations to be granted special privileges that are not normally given. If these privileges are misused they can compromise the integrity and the security of the operating system. In discussing the concepts of “statements of integrity” (see 2.4, “Statements of Integrity”) we noted that there are two classes of programs. While most programs do not have access to operating system level functions there are some that do need this level of access. Such programs have to be worthy of the same level of trust as the operating system itself. This section discusses such programs.

Trusted programs and the trusted base

IBM produces many programs which make use of the trusted capabilities of system levels programs. In z/OS these programs are granted access to APF authorization (see Chapter 7.2.3, “Authorised Program Facility” on page 151 for details), while in z/VM the authority is granted via CP privilege classes (see “z/VM privileges” on page 114 for further details).

The reasons why programs need such levels can be many. However, one of the more significant reasons is so that these programs can perform authentication checks for identity and produce control blocks which can be trusted as representing the true identity of individuals and processes. If such control blocks could be produced by any program then they could not be trusted by all other programs.

Assessing the trustworthiness of an authorized program

In general the difficulty of assessing the trustworthiness of an authorized program will vary with the origin of the program: it would probably be a straight forward task to establish the trustworthiness of an in-house authorized program, whereas authorized programs obtained from vendors may need to go under extensive testing and deep analysis before being granted the trusted status.

However, IBM has many years of experience in producing such program code and knows the rules which must be followed in order to ensure that exposures are not introduced. These rules are well documented inside IBM laboratories and are followed. So well designed and well developed programs are the key to avoiding such pitfalls.

2.7 Interoperability

In order for the IBM mainframe to exist in today’s computing world it must be capable of operating with and to the standards used by the industry. This section highlights some of the more significant points at which the System z operating systems and other software comply with and cooperate with those standards.

2.7.1 An important set of universally adopted standards

It is obvious that platforms and products interoperability in today’s heterogeneous world can be achieved only by supporting commonly adopted standards for communication protocols and data formats. Likewise portability of applications that use specific security services requires the commonly adopted APIs for these services to be implemented on the different platforms that are candidates for hosting these applications.

To name a few of very well known and widely implemented standards:

- LDAP (Lightweight Directory Access Protocol)
  LDAP addresses both the structure and format of data kept in a repository and the client-server protocol to be used to access these data. LDAP is itself derived from the X.500 standards, written in the 1980’s to address both data format, repository structures

and access protocols to network accessible repositories. The LDAP protocol is implemented on TCP/IP only.

LDAP in itself does not address security functions beyond the authentication of the directory users and their authorization to use the directory data. However LDAP is linked to installations security in several ways: many systems today, for instance, are using an LDAP directory for their user registry, and in some cases for their authorization rules data base.
Many RFCs have been issued that pertain to the LDAP operations and data formats, among them RFC 2251 and RFC 3377 address the Lightweight Directory Access Protocol V3 and RFC 1823 the LDAP client API.

- X.509
The X.509 standards are in use today to describe the format of digital certificates and certificate revocation lists, and how they should be used in the context of a Public Key Infrastructure (PKI). Since 1995, the PKIX IETF working group has provided widely adopted RFCs related to the miscellaneous issues being faced when operating a Public Key Infrastructure where X.509 digital certificates are used.
The use of digital certificates entails the use of cryptographic algorithms, as certificates carry an asymmetric public key. Certificates are in use today in many secure protocols where cryptographic processes are involved, for instance, in strong user authentication or data integrity checking.
The RFC 2459 specifies the use of X.509 digital certificates and certificate revocation lists in a PKIX Public Key Infrastructure.

- SSL and TLS
Secure Socket Layer is a secure protocol originally developed by the Netscape Company in the early 1990's. It protects TCP sockets based communication using cryptographic techniques for the authentication of the communicating parties, and the integrity and confidentiality of the transferred data. SSL has been stabilized at the SSL V3 level since 1996.
As of today SSL is superseded, when designing new applications, by TLS (Transport Layer Security) - TLS can be thought of an IETF standardized version of SSL, with specific improvements. Note however that, although SSL and TLS are not compatible protocols, the vast majority of today's TLS-enabled applications are designed to fall back to SSL if the communicating partner only supports SSL.
TLS is specified in RFC 2246.

- Kerberos
Kerberos is an authentication protocol intended for authentication of clients and servers in a non-secure TCP/IP network. Kerberos is based on the use of symmetric cryptography, as opposed to SSL/TLS which uses asymmetric algorithms. Optionally, the Kerberos protocol can also provide cryptographic session keys that applications use for data confidentiality and integrity.
Kerberos is described in RFC 1510.

- IPSec
IPSec, for *Internet Protocol Security,* is a secure set of protocols used to protect any stream of IP packets transferred between two TCP/IP stacks. As for SSL/TLS the protection is based on the use of cryptographic techniques for mutual authentication of the communicating parties, and confidentiality and integrity of the exchanged data. Note however that the IPSec protocol is performed by the TCP/IP stack at the network layer, whereas SSL/TLS is performed by the application at the transport layer.
The IPSec specifications are covered by many RFCs, mainly the RFC 2401 to RFC 2410 series.

- Cryptographic algorithms
The secure protocols mentioned above exploit cryptographic processes and cryptographic

algorithms. Their interoperability also depends on the use of public and commonly adopted algorithms. Examples of very well known public algorithms are:

- Data Encryption Standard (DES)
  DES is a symmetric algorithm developed in the 1970's, heavily used in the Industry. However the key length of DES (56 bits) is deemed to be too short to offer proper protection of data today, with respect to the current state of the art in computer technologies. Users have been encouraged for the past few years to migrate to Triple-DES which uses keys of 112 bits or 168 bits.
- Advanced Encryption Standard (AES) has been designated as the successor of DES and Triple-DES. It is a symmetric algorithm that operates with key lengths of 128, 192 or 256 bits. Triple-DES users are progressively shifting today to the use of AES.
- RSA (Rivest - Shamir - Adleman) is an asymmetric algorithm that can be used to encrypt or sign data. RSA works with key lengths such as 512 bits, 768 bits, 1024 bits, 2048 bits and 4096 bits. RSA is the archetype of asymmetric algorithms and is widely in use today, in particular with the SSL/TLS protocols.
- Message Digest 5 (MD5), Secure Hash Algorithm (SHA) are hash algorithms that are used for data integrity checking and digital signature. MD5 yields a hash value of a fixed length of 128 bits, SHA comes with different lengths of the hash value: 160, 224, 256, 384 or 512 bits. The current terminology designates the 160-bit hash value algorithm as SHA-1, all the other hash lengths are comprised in the generic "SHA-2" designation.

2.7.2 The role of the mainframe in a Security Architecture

The applications running in System z are obviously consumers of security services. These services are provided by the operating system (e.g. z/OS, z/VSE or Linux) themselves or by runtime environments created by middleware software such as WebSphere® Application Server. However System z can also act as a security services provider to other platforms in an installation. A few of the services provided by System z are:

- LDAP directory services
  These services can be hosted by Linux for System z or z/OS. In the z/OS case the LDAP protocol is also extensively used to provide remote access to other services besides the simple data repository service.
  Note that LDAP services are also implemented in z/VM, starting with z/VM 5.3.
- X.509 (PKIX) Certificate Authority
  There are today on the market Certificate Authority software products that can be hosted by Linux for System z.
  z/OS comes with a complete set of Certificate Authority functions: the z/OS PKI Services. The z/OS PKI Services can be used to provide Certificate Authority services within one organization or across several organizations.
- Kerberos Key Distribution Center (KDC)
  Likewise, there are Kerberos KDC products on the market that can be hosted by Linux for System z.
  Again, z/OS comes with an imbedded Kerberos KDC support: the z/OS Network Authentication Service. z/OS can therefore provide Kerberos tickets to any platform that supports the Kerberos V5 protocol.

These services are provided as standard features in z/OS. However, it is up to each enterprise to choose whether to make use of them. Providing these services from a z/OS system ensures that the customer benefits from the platform's intrinsic reliability and security features.

Part 2

Technical view

This Part provides a more in depth look at Security on the IBM Mainframe.

3

z/Architecture: Hardware and z/OS Concepts

In this chapter, we provide an overview of z/Architecture® from a hardware and z/OS operating system stand point.

Going through this chapter you will be able to understand how System z manages the z/OS storage, as well as capture how workload separation is provided by z/OS Address Spaces.

3.1 System components

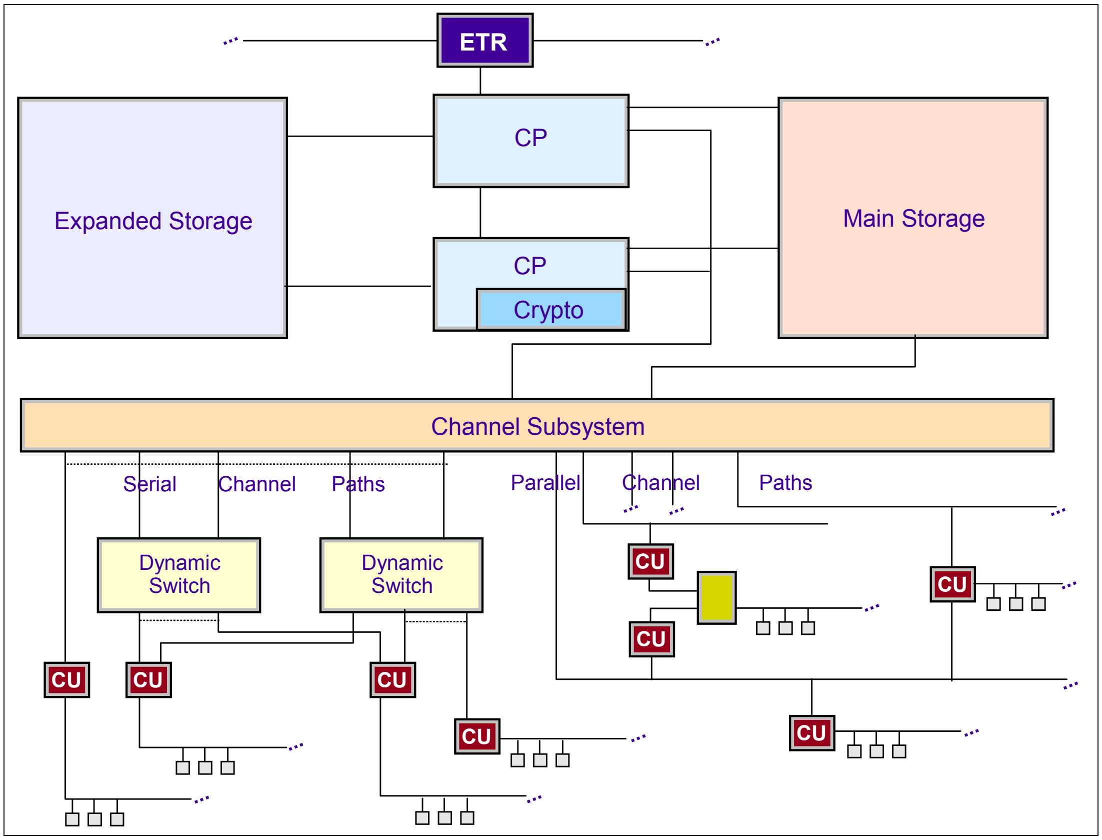


Figure 3-1 System components

Server components

Generically, a system consists of the following:

- Main storage.
- One or more central processing units—previously known as CPU. System z uses the term central processor (CP).
- Operator facilities (Service Element, which is not represented in Figure 3-1).
- A channel subsystem (formed by SAPs and channels).
- I/O devices (for example, disks also called DASD, tape, printers, teleprocessing); I/O devices are attached to the channel subsystem through *control units*. The connection between the channel subsystem and a control unit is called a *channel path*.

System Assist Processor (SAP)

System Assist Processor (SAP) is exactly the same as a central processor, but with a different microcode loaded. We will describe microcode ahead. The SAP acts as an offload engine for the CPs. Different server models have different numbers of SAPs. The SAP relieves CP involvement, thus guaranteeing one available channel path to execute the I/O operation. In other words, it schedules and queues an I/O operation, but it is not in charge of the movement between central storage (CS) and the channel.

Channels

A *channel* assists in the dialog with an I/O control unit, to execute an I/O operation—that is, the data transfer from or to memory and the device. Previously, there was no need for channels because only one process at a time was loaded in storage. So if this process needed an I/O operation, the CP itself executed it—that is, it communicated with the I/O control unit. There was no other process to be executed in memory. However, now that we are able to have several processes in memory at the same time (multiprocessing), using the CP to perform the I/O operations is inefficient. The CP is an expensive piece of hardware, and other independent processes may require processing. For this reason, the concept of using channels was introduced.

Channel paths

A *channel path* employs either a parallel-transmission electric protocol (old fashion) or a serial-transmission light protocol and, accordingly, is called either a *parallel channel path* or a *serial channel path*. For better connectivity and flexibility, a serial channel may connect to a control unit through a *dynamic switch* that is capable of providing multiple connections between entities connected to the ports of the switch (that is, between channels and I/O control units).

Expanded storage

Expanded storage is a sort of second level memory introduced because of the architecture limitation of the 2 GByte size of central storage per z/OS image. It is not available in z/Architecture, where this 2 GB limitation does not exist anymore.

Crypto

To speed up cryptographic computing, a cryptographic facility is included in each central processor. This facility is called Central Processor Assist for Cryptographic Functions. The IBM common cryptographic architecture (CCA) defines a set of cryptographic functions, external interfaces, and a set of key management rules which pertain to both the Data Encryption Standard (DES)-based symmetric algorithms and the Public Key Algorithm (PKA) asymmetric algorithms.

ETR

An *external time reference* (ETR) may be connected to the server to guarantee time synchronization between distinct servers. The optional ETR cards provide the interface to IBM Sysplex Timers, which are used for timing synchronization between systems in a sysplex environment.

3.2 z/OS storage concepts

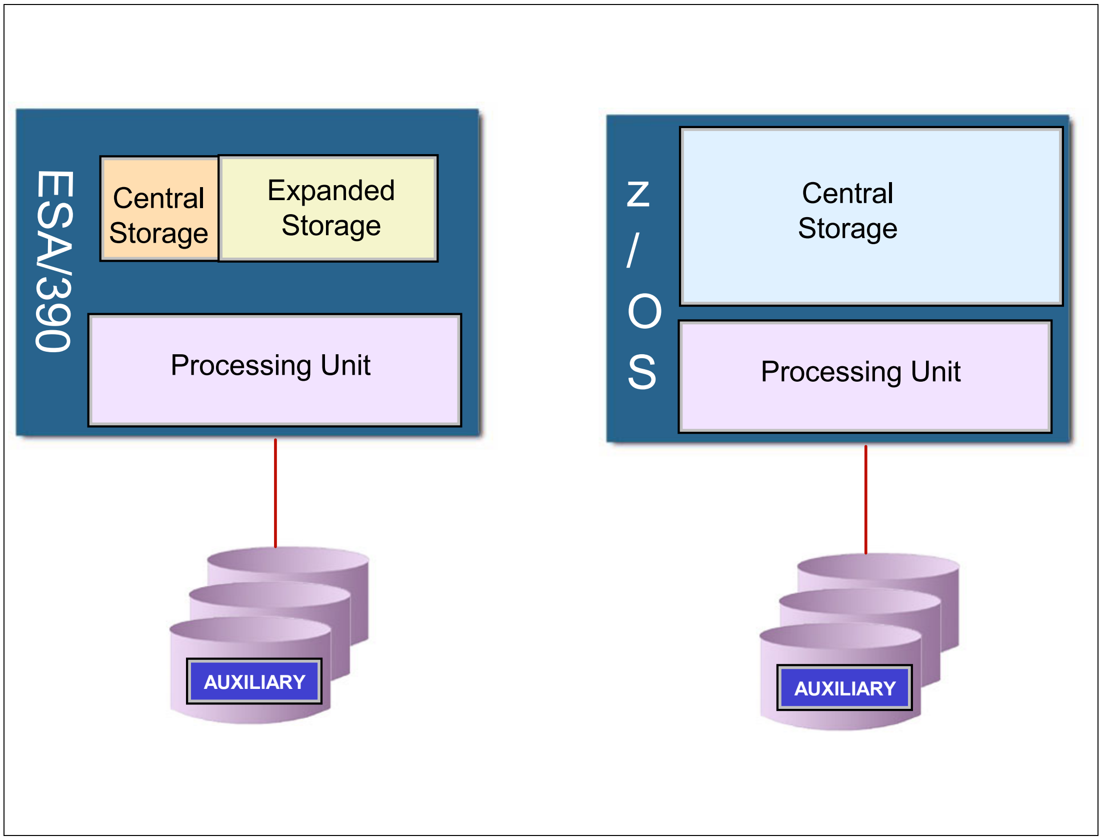


Figure 3-2 Overview of processor storage

3.2.1 Processor storage overview

Processor storage on the previous mainframe architectures (/370-XA until ESA/390) consisted of central storage plus expanded storage. The terms central storage, main storage, real storage, and memory always refer to *physical* memory, which is the one that resides within the processor cage. The central storage has the following characteristics:

- The processor is directly accessible (for programs and data).
- It is volatile and fast when compared with magnetic storage (DASD, tape).

Expanded storage (with the same technology of central storage) provided a way to relieve the performance constraint imposed by the 31-bit real address size in a logical partition that limited central storage to 2 GB. Expanded storage served as a high-speed backing store for paging and for large data buffers.

In z/Architecture, there is no expanded storage because the 31-bit real address limitation is relieved to a 64-bit real address (up to 16 Exa real addresses). Currently, the largest mainframe (the z10-EC) has a total central storage capacity of 1.5 TBytes.

The system initialization process begins when the system operator selects the LOAD function at the system console. This causes an initial program load (IPL), which is equivalent to a boot in other platforms. z/OS locates all of the usable central storage that is online and available in the IPLed logical partition, creating a virtual storage environment for the building of various system areas. z/OS uses central storage to map the virtual storage, which implies allocating and using auxiliary storage.

- **Central storage**

  Central storage, also referred to as main storage, provides the system with a volatile memory space that is directly addressable by the processor, with fast access for the electronic storage of data.

  Because of the volatile property of central storage, modern mainframes have an Internal Battery Feature (IBF) that keeps central storage running so operators can perform normal shutdown procedures in power failure situations.

  Both data and programs must be loaded into central storage (from magnetic devices such as disks and tapes) before they can be processed by the processors. The maximum central storage size per logical partition is restricted by the system architecture, as follows:

  – In System/370 architecture, the maximum is 16 MB.
  – From S/370-XA architecture to ESA/390 architecture, the maximum is 2 GB.
  – In z/Architecture, the maximum is 16 EX (exabytes), however it is limited by the z/OS level to 4 TBytes.

- **Auxiliary storage**

  Auxiliary storage consists of z/OS paging data sets (files) located in direct access storage devices (DASD). Note that DASD in mainframe terminology is referred to as “disk” on other platforms. DASD is a non-volatile magnetic memory. Paging data sets are used to implement virtual storage, which contain the paged-out portions of all virtual storage address spaces.

3.2.2 The address space concept

The System/370 was the first IBM architecture to use virtual storage and address space concepts emulating the first virtual system. The address space is a set of contiguous virtual addresses available to a program instructions and its data. The range of virtual addresses available to a program starts at 0 and can go to the highest address permitted by the operating system architecture. The address space size is decided by the length of the fields that keeps such addresses. Because it maps all of the available addresses, an address space includes system code and data as well as user code and data. Thus, not all of the mapped addresses are available for user code and data. The S/370 architecture used 24 bits for addressing. So, the highest accessible address in the MVS/370 was 16 Megabytes, which was also the address space size.

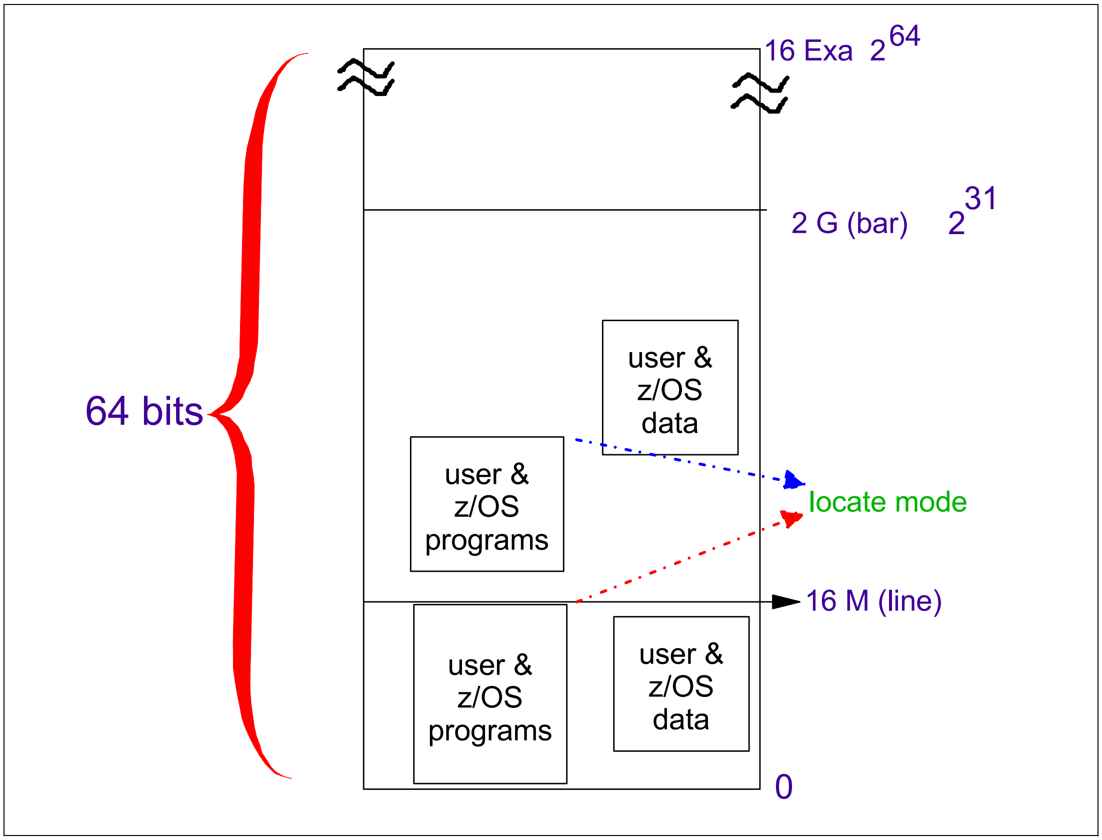


Figure 3-3 Address space concept

With MVS/XA, the XA architecture extended to 31 bits for addressing and the address space size went from 16 MB to 2 GBytes. The 16 MByte address became the division point between the two architectures and is commonly called the *line*.

For compatibility, programs running in MVS/370 should run in MVS/XA, and new programs should be able to exploit the new technology. So, the high-order bit of the address (4 bytes) is *not* used for addressing, but rather to indicate to the hardware how many bits are used to solve an address: 31 bits (bit 32 on) or 24 bits (bit 32 off).

The z/Architecture extended to 64 bits with z/OS and the address space size went from 2 GB to 16 Exabytes. The 2 GByte address is called the *bar*. The addresses above the bar are used for data only.

Addressing mode and residence mode

With the MVS/XA came the concept of *addressing mode* (AMODE). AMODE is a program attribute to indicate which hardware addressing mode should be active to solve an address; that is, how many bits should be used for solving and dealing with addresses.

- AMODE=24: Indicates that the program may address up to 16 MByte virtual addresses.
- AMODE= 31: Indicates that the program may address up to 2 GByte virtual addresses.
- AMODE= 64: Indicates that the program may address up to 16-ExaByte virtual addresses (only in z/Architecture).

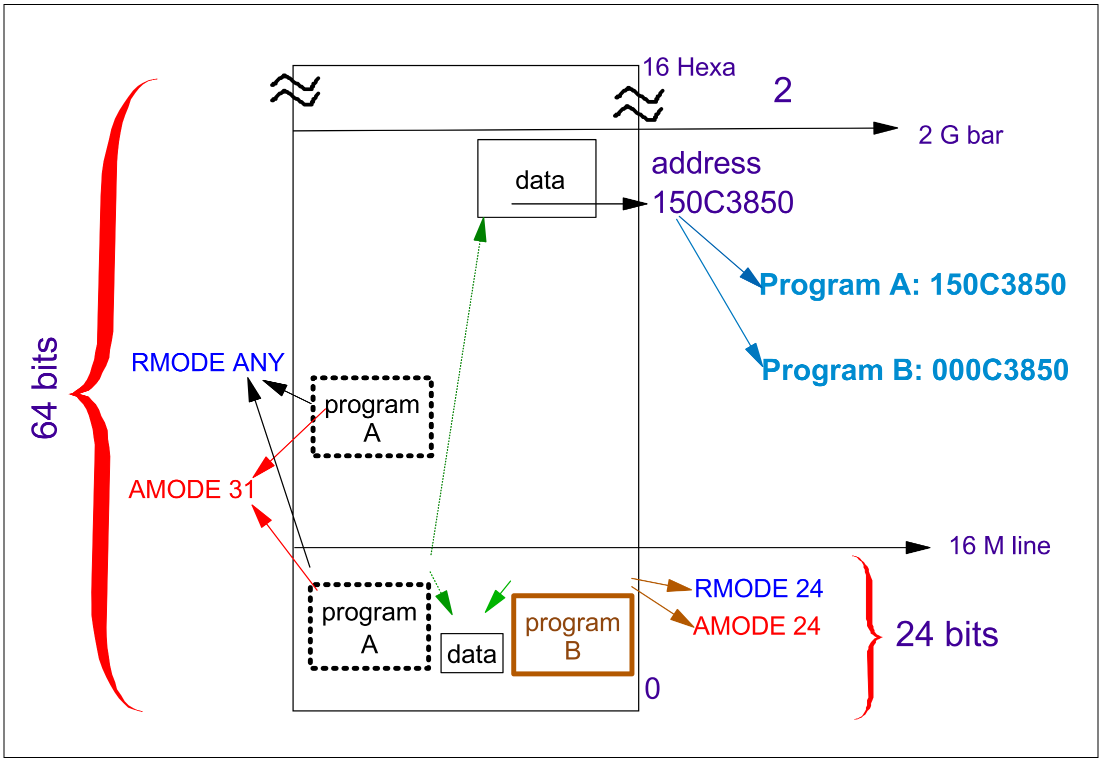


Figure 3-4 Addressing mode and residence mode

The concept of *residence mode* (RMODE) is used to indicate where a program should be placed in the virtual storage (by z/OS program management) when the system loads it from DASD, as explained here:

- RMODE=24: Indicates that the module *must* reside below the 16 MByte virtual storage line. Among the reasons for RMODE24 is that the program is AMODE24, the program has control blocks that must reside below the line.
- RMODE= ANY: Indicates that the module may reside anywhere in virtual storage, but preferentially above the 16 MByte virtual storage line. Because of this, such an RMODE is also called RMODE 31. Note that in z/OS, there is no RMODE=64, because the virtual storage above 2 G is not suitable for programs, only data.

AMODE and RMODE are load module attributes assigned when load modules are created by the Binder program and are placed in load module's directory entry in a partitioned data set.

Storage managers

In a z/OS system, storage is managed by the following components:

- **Virtual Storage Manager (VSM):** The VSM is the z/OS component that manages virtual storage. Its main function is to control the use of virtual storage addresses. The existence of an address space does not imply that all virtual addresses are automatically available to programs. Virtual storage addresses must be requested from and returned to VSM by programs through the use of macro instructions. Each installation can use virtual storage parameters to specify how certain virtual storage areas are to be allocated to programs.
- **Real Storage Manager (RSM):** RSM is the z/OS component that controls the allocation of central storage frames. RSM acts together with the Auxiliary Storage Manager (ASM) to support the virtual storage concept. It directs the movement of pages between central storage and auxiliary storage. RSM builds segment and page tables. These tables are

used to translate a virtual address to a real address. It also acts with VSM to ensure that a page that was requested via a macro is backed up in a central storage frame.

- **Auxiliary Storage Manager (ASM):** ASM is a z/OS component responsible for transferring virtual pages between central frames and auxiliary storage slots (page data sets). This is done as either a paging operation (one page at time) or as a physical swapping operation (an address space, all pages at a time). ASM manages the transfer by initiating the I/O and by maintaining tables to reflect the current status of the slots. This status includes the location of each page in each slots. To page efficiently, ASM divides the pages into classes, namely PLPA, common, and local. There is at least one page data set for each class. In addition, output to virtual I/O (VIO) devices may be stored in local paging data sets. VIO are user temporary data sets allocated in central and auxiliary storage.

Paging and swapping

Paging is the movement of pages between central storage frames and auxiliary storage slots.

There are two types of paging operations:

- Page-in, which flows from a slot to a frame. It is caused by a page fault. A page fault is an interrupt caused by the hardware in charge of translating a virtual address into a real address. The page fault happens because the page is not currently mapped to a frame. RSM gains control and, through ASM, provides a page-in operation to retrieve the page from auxiliary storage.
- Page-out, which flows from a frame to a slot. It is caused when a changed page needs to be stolen from central storage because this memory is under contention. RSM calls ASM to schedule the paging I/O necessary to send these pages to auxiliary storage.

Swapping is the primary function used by WLM (a z/OS component in charge of performance) to exercise control over the distribution of resources and system throughput. One reason for swapping is, for example, pageable storage shortages. There are two types of swapping:

- Physical swapping: Transferring all pages in an address space between central storage and auxiliary storage.
  - A physical swapped-in address space is an active one (its programs can be executed) that has pages in central storage and pages in auxiliary storage.
  - A physical swapped-out address space is an inactive one having all pages in auxiliary storage, so it cannot execute its programs until it is swapped-in.
- Logical swapping: To reduce the processor and channel subsystem overhead involved during a physical swap needing to access auxiliary storage, WLM performs logical swaps where possible. In a logical swap, LSQA fixed frames and recently referenced frames are kept in central storage (in contrast to physical swaps, where these frames are moved to auxiliary storage). Since z/OS 1.8, all swaps are logical.

Auxiliary page data sets

Auxiliary page data sets are formatted in slots; they should contain pages that for some reason should not stay in central storage frames. As mentioned, ASM divides the pages of the system into classes: private, CSA, and PLPA. Based on these classes, there are three types of page data sets.

- PLPA page data set: This unique and required page data set contains pageable link pack area pages.
- Common page data set: This unique and required page data set contains the CSA non-fixed virtual pages of the system common area.

- Local page data sets: These contain the private area pages of all address space pages, data spaces, and any VIO data sets.

Peaks in central storage demand may occur during system operation, resulting in heavy use of local page data set slots. To address this situation, local page data sets can be dynamically added to and deleted from the paging configuration without re-IPLing the system.

31-bit address space map

Since the introduction of MVS/XA architecture, the address space size is 2 G addresses because the fields keeping the virtual addresses are 31 bits in size.

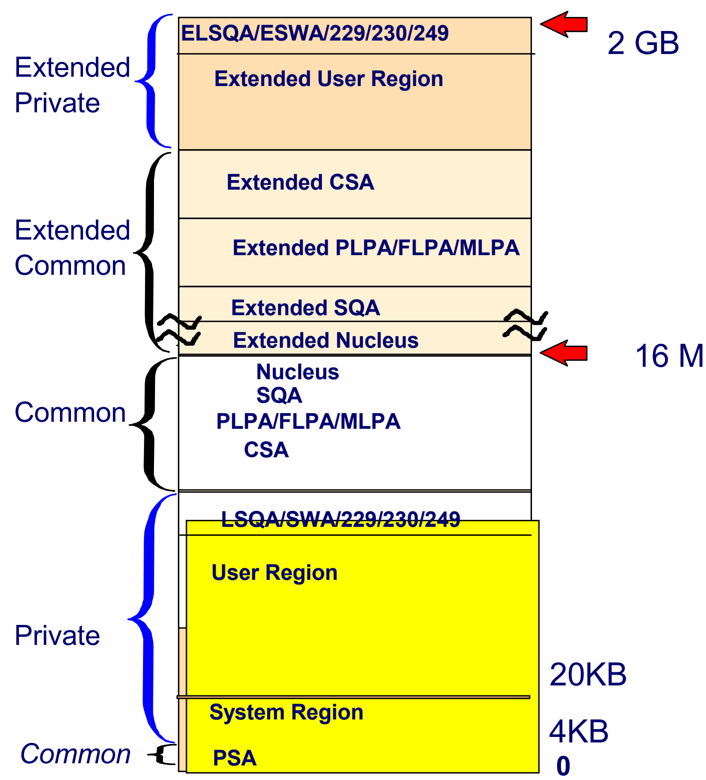


Figure 3-5 31-bit address space map

The virtual address space is divided into areas (sets of addresses) according to their use. Virtual storage allocated in each address space is divided between system requirements and user requirements. z/OS itself requires space from each of the basic areas. Each virtual address space consists of:

- The common area below 16 MByte addresses
- The private area below 16 MByte addresses
- The extended common area above 16 MByte addresses
- The extended private area above 16 MByte addresses

Most of z/OS areas exist both below and above 16 MByte address, providing an environment that can support both 24-bit and 31-bit addressing. However, each area and its counterpart above 16 MByte address can be thought of as a single logical area in virtual storage.

The common virtual storage area

The z/OS implementation of virtual storage is to have one address space per set of related programs. The advantage of this design is isolation; any error is contained in one address space and cannot be propagated to another address space. However, the system needs communication between programs from different address spaces. To provide this communication means, the common area was introduced. All address spaces in a z/OS system image share a virtual storage area known as the *common area*. That means that all address spaces programs in this z/OS access the same common data and the same common programs, with the same virtual address. The common area is created at IPL (boot) time by z/OS. The following virtual storage areas are located in the common area:

- Prefixed storage area (PSA) with 8 KBytes in z/Architecture
- Common service area (CSA)
- Link pack areas: Pageable (PLPA), Fixed (FLPA), and Modified (MLPA)
- System queue area (SQA)
- Nucleus

Each storage area in the common area (below 16 M) has a counterpart in the extended common area (above 16 M), except the PSA.

z/OS nucleus

The nucleus in the common area contains the z/OS nucleus programs (kernel) and extensions to the nucleus that are initialized during IPL processing. The nucleus contains the most important z/OS programs. The nucleus RMODE24 programs reside below the 16 M line. The nucleus RMODE31 programs reside above the 16 M line.The nucleus is always fixed in central storage; that is, no pages from the nucleus can be stolen to page data sets slots.

System queue area (SQA/ESQA)

The system queue area (SQA) is a common area containing control blocks used by z/OS to manage transaction workloads and the use of system resources. The number of active address spaces (which depends on the workload executed in the system) affects the system's use of SQA. SQA is allocated directly below the nucleus. Extended SQA (ESQA) is allocated directly above the extended nucleus. Both allocations occur at IPL time. When SQA/ESQA pages are in use, they are fixed in central storage. Ensuring the appropriate size of SQA/ESQA and CSA/ECSA is critical to the long-term operation of z/OS.

Common service area (CSA and Extended CSA)

The common service area is a common area containing control blocks used by subsystem programs such as JES2, DFSMS, and RACF, and access methods like VSAM. CSA/ECSA normally contains data referenced by a number of system address spaces, enabling address spaces to communicate by referencing the same piece of CSA data. In a sense, CSA/ECSA looks like SQA/ESQA. CSA is allocated directly below the MLPA. ECSA is allocated directly above the extended MLPA. The common service area (CSA) contains pageable and fixed data areas that are addressable by all active virtual storage address spaces.

Link pack area (LPA and Extended LPA)

The link pack area contains programs that are preloaded at IPL time in the common area. These programs can be certain z/OS routines, access methods code, other read-only z/OS programs (the ones not modified along its execution), and any read-only reenterable user programs selected by an installation. Because such code is in the common area, all these single copy programs can be executed in any address space. Their copy is not self-modifying (reentrant), meaning that the same copy of the module can be used by any number of tasks in any number of address spaces at the same time. This reduces the demand for central storage and lowers the program fetch overhead.

The LPA/ELPA size depends on the number of modules loaded in it. All modules placed in LPA are assumed to be APF-authorized. Being APF-authorized means that a program can invoke any routine which accesses protected system and private areas. It is possible to dynamically include new load modules to LPA without an IPL. The RMODE attribute of the program (load module) decides its location (that is, LPA or ELPA).

The LPA is divided into:

- Pageable LPA (PLPA/EPLPA): In this area, page faults (the page is not mapped in a central storage frame) may occur.
- Fixed LPA (FLPA/EFLPA): This area is used for modules that have to be fixed in memory, instead of pageable.
- Modified LPA (MLPA and EMLPA): The MLPA can be used at IPL time to temporarily modify or update the PLPA with new or replacement modules.

31-bit address space private area

There are two private areas: below the 16 MByte address line (PVT), and above the 16 MByte address line (EPVT). Their size is the complement of the common area's size. The virtual addresses within the private area is unique to the programs running in such areas.

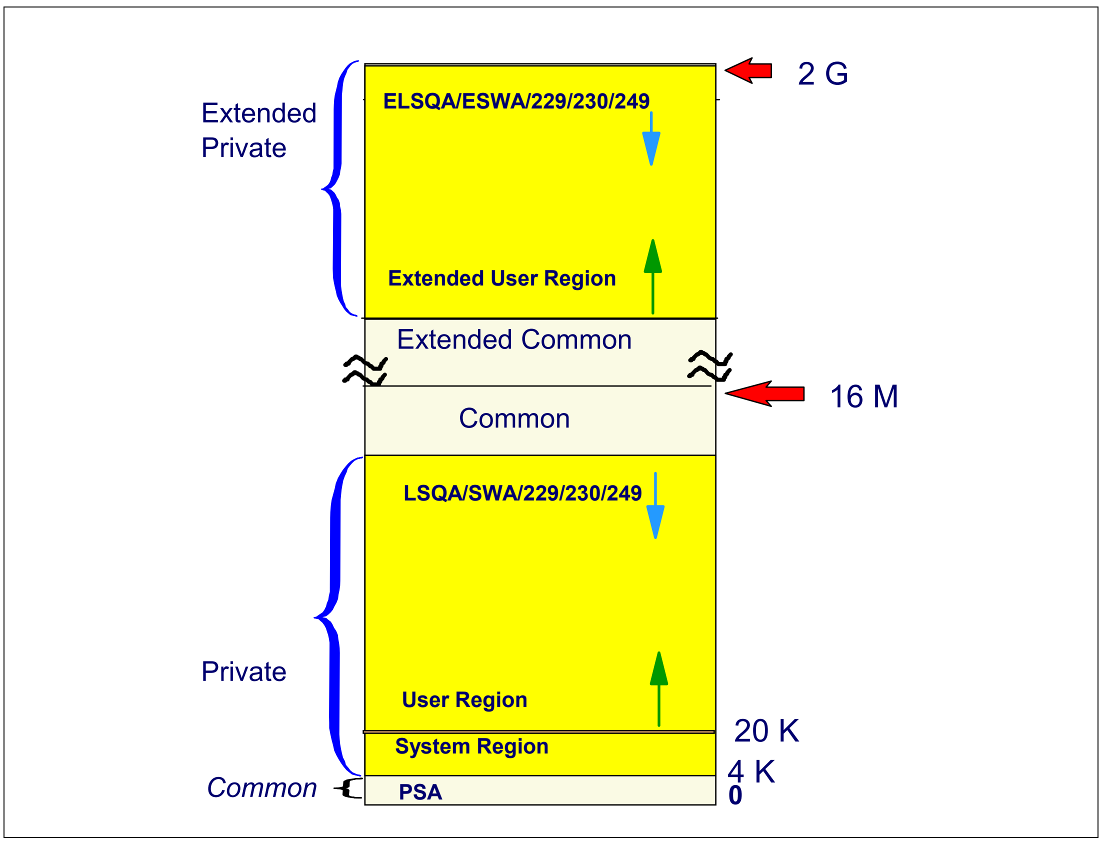


Figure 3-6 The user's private area (user region)

The private area is formed by the following areas:

- Subpools 229, 230, and 249

  This area allows private storage to be obtained in the requestor's storage protect key. The area is used for control blocks that can be obtained only by authorized programs (as z/OS) having appropriate storage protect keys. A subpool is a virtual storage area with the same properties regarding storage key, pageable or fixed, private or common, fetch protected or not, and so on. When a program requests virtual storage addresses, it must indicate the subpool number.

- Local System Queue Area (LSQA) / Extended Local System Queue Area (ELSQA)

  This area contains tables and control blocks queues associated with the address space.

- Scheduler Work Area (SWA) / Extended Scheduler Work Area (ESWA)

  This area contains control blocks that exist from task initiation to task termination.

- A 16 KByte system region area
- User region

  This region is used for running user program applications (loaded at subpools 251/252) and storing user program data (subpools from 0 to 127).

Data spaces and hiperspace

ESA/370 and MVS/ESA introduced an horizontal growth to virtual storage with data space, a new type of z/OS address space, as well as hiperspace, a new type of service.

Data space is a type of virtual storage space with a range up to 2 GBytes of contiguous virtual storage. The virtual storage map of a data space is quite different; that is, the entire 2 GB is available for user data and does not contain specific areas as an address space. A data space can hold only data (operands accessed by instructions located in address spaces); it does not contain z/OS control blocks or programs in execution. Program code does not execute in a data space, although a program can reside in a data space as data (to be executed, however, it needs to be copied to an address space). A program can refer to data in a data space at bit level, as it does in a work file. A program references data in a data space directly, in much the same way as it references data in an address space. Before accessing data in a data space, a program must change the processor access mode; that is, it must use special assembler instructions to change its access mode.

High performance data access, known as *hiperspace*, is a kind of data space created with the same RSM services used to create a data space. It provides the applications an opportunity to use expanded storage as a substitute to I/O operations. Hiperspaces differ from data spaces in the following ways:

- Main storage is never used to back the virtual pages in hiperspace, whose pages are located in expanded or auxiliary.
- Data can be retrieved and stored between a hiperspace and a data space only using MVS services.
- Data is addressed and referred to as a 4 K block.

Although z/OS do not support Expanded Storage when running under the z/Architecture, hiperspace continues to operate in a compatible manner, that is, a hiperspace is now mapped in central storage (instead of expanded) and auxiliary storage.

Programs can use data spaces and hiperspaces to obtain more virtual storage than a single address space gives a user. It also can be used to isolate data from other tasks (programs) in the address space. Data in an address space is accessible to all programs executing in that address space. It is feasible to move some data to a database or hiperspace for security or integrity reasons. It is possible to restrict access to data in those spaces to one or several units of work.

Data spaces and hiperspaces are suitable to share data among programs that are executing in the same address space, or different address spaces. Instead of keeping the shared data in common areas, you can create a database or hiperspace for the data you want your programs to share. Use this space as a way to separate your data logically by its own particular use. You could also use it to provide an area in which to map a data-in virtual objects.

64-bit address space map

As previously mentioned, z/Architecture broke the 2 GBytes (31-bit) central storage limit and the 2 GBytes (31-bit) address limit. It expanded the limit to 16 Exabytes (64-bit).

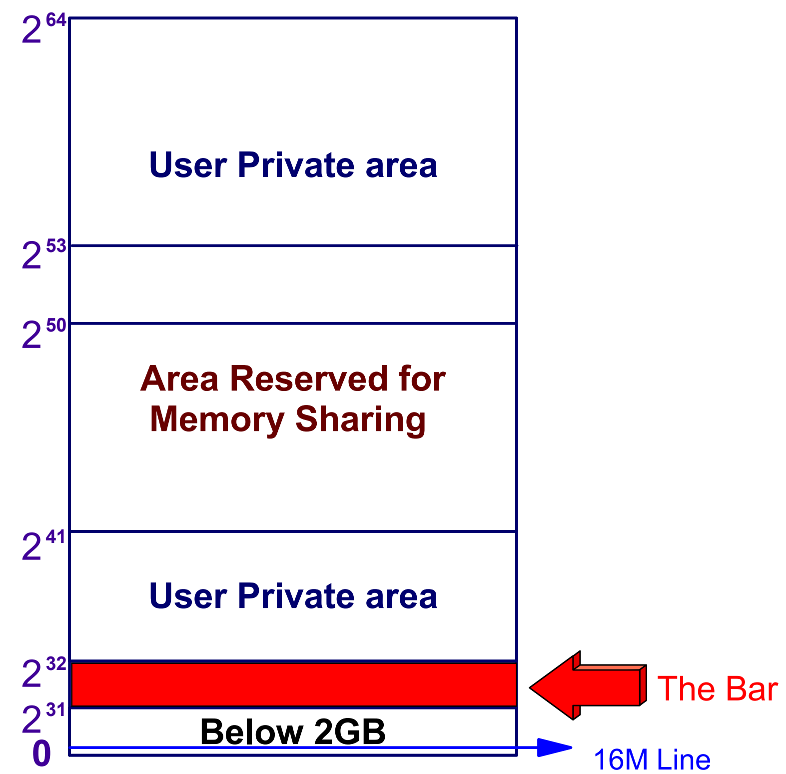


Figure 3-7 64-bit address space map

The z/Architecture supports 16 Exabyte addresses in a z/OS address space, however, any new created address space in z/OS is initialized with 2 GByte addresses (as it was previously), but with the potential to go beyond.

For compatibility, the layout of the virtual storage areas for an address space is the same below 2 G. The area that separates the virtual storage area below the 2 Gigabyte address from the user private area above is called the *bar*, as shown in Figure 3-7 above. The bar is 2 GByte addresses thick. In a 64-bit virtual storage environment, the terms “above the bar” and “below the bar” are used to identify the areas between 2**31 and 2**64-1, and 0 and 2**31-1, respectively. It is similar to the terminology that related “below the line” to 24-bit addresses, and “above the line” to 31-bit addresses.

The 64-bit address space map is shown below:

- **0 to 2**31:** The layout is the same as 31-bit address space map; see Figure 3-5 on page 45.
- **2**31 to 2**32**: From 2 GByte to 4 GByte is considered the *bar*. Below the bar can be addressed with a 31-bit address. Above the bar requires a 64-bit address.

- **2**31 - 2**41:** The Low Non-shared area starts at 4G and goes to 2**41.
- **2**41 - 2**50:** The Shared Area starts at 2**41 and goes to 2**50 or higher if requested (up to **2 **53**).
- **2**50 - 2**64:** The High Non-shared area starts at 2**50 or wherever the Shared Area ends and goes to 2**64.

The shared area may be shared between programs running in private areas from specific address spaces. In contrast, the below the bar common area is shared between all address spaces.

The area above the bar is designed to keep data (such as DB2 buffer pool and WebSphere data), and not to load modules (programs). There is no RMODE64 as a load module attribute. However, such programs running below the bar may request virtual storage above the bar and access it. In order to access such an address, the program must be AMODE64.

User private area

The area above the bar is intended for application data; no programs run above the bar. No system information or system control blocks exist above the bar, either. Currently there is no common area above the bar.

The *user private area*, as shown in Figure 3-7 on page 49, includes:

- Low private: The private area below the line
- Extended private: The private area above the line
- Low Non-shared: The private area just above the bar
- High Non-shared: The private area above Shared Area

For virtual storage above the bar, there is no practical limit to the amount of virtual address range that an address space can request. However, there are practical limits to the central storage and auxiliary storage needed to back the request. Therefore, a limit is placed on the amount of usable virtual storage above the bar that an address space can use at any one time.

Dual address space (cross memory)

Synchronous cross-memory communication enables one program to provide services synchronously to other programs. Synchronous cross-memory communication takes place between address space 2, which gets control from address space 1 when the program call (PC) instruction is issued. Address space 1 has previously established the necessary environment, before the PC instruction transfers control to an address space 2 program called a *PC routine*. The PC routine provides the requested service and then returns control to address space 1.

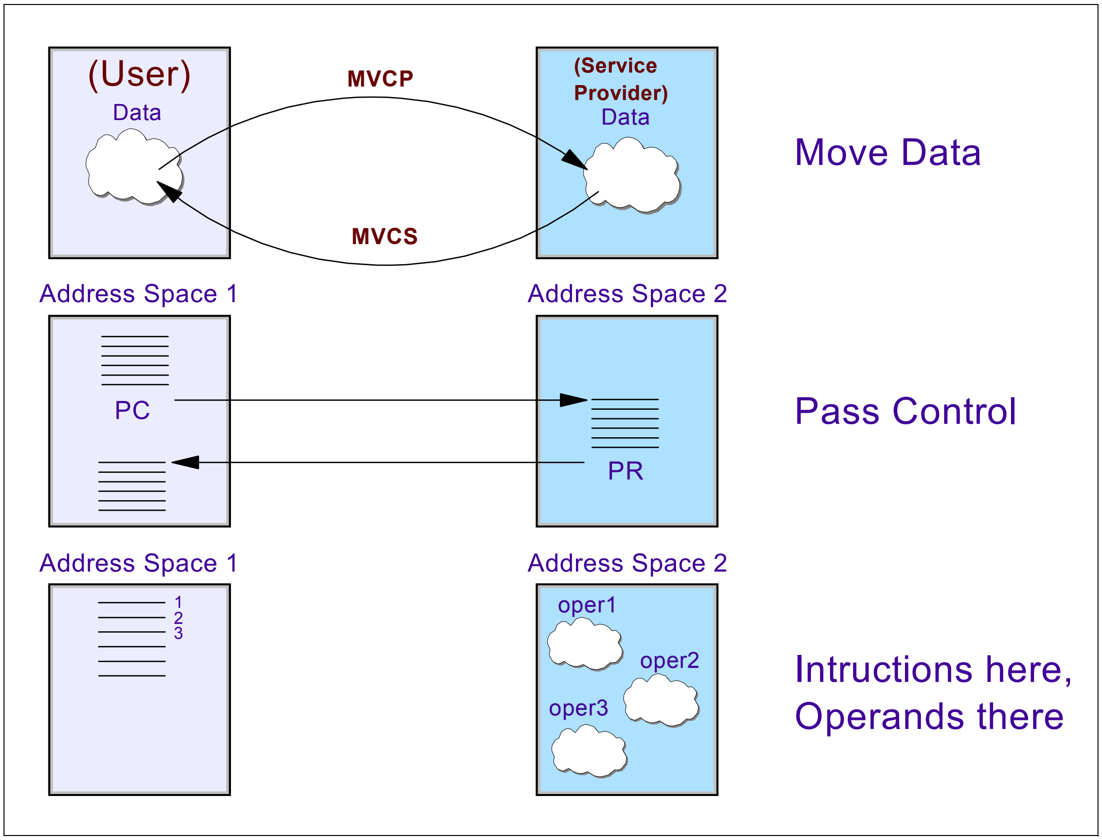


Figure 3-8 Cross memory

The user program in address space 1 and the PC routine can execute in the same address space or, as shown in Figure 3-8 above, in different address spaces. In either case, the PC routine executes under the same TCB as the user program that issues the PC. Thus, the PC routine provides the service synchronously.

Dual address space or cross-memory (XM) is an evolution of virtual storage. It has three objectives:

- Move data synchronously between virtual addresses located in distinct address spaces.

  This can be implemented by the use of the SET SECONDARY ADDRESS REGISTER (SSAR) instruction. It points to an address space and makes it secondary. A secondary address space has its Segment Table pointed by Control Register 7 instead of Control Register 1. Next, using the MOVE CHARACTER TO SECONDARY (MVCS) or MOVE CHARACTER TO PRIMARY (MVCP), the objective can be accomplished.

- Pass the control synchronously between instructions located in distinct address spaces.

  There is an instruction PROGRAM CALL (PC) able to do that. To return the origin address space, there is the instruction PROGRAM RETURN (PR).

- Execute one instruction located in one AS and its operands are located in other address space.

MVCP and MVCS

The PC routine can, if necessary, access data in the user's address space without using Access Registers. The MVCP instruction moves data from the secondary address space (the user) to the primary address space (the service provider). The MVCS instruction moves data from the primary address space (the service provider) to the secondary address space (the user). To use the MVCP or MVCS instructions, the service provider must have obtained SSAR authority to the user's address space before the PC routine receives control.

3.2.3 System initialization

The system initialization process (IPL) prepares the system control program (z/OS) and its environment to do work for the installation. The process essentially consists of:

- System and storage initialization (virtual, real and auxiliary), including the creation of the common area and the system component address spaces
- Master scheduler initialization and subsystem initialization

When the system is initialized, z/OS creates system component address spaces. z/OS establishes an address space for the master scheduler and other system address spaces for various subsystems and system components. Note that some z/OS components do not need a specific address space. Some of the system component address spaces are:

- Program call/authorization for cross-memory communications
- System trace
- Global resource serialization
- Dumping services

Initializing z/OS address spaces

When you start z/OS, master scheduler initialization routines initialize system services such as the system log and communications task, and start the master scheduler address space. Each address space created has a a number associated to it, known as the address space id (ASID). Because the master scheduler is the first address space created in the system, it becomes address space number one (ASID=1).

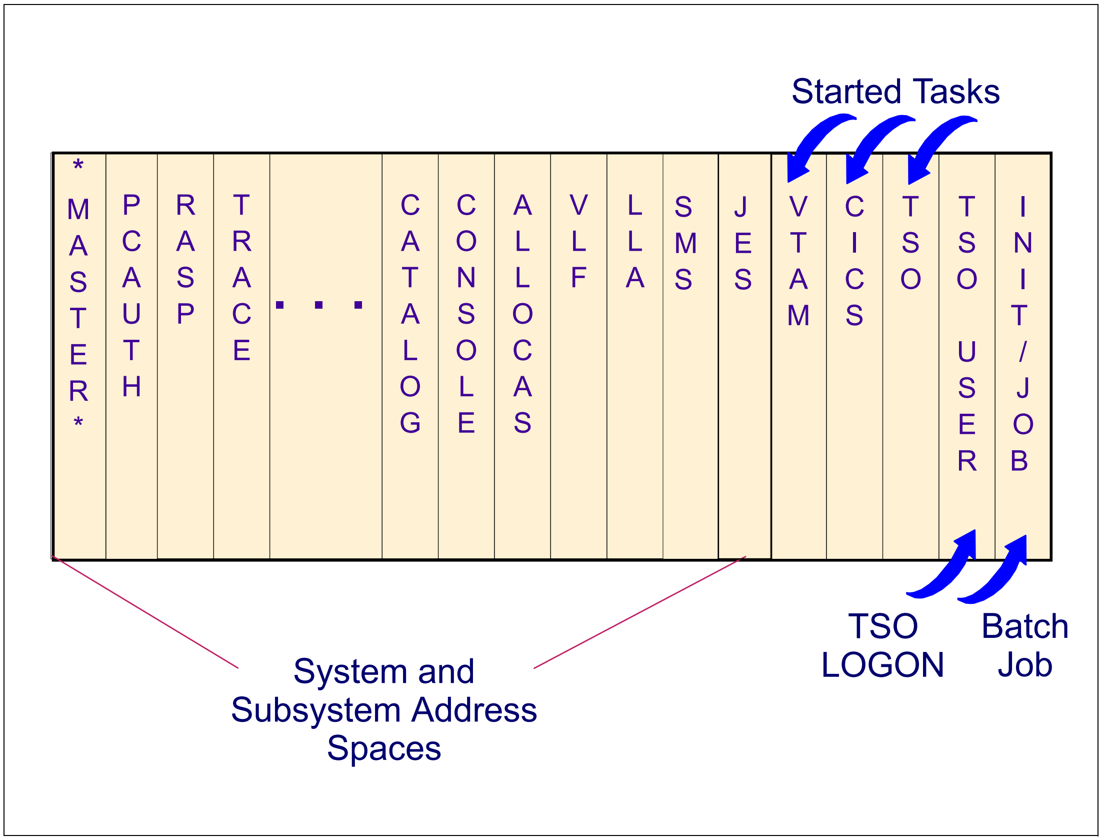


Figure 3-9 System and subsystem address spaces in z/OS

Other system address spaces are then started during the initialization process of z/OS. Subsystem address spaces are started. The master scheduler starts the job entry subsystem (JES2 or JES3). JES is the primary job entry subsystem. Then other defined subsystems are started. Figure 3-9 above shows four types of address spaces:

System The system address spaces are started following initialization of the master scheduler. These address spaces perform functions for all the other types of address spaces that are started in a z/OS system.

Subsystem A subsystem is a service provider that performs one function or many functions, but does nothing until it is requested. Subsystem initialization is the process of readying a subsystem for use in the system.

TSO logon These address spaces start when a user issues a logon to TSO/E. Each TSO user executes in a separate address space.

Batch job These address spaces run Initiator code that is in charge of allocating resources to a JCL stream that is passed to JES

Dispatchable tasks

In z/OS, processes are called *dispatchable units*, which consist of tasks and service requests. The control program creates a task in the address space as a result of initiating execution of the process.

Multiprogramming

Multiprogramming means that many tasks can be in a system at the same time, with each task running programs in its own address space (or sometimes in the same address space). In a single processor system, only one of these tasks can be active at a time. However, the active task can lose control of the processor at any time (for example, because of I/O requests that place the task in a wait state, meaning it is not a candidate to get the processor). The operating system then selects which task should get control next, based in a number called *dispatching priority*.

Multiprocessing

Multiprocessing is a logical expansion of multiprogramming. Multiprocessing refers to the execution of more than one task simultaneously on more than one processor. All processors operate under a single copy of the operating system and share the same memory.

- Each processor has a current Program Status Word (PSW), its own set of registers, and assigned storage locations.
- When a single processor shares central storage with other processors, then all of them are controlled by a single operating system copy. This is called a *tightly coupled multiprocessing complex*. When a single processor shares a common workload with others, but does not share central storage, this is called a *loosely coupled multiprocessing complex*.

3.2.4 Hardware registers

Registers

The Central Processor provides registers that are available to programs, but that do not have addressable representations in main storage. They include the current program-status word (PSW), the general registers, the floating-point registers and floating-point-control register, the control registers, the access registers, the prefix register, and the registers for the clock comparator and the CP timer.

Each Central Processor in an installation provides access to a time-of-day (TOD) clock, which is shared by all CPs in the installation. The instruction operation code determines which type of register is to be used in an operation. There are several types of registers, as explained in the following sections.

General registers

General registers (GRs) are used to keep temporary data (operands) loaded from memory to be processed or already processed. Instructions may designate information in one or more of 16 general registers. The general registers may be used as base-address registers and index registers in address arithmetic, and as accumulators in general arithmetic and logical operations.

Each register contains 64 bit positions. The general registers are identified by the numbers 0-15, and are designated by a four-bit R field in an instruction. The data is in binary integer format, also called *fixed point*. There are certain CP instructions that are able to process data stored in GRs. Its contents can also be used for the execution of a CP instruction to point to the address of a storage operand.

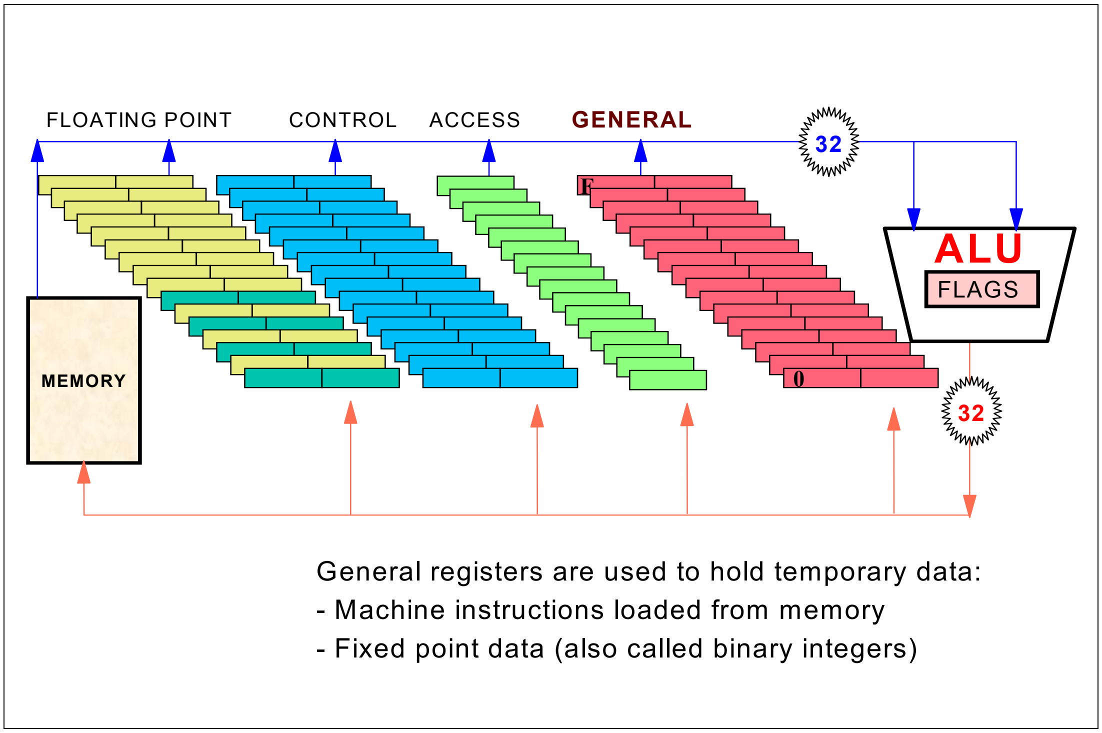


Figure 3-10 CP registers (general)

Control registers

The CP has 16 control registers (CRs), each having 64 bit positions. The bit positions in the registers are assigned to particular facilities in the system, such as program-event recording, and are used either to specify that an operation can take place, or to furnish special information required by the facility. Control Registers are registers accessed and modified by z/OS through privileged instructions. All the data contained in the CRs are architected containing information input by z/OS and used by hardware functions (such as Crypto, cross memory, virtual storage, and clocks) implemented in the server.

Access registers

Access registers (ARs) are used by z/OS to implement *data spaces* through activating the access register mode in the Central Processor. The CP has 16 access registers numbered 0-15. An *access register* consists of 32 bit positions containing an indirect specification of an address-space-control element. An *address-space-control element* is a parameter used by the dynamic-address-translation (DAT) mechanism to translate references to a corresponding address space. When the CP is in a mode called the access-register mode (controlled by bits in the PSW), an instruction B field, used to specify a logical address for a storage-operand reference, designates an access register, and the address-space-control element specified by the access register is used by DAT for the reference being made. Instructions are provided for loading and storing the contents of the access registers, and for moving the contents of one access register to another.

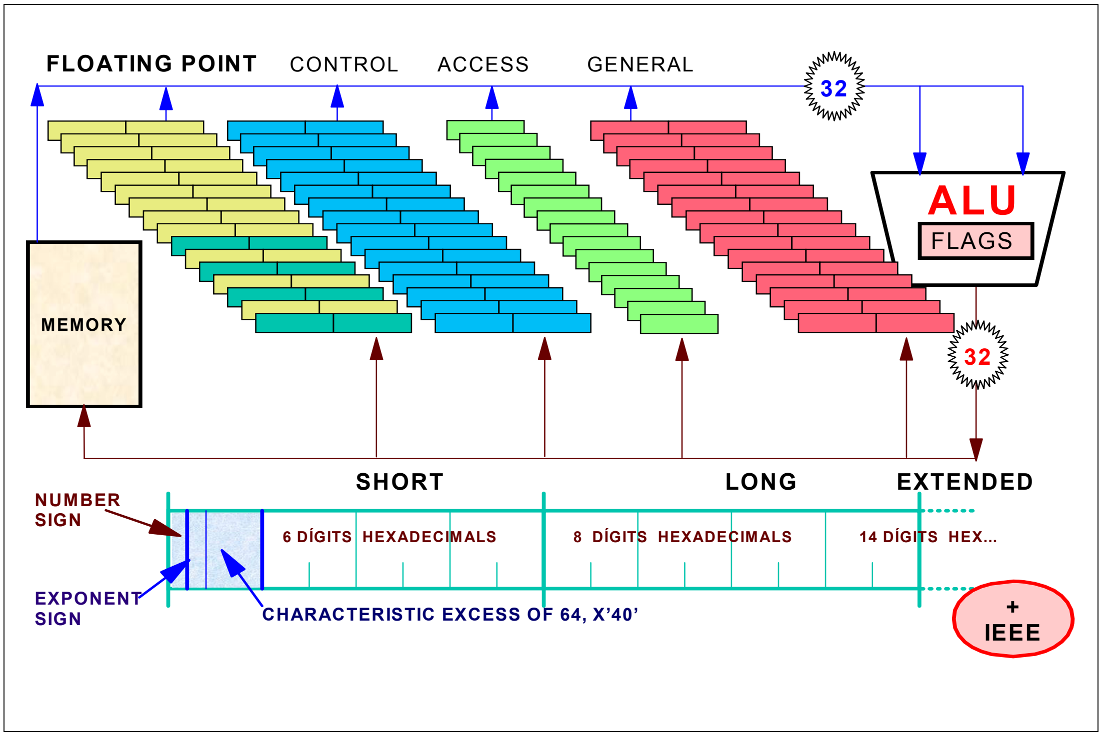


Figure 3-11 Floating point registers

Floating point registers

All floating-point instructions use the same floating-point registers. The Central Processor has 16 floating-point registers. The floating-point registers are identified by the numbers 0-15 and are designated by a four-bit R field in floating-point instructions. Each floating-point register is 64 bits long and can contain either a short (32-bit) or a long (64-bit) floating-point operand. Floating point registers (FPRs) are used to keep temporary data (operands) loaded from memory to be processed or already processed. There is also a floating-point-control (FPC) register, a 32-bit register to control the float point instructions execution. It contains mask bits, flag bits, a data exception code, and rounding-mode bits.

Program-Status Word (PSW)

The current PSW is a storage circuit located within the CP. It contains information required for the execution of the currently active program; that is, it contains the current state of a CP. It has 16 bytes (128 bits). The PSW includes the instruction address, condition code, and other information used to control instruction sequencing and to determine the state of the CP. The active or controlling PSW is called the *current PSW*.

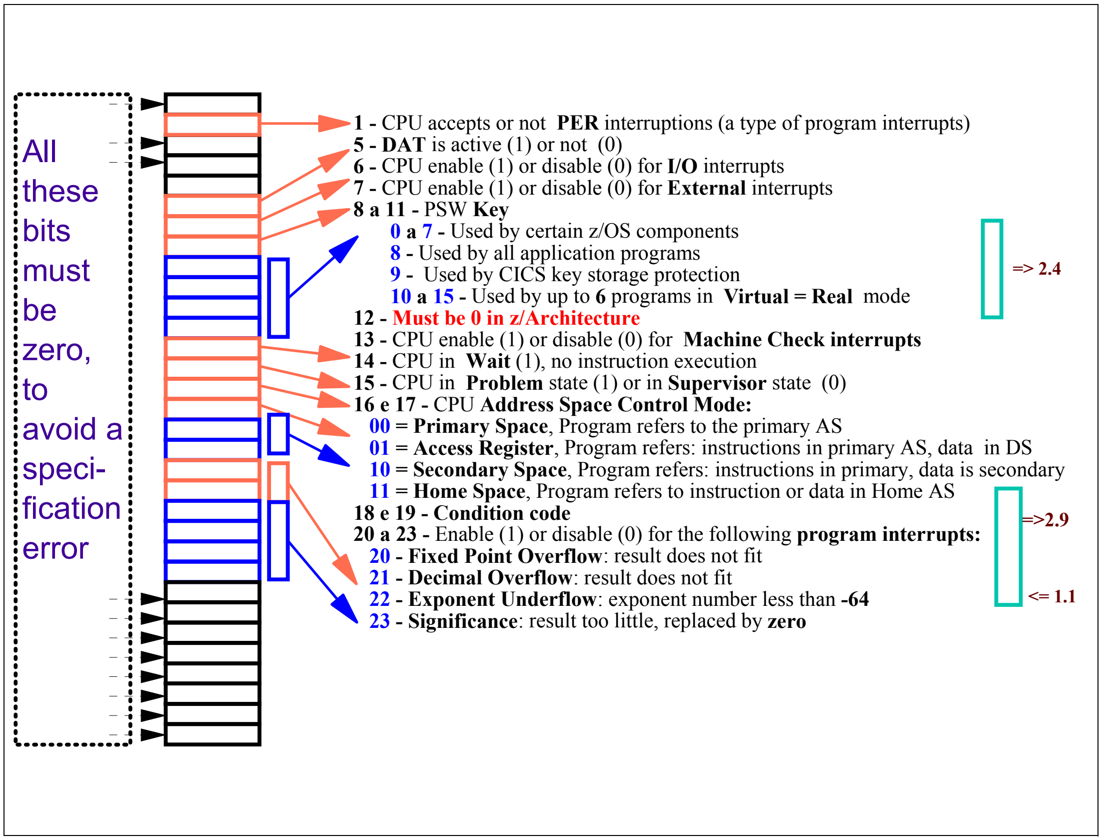


Figure 3-12 PSW from bit 0 to bit 31

Problem or supervisor bit mode (bit 15)

CP instructions can be classified as *privileged* and *non-privileged*. Note that, if misused, privileged instructions may damage system integrity and security. Privileged instructions should be executed only by z/OS programs. When the CP is in the supervisor state (bit 15 Off), it can execute any instruction. When the CP is in the problem state (bit 15 On), it can only execute non-privileged instructions. z/OS manages that, when its code is executing, bit 15 is Off; when an application program is executing, bit 15 is On.

PSW key (bits 8-11)

The PSW key is used by a hardware mechanism within the CP called *storage protection*. It guarantees that programs running processes do not alter or access areas in storage that belong to other processes.

Instruction address (bits 64 to 127)

Bits 64 to 127 point to the storage address of the next instruction to be executed by this CP. When an instruction is fetched from central storage, its length is automatically added to this field. Then, it will point to the next instruction address. However, there are instructions as a BRANCH that may replace the contents of this field, pointing to the branched instruction. The address contained in this PSW field may have 24, 31 or 64 bits, depending on the addressing mode attribute of the executing program. For compatibility reasons, old programs that still address small addresses are still allowed to execute. When in 24- or 31-bit addressing mode, the left-most bits of this field are filled with zeroes.

Central Processor interrupts

The CP has an interrupt capability, which permits it to switch rapidly to another program in response to exceptional conditions and external stimulus. When an interrupt occurs, the CP places the current PSW in an assigned storage location, called the old-PSW location, for the particular class of interrupt. The CP fetches a new PSW from a second assigned storage location. This new PSW determines the next program to be executed. When it has finished processing the interrupt, the program handling the interrupt may reload the old PSW, making it the current PSW again, so that the interrupted program can continue.

There are six classes of interrupt: external, I/O, processor check, program, restart, and supervisor call. Each class has a distinct pair of old-PSW and new-PSW locations permanently assigned in real storage.

Program-status-word format

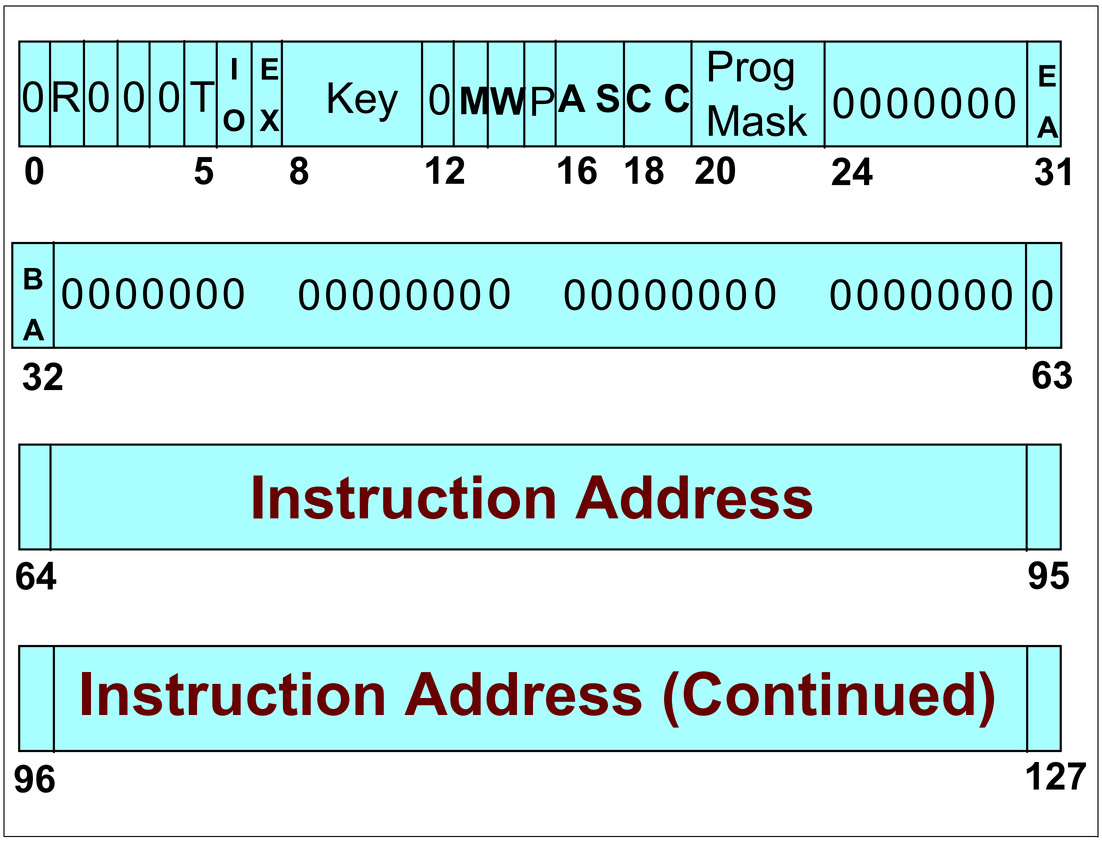


Figure 3-13 Program-status-word format

PER mask - R (bit 1)

Bit 1 controls whether the CP is enabled for interrupts associated with program-event recording (PER). When the bit is zero, no PER event can cause an interruption. When the bit is one, interruptions are permitted, subject to the PER-event-mask bits in control register 9.

DAT mode - T (bit 5)

Bit 5 controls whether implicit dynamic address translation of logical and instruction addresses used to access storage takes place. When the bit is zero, DAT is off, and logical and instruction addresses are treated as real addresses. When the bit is one, DAT is on, and the dynamic-address-translation mechanism is invoked.

I/O mask - IO (bit 6)

Bit 6 controls whether the CP is enabled for I/O interruptions. When the bit is zero, an I/O interruption cannot occur. When the bit is one, I/O interruptions are subject to the I/O-interruption subclass-mask bits in control register 6. When an I/O-interruption subclass-mask bit is zero, an I/O interruption for that I/O-interruption subclass cannot occur; when the I/O-interruption subclass-mask bit is one, an I/O interruption for that I/O-interruption subclass can occur.

External mask - EX (bit 7)

Bit 7 controls whether the CP is enabled for interruption by conditions included in the external class. When the bit is zero, an external interruption cannot occur. When the bit is one, an external interruption is subject to the corresponding external subclass-mask bits in control register 0; when the subclass-mask bit is zero, conditions associated with the subclass cannot cause an interruption; when the subclass-mask bit is one, an interruption in that subclass can occur.

PSW key (bits 8-11)

Bits 8-11 form the access key for storage references by the CP. If the reference is subject to key-controlled protection, the PSW key is matched with a storage key when information is stored or when information is fetched from a location that is protected against fetching. However, for one of the operands of each of MOVE TO PRIMARY, MOVE TO SECONDARY, MOVE WITH KEY, MOVE WITH SOURCE KEY, and MOVE WITH DESTINATION KEY, an access key specified as an operand is used instead of the PSW key.

Processor-check mask - M (bit 13)

Bit 13 controls whether the CP is enabled for interruption by processor-check conditions. When the bit is zero, a processor-check interruption cannot occur. When the bit is one, processor-check interruptions due to system damage and instruction-processing damage are permitted, but interruptions due to other processor-check-subclass conditions are subject to the subclass-mask bits in control register 14.

Wait state - W (bit 14)

When bit 14 is one, the CP is waiting; that is, no instructions are processed by the CP, but interruptions may take place. When bit 14 is zero, instruction fetching and execution occur in the normal manner. The wait indicator is on when the bit is one. When in wait state, the only way of getting out of such state is through an Interruption. Certain bits, when off in the current PSW, place the CP in a disabled state; the CP does not accept Interrupts. So when z/OS, for any error reason (software or hardware) decides to stop a CP, it sets the PSW to the Disable and Wait state, forcing an IPL in order to restore the CP back to the running state.

Problem state - P (bit 15)

When bit 15 is one, the CP is in the problem state. When bit 15 is zero, the CP is in the supervisor state. In the supervisor state, *all* instructions are valid. In the problem state, only those instructions are valid that provide meaningful information to the problem program and that cannot affect system integrity; such instructions are called unprivileged instructions.

The instructions that are never valid in the problem state are called privileged instructions. When a CP in the problem state attempts to execute a privileged instruction, a privileged-operation exception is recognized. Another group of instructions, called semi-privileged instructions, are executed by a CP in the problem state only if specific authority tests are met; otherwise, a privileged-operation exception or a special-operation exception is recognized.

Address-space control -AS (bits 16-17)

Bits 16 and 17, in conjunction with PSW bit 5, control the translation mode.

Condition code - CC (bits 18-19)

Bits 18 and 19 are the two bits of the condition code. The condition code is set to 0, 1, 2, or 3, depending on the result obtained in executing certain instructions. Most arithmetic and logical operations, as well as some other operations, set the condition code. The instruction BRANCH ON CONDITION can specify any selection of the condition-code values as a criterion for branching. The part of the CP that executes instructions is called the arithmetic logic unit (ALU). The ALU has internally four bits that are set by certain instructions. At the end of these instructions, this 4-bit configuration is mapped into bits 18 and 19 of the current PSW. Reason code - a code passed in the GPR 15 detailing how a task ended.

Program Mask (bits 20-23)

During the execution of an arithmetic instruction, the CP may find some unusual (or error) condition, such as overflows, lost of significance, or underflow. In such cases, the CP generates a program interrupt. When this interrupt is treated by z/OS, usually the current task is abnormally ended (ABEND). However, in certain situations programmers do not want an ABEND, so by using the instruction SET PROGRAM MASK (SPM), they can mask such interrupts by setting some of the program mask bits to OFF. Each bit is associated with one type of condition:

- Fixed point overflow (bit 20)
- Decimal overflow (bit 21)
- Exponent underflow (bit 22)
- Significance (bit 23)

The active program is informed about these events through the condition code posted by the instruction where the events described happened.

Extended addressing mode - EA, BA (bits 31-32)

The combination of bits 31 and 32 identify the addressing mode (24, 31 or 64) of the running program. Bit 31 controls the size of effective addresses and effective-address generation in conjunction with bit 32, the basic-addressing-mode bit. When bit 31 is zero, the addressing mode is controlled by bit 32. When bits 31 and 32 are both one, 64-bit addressing is specified.

Prefixed save area (PSA)

Figure 3-14 on page 61 depicts the layout of the PSA in z/Architecture. The PSA maps the storage that starts at location 0 for the related server. The function of the PSA is to map fixed hardware and software storage locations for the related server.

END.	Length	Function
0	8	Restart **NEW** PSW; **IPL PSW**
8	8	Restart **OLD** PSW; **IPL CCW1**
16	8	**CVT address; IPL CCW2**
24	8	External **OLD** PSW
32	8	Supervisor Call **OLD** PSW
40	8	Program Check **OLD** PSW
48	8	Machine Check **OLD** PSW
56	8	Input / Output **OLD** PSW
88	8	External **NEW** PSW
96	8	Supervisor Call **NEW** PSW
104	8	Program Check **NEW** PSW
112	8	Machine Check **NEW** PSW
120	8	Input / Output **NEW** PSW
128	4	**External** Interr.Parameter
132	4	**CPU** Address + **External** Code
136	4	**SVC** Interruption: **ILC** + Code
140	4	**Program** Interruption: **ILC** + Code
144	4	**Translation** Exception ID

END.	Length	Function
149	1	**Monitor class**
150	6	**PER** (**1** or **2**) Code + **PER** Address
156	4	**Monitor** Code
160	2	**Exception** Access + **PER** Access
184	4	**SID** (0001+Subchannel #) **=> 3.1**
188	4	**I/O** Interr.Param.subchannel (@ **UCB**)
216	8	St.Status / Mach.Check **CPU Timer** SA
224	8	St.Status / Mach.Check **Clock Comp**.SA
232	8	**Machine Check** Interruption Code
244	4	External **Damage** Code
248	4	**Failing** Storage Address
256	16	St.Status **PSW** SA; Fixed **Logout** area
272	16	Reserved
288	64	St.Status / Mach.Check **Access** reg.SA
352	32	St.Status / Mach.Check **Flt.Pt**. reg.SA
384	64	St.Status / Mach.Check **General** reg.SA
448	64	St.Status / Mach.Check **Control** reg.AS
		....

Figure 3-14 Prefixed save area (PSA)

3.2.5 Interrupt events

An interrupt occurs when the CP detects one of six events. During interrupt processing, the CP does the following:

- Stores (saves) the current PSW in a specific central storage location named old PSW
- Fetches, from a specific central storage location named new PSW, an image of PSW and loads it in the current PSW
- Stores information identifying the cause of the interrupt in a specific central storage location called interrupt code

Old and new PSWs are just copies of the PSW current contents. Processing resumes as specified by the new PSW instruction address and status. The old PSW stored on an interrupt normally contains the status and the address of the instruction that would have been executed next had the interrupt not occurred, thus permitting later the resumption of the interrupted program (and task).

Six groups of events cause interrupts: Supervisor Call (SVC), Input/Output, Program check, External, processor check, and Restart. For each type of interrupt there are, in central storage, a trio of locations for old PSW, new PSW, and interrupt codes. These locations are kept in an 8 KB area at the very beginning of central storage called the Prefix Storage Area (PSA). Depending on the type of the interrupt, a CP may be temporarily disabled by z/OS because of integrity reasons, as described by bits 6 and 7 in the PSW and also by bit settings in the CRs. In this case, the interrupt is not lost but stacked in the original hardware element, or handled by other CPs in the server.

Reasons for interrupts

When the Central Processor finishes the execution of one instruction, it executes the next sequential instruction (the one located in an address after the one just executed). The instruction address field in the current PSW is updated in order to execute the next instruction. If the logic (set of logically connected instructions) of the program allows it, the next instruction can branch to another instruction through the BRANCH ON CONDITION instruction.

In a sense, an interrupt is a sort of branching—but there is a logical difference between a BRANCH instruction issued in a program and an interrupt. A BRANCH is simply a twist in the logic of the program. In an interrupt, however, one of the six interruption conditions occurred which needs to be brought to the attention of z/OS immediately.

Interrupt processing

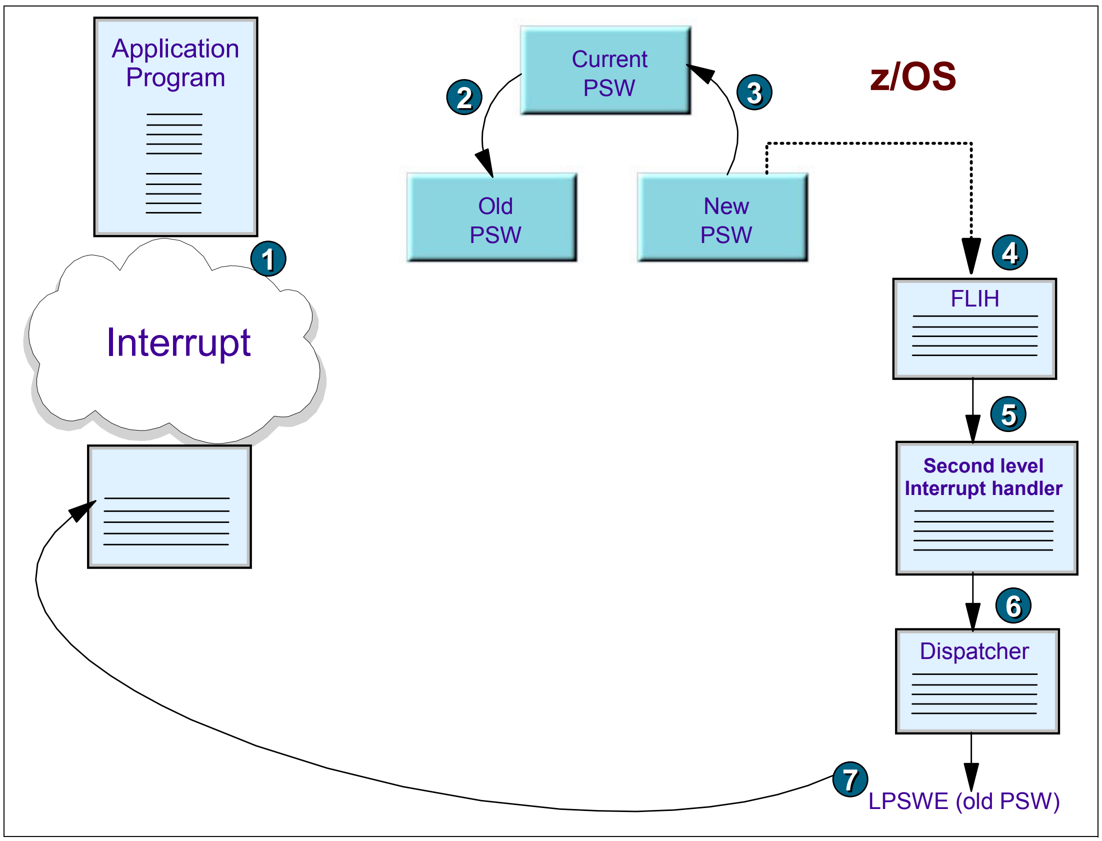


Figure 3-15 Interrupt processing

Step 1

The application program is interrupted by one of the six classes of interrupts.

Step 2 and Step 3

The CP was following the sequence of the instructions pointed by the instruction address in the current PSW and suddenly, after the interrupt, it now addresses (and executes) the instruction pointed to by the copy of PSW located in the new PSW, which is now loaded in the current PSW. Each type of interrupt has two related PSWs, called old and new, in permanently assigned real storage locations. Each type of interrupt involves storing information that identifies the cause of the interrupt, storing the current PSW at the old-PSW location, and fetching the PSW at the new-PSW location, which becomes the current PSW.

Note that all the events generating interrupts have something in common: they cannot be processed by an application program. Instead, they need z/OS services (see Step 4). So why is it that a simple branch to z/OS code does not solve the problem? The reason is because a branch does not change the bits of the current PSW.

Step 4

Control is passed to the first level interrupt handler (FLIH). z/OS will use privileged instructions to respond to the event; the new PSW (prepared by z/OS itself) has bit 15 turned off. z/OS needs to access a storage location to respond to the event, and the new PSW (prepared by z/OS) has the PSW key set for that storage location.

Step 5

Control is passed, for each type of interrupt, to a second level interrupt handler for further processing of the interrupt.

Step 6

The MVS dispatcher that dispatches all waiting tasks can dispatch the interrupted program if an available CP is ready for new work.

Step 7

The MVS dispatcher does this by using the old PSW, which has been saved, and which contains CP status information necessary for resumption of the interrupted program. At the conclusion of the program invoked by the interruption, the instruction LOAD PSW EXTENDED may be used to restore the current PSW to the value of the old PSW and return control to the interrupted program.

Program check interrupt type

This interrupt is generated by the CP when something is wrong during the execution of an instruction. Usually the z/OS reaction to a program interrupt is to ABEND the task that was executing the program in error. However, getting something wrong during instruction execution does not necessarily indicate an error. For example, a page fault program interrupt (interrupt code 11) indicates that the virtual storage address does not correspond to a real address in central storage, but the task is not ABENDed.

Supervisor call interrupt type

This interrupt is triggered in the CP by the execution of the SUPERVISOR CALL (SVC) instruction. When executed by the CP, the SVC instruction causes an SVC interrupt; that is, the current PSW is stored in the PSA at the SVC old PSW, a new PSW from the PSA is loaded in the current PSW, and the second byte of the instruction is stored in the SVC interrupt code in PSA memory. The reason for such interrupts is part of the architecture, where an application program running in problem mode (bit 15 of the current PSW on) may pass control to z/OS asking for a service. After the request is processed by the z/OS SVC routine, the interrupted program can regain control by restoring of its registers and the LPSWE instruction issued against the copy of the SVC old PSW.

Input/output interrupt type

An I/O operation is requested by a task to the input/output supervisor (IOS) through an SVC 0 instruction. After the SVC interrupt processing, the SVC FLIH passes control to IOS. In z/Architecture, the I/O operation is not handled by the CP executing z/OS code. There are less expensive and more specialized servers to do the job: the channels. When IOS issues the privileged START SUBCHANNEL (SSCH) instruction, the CP delegates to a channel the execution of the I/O operation. Then, the I/O operation is a dialogue between the channel and an I/O control unit, in order to move data between central storage and the I/O device controlled by such a controller. After the execution of the SSCH instruction, IOS returns control to the task issuer of the SVC 0. This task places itself in wait, till the end of the I/O operation. Now, how does IOS and the CP become aware that the I/O operation handled by the channel is finished? This is handled through an I/O interrupt triggered by the channel. The I/O new PSW points to the IOS code in z/OS (I/O FLIH) and the interrupt codes tell IOS which device has an I/O operation that has completed. The final status of the I/O operation is kept in a control block called the Interrupt Request Block (IRB). The I/O old PSW has the current PSW at the moment of the I/O interrupt, so it can be used to resume the processing of the interrupted task.

External interrupt type

This type of interrupt has eight different causes, usually not connected with what the active program is doing, as follows:

- 0040 Interrupt key - An interrupt request for the interrupt key is generated when the operator activates that key in the Hardware Management Console.
- 1004 Clock comparator - The contents of the TOD Clock became equal to the Clock Comparator.
- 1005 CPU timer - The contents of the CPU Timer became negative.
- 1200 Malfunction alert - Another CPU in the Multiprocessing tightly coupled complex is in check stop state due to an hardware error. The address of the CPU that generated the condition is stored at PSA locations 132-133.
- 1201 Emergency signal - Generated by the SIGNAL PROCESSOR instruction when z/OS, running in a CPU with a hardware malfunction, decided to stop (Wait Disable) that CPU. The address of the CPU sending the signal is provided with the interrupt code when the interrupt occurs.
- 1202 External call - Generated by the SIGNAL PROCESSOR instruction when a program wants to communicate synchronously or asynchronously with another program running in another CPU. The address of the CPU sending the signal is provided with the interrupt code when the interrupt occurs.
- 1406 ETR - An interrupt request for the External Timer Reference (ETR) is generated when a port availability change occurs at any port in the current server-port group, or when an ETR alert occurs.
- 2401 Service signal - An interrupt request for a service signal is generated upon the completion of certain configuration control and maintenance functions. A 32-bit parameter is provided with the interrupt to assist the program in determining the operation for which the interrupt is reported.

Processor check interrupt type

This type of interrupt is a part of the processor check-handling mechanism. This mechanism provides extensive equipment malfunction detection to ensure the integrity of system operation and to permit automatic recovery from some malfunctions. This detection design is mainly based on the redundancy of components. For example, within each CP there are execution units, two of them executing the same instruction and a third comparing the results. Equipment malfunctions and certain external disturbances are reported by means of a processor check interrupt to assist z/OS in program damage assessment and recovery. The interrupt supplies z/OS with information about the extent of the damage and the location and nature of the cause. Four hardware mechanisms may be used to provide recovery from server-detected malfunctions: error checking and correction, CP retry, channel subsystem recovery, and unit deletion.

There are two types of processor-check-interrupt conditions: *exigent* conditions and *repressible* conditions:

- Exigent processor-check-interrupt conditions are those in which damage has or would have occurred such that execution of the current instruction or interrupt sequence cannot safely continue.
- Repressible processor-check-interrupt conditions are those in which the results of the instruction processing sequence have not been affected.

Restart interrupt type

The restart interrupt provides a means for the operator (by using the restart key in HMC) or a program running on another CP (through a SIGNAL PROCESSOR instruction) to invoke the execution of a specified z/OS component program. The CP cannot be disabled for this interrupt. In z/OS, the specific component does an evaluation of the system status, reporting hangs, locks, and unusual states in certain tasks. It gives the operator a chance to cancel the offending task. It maybe the last chance in order to avoid an IPL.

Storage protection logic

z/OS may alter storage key bits by issuing the SSKE (Set Storage Key Extended) instruction, and may inspect them by the ISKE (Insert Storage Key Extended) and IVSK (Insert Virtual Storage Key) instructions. The reference bit is inspected and switched off after inspection by the RRBE (Reset Reference Bit Extended) instruction. On top of that, the CPU storage hardware switches on the reference bit when the frame is accessed by any CP or any channel, and also switches on the change bit when the frame contents are changed by those components.

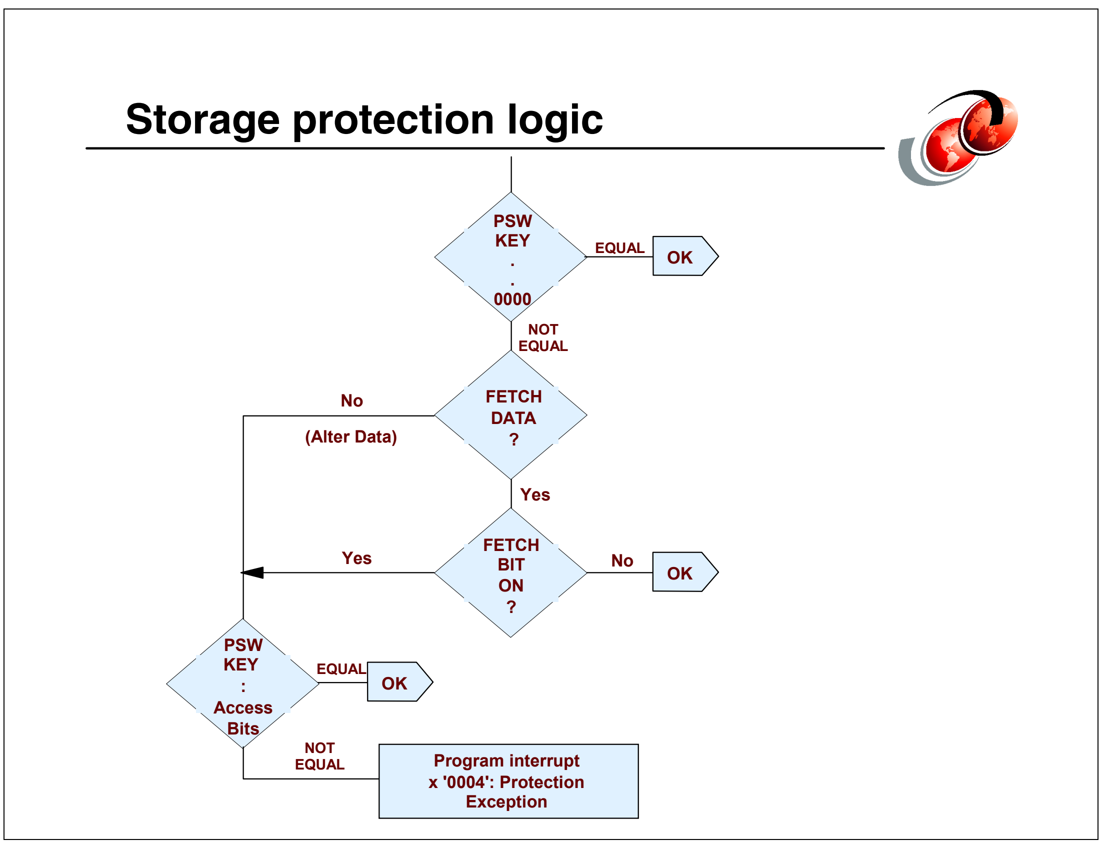


Figure 3-16 Storage protection logic

z/OS exploits storage protection by managing frame storage key values and running the program PSW key field in the current PSW. Several z/OS routines run with PSW key zero, and others run with PSW key one. Application code has PSW key eight. The following conclusions can be reached from the logic:

- If a running program has PSW key equal to 0000, it may access any frame in memory.
- If the fetch bit is off in a frame, any program can read the contents of that frame.
- To read the contents of a frame where the fetch bit is on, the PSW key of the running program must match the access control 4 bits in the storage key of the frame.
- To alter (write) the contents of a frame, the PSW key of the running program must match the access control 4 bit in the storage key of the frame.

4

Virtualization

In this chapter the reader is given a high level description of the virtualization technologies that IBM provides in System z, namely the PR/SM ('Processor Resource/Systems Manager™") facility and the IBM z/VM hypervisor program product. The security- relevant issues of performing virtualization are addressed and how they are solved in the IBM implementations of these technologies is discussed. Finally we address the Common Criteria certification of the System z PR/SM facility.

4.1 System z virtualization security and the IBM Security Blueprint

We are briefly describing in this section what the generic threats for a virtualized environment are and how the PR/SM implementation meets the security services and infrastructure as specified in the IBM Security Strategy Blueprint.

4.1.1 The threats

At a global level, the threat encountered when exploiting virtualized environments do not differ from the threats a physical computing environment is exposed to, they are:

- The capability for users to gain unauthorized access to data
  Users may gain access to data belonging to another instance of a virtualized environment, for which they do not have clearance, specific authorization, or a need-to-know. This may be achieved either directly, for example:
  - by reading storage allocated to another instance of environment
  - or by failure to clear a resource that might be re-allocated among environments)

  Or indirectly (for example, through a covert channel.

  Unauthorized access to audit data may lead to a false record of System Administrator actions.
- The capability for users to gain unauthorized access to system resources (i.e. channel path, control unit, I/O device, physical or virtualized processor): such actions being contrary to the security or resource policy of an Organization.

By essence virtualized environment do not provide a physical separation that could contribute to lower the exposure to these threats, therefore their inherent security relies on the proper design and implementation of the virtualization mechanisms, both from the operation and administration standpoint. As for physical computing environment, proper attention should also be given to physical security and the effectiveness of operating procedures.

We are addressing in this chapter general principles of virtualization mechanisms and the security related design points. We then describe the implementation of System z built-in virtualization exploiting the PR/SM (Processor Resource / Systems Manager) facility. Specific Security functions are embedded in the PR/SM implementation that we explained in this chapter.

4.1.2 Mapping to the Blueprint Security Services and Infrastructure

As it is explained in this chapter and Chapter 5, the implementations of virtualization mechanisms in System z meet the characteristics of a trusted computing environment, and as such provide the high level of host integrity and security that installations require when it comes to implement the security services their security policy relies on.

We are addressing in this section the security functions related to the implementation of PR/SM itself, and show that they map to the IBM Security Blueprint Security Services and Infrastructure.

Security Information and Event Management Infrastructure

Events classified as security related, for instance the modification of a virtualized environment, are duly recorded in a Security Log hold in the System z Support Element (SE).

These log entries, which contain proper information about the nature of the event, the identity of the responsible entity and a reliable timestamp can be searched, but not modified, by authorized personnel using SE provided tools.

Identity, Access and Entitlement Infrastructure

A user provisioning scheme is implemented in the System z SE, so that users operating the system can be authenticated before accessing the system's management functions. The users' privileges are granted on the basis of the role the user is in.
The same concept applies, with a specific implementation, for the internal operations of System z PR/SM, where virtual environments are given a unique identity used to control access to system's resources.

Security Policy Infrastructure

PR/SM provides the virtualized environments with programmatic system's management functions that security administrators can enabled or disabled at the system or virtualized environment level. These functions although deemed useful in many installations might be incompatible with other installations' security policies, in that case the virtualized environments will operate with these functions disabled.

Cryptographic, Key and Certificate Infrastructure

The PR/SM facility, although it provides access to cryptographic devices to the virtualized environment, does not itself exploit cryptography.

Network Security

The PR/SM facility provides a fully hardware and firmware simulated Ethernet LAN, known as "HiperSockets™" that can connect virtualized environments co-existing in the same physical system. The network remains fully internal to the PR/SM firmware and cannot be attacked by physical means.

Storage Security

Storage security is required to protect the physical repository of data belonging to the system and the virtualized environments configuration. This is provided by the System z SE which keeps these data access controlled on its hard disk with mirroring to the backup SE hard disk.

Host and Endpoint Security

PR/SM is implemented in such a way that it exploits the native physical components of the system, themselves in charge of ensuring proper workloads isolation. That is there are no specific data paths in the system that are dedicated to the operations of virtualized environments. These environments therefore benefit of the proven integrity and security of the native implementation.

Application Security

As a primary design point PR/SM is not involved in Application Security. The virtualized environments provide the same integrity and security oriented functions as provided by the physical system, however this is up to the applications running in these environments to properly use these functions.

Service Management and Process Automation

The same services as globally implemented in the System z hardware apply to PR/SM, that is the capability of automatic reporting of problems to a remote support center, with proper contextual information, and the automated initialization of the system and its environment using predefined configuration data.

Physical Security

Globally speaking System z in itself does not contribute in itself to or implement, with the one exception which follows, Physical Security. Note however that physical security is built in the Crypto Express 2 (CEX2) coprocessors: sensors are in place in the devices to detect various kinds of physical attempts to read the secrets they contain. A detected tampering event provides an automatic erasure of these secrets (cryptographic coprocessors “zeroize” function).

IT Security Services and Mechanisms

Again here for System z the remote support infrastructure provides these services.

4.2 Introduction to Virtualization

This section introduces the concept of virtualization.

4.2.1 What it is

Virtualization in the IT world refers to a set of techniques of which purpose is to deliver an abstract view of computer resources, sometimes called an ‘image’. In a virtualized computing environment users’ programs and operating systems are specified to use resources, like memory, processors or I/O devices, which do not correspond one-to-one to actual physical resources. These virtual resources are backed by real physical resources of which use is optimized by sharing a single physical resource between several virtualized environments, or by physical resources fundamentally different from the virtual one that the programs are exploiting. In this latter case virtualization heavily involves simulation, as it is the case, for instance, to simulate now-obsoleted devices that were in common use when the executing program was written.

Today techniques are available to apply Virtualization to any physical entity being part of an I/T logical infrastructure, such as disk storages or communication links, with a scale ranging from virtualizing a set of installations as a “cloud” down to the creation of multiple virtualized computing environments in a PC.

To remain consistent with the contents of this book we will focus on virtualization techniques and mechanisms, as they are available as of the writing of this book, in the IBM System z mainframe. A schematic example of virtualized environments creation in System z is given in Figure 4-1. In this figure the virtualized environments execute IBM (z/OS, z/VM) and Linux for system z operating systems with, each of them, managing the execution of its own customers’ applications. We are to discuss in some details in this chapter as to how virtualization is achieved in System z, the result being that, as shown in Figure 4-1, a single physical System z can appear to users as multiple installations, each one with its operating system of choice.

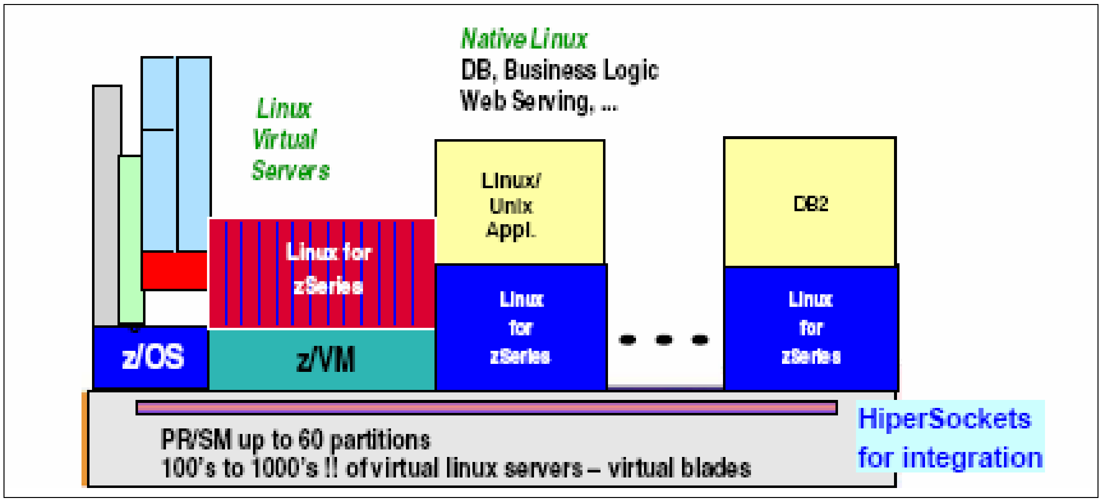


Figure 4-1 Multiple virtualized environments in a System z.

4.2.2 Why Virtualization?

Beside the quite appealing capability of allowing, to some extent, to easily adapt the environment to the programs (and not the opposite, as it is most of the time the case), there are strong business drivers behind the adoption of Virtualization. Although the magnitude of the benefits of Virtualization may vary, depending on the workload context, the specific virtualization technologies selected, or the existing IT infrastructure, users can expect to achieve gains in one or many of the following areas:

- Higher physical resources utilization
  Virtualization enables the dynamic sharing of physical resources and resource pools, resulting in higher resource utilization, especially for variable workloads where the average needs are much less than an entire dedicated resource.
- Lower management costs
  Virtualization can improve staff productivity by reducing the number of physical resources that must be managed; hiding some of the resource complexity; simplifying common management tasks through automation, better information and centralization; and enabling workload management automation.
- Usage flexibility
  Virtualization enables resources to be deployed and reconfigured dynamically to meet changing business needs.
- Higher availability
  Virtualization enables physical resources to be removed, upgraded, or changed without affecting users.
- Increased scalability
  Resource partitioning and aggregation enable a virtual resource, depending on the product, to be much smaller or much larger than an individual physical resource, meaning that the user can make scale adjustments without changes to the physical resource configuration.
- Interoperability and investment protection
  Virtual resources can provide compatibility with interfaces and protocols that are unavailable in the underlying physical resources. This is increasingly important for supporting existing systems and ensuring backward compatibility.

- Improved provisioning
  Virtualization can enable resource allocation to a finer degree of granularity than individual physical units.
- Consolidation
  Virtualization enables multiple applications and operating systems to be supported in one physical system, as well as consolidating servers into virtual machines on either a scale-up or scale-out architecture, It also enables systems to treat computing resources as a uniform pool that can be allocated to virtual machines in a controlled manner.

4.2.3 Issues with Virtualization - How they are addressed in System z

The virtualization mechanisms can be thought as a layer of translation for the interactions between the executing programs, which “see” virtual resources and the real resources which are actually exploited by these programs. This translation layer remains transparent for the system’s users, except for the set of users precisely in charge of administering the Virtualization mechanisms.

Hardware of software implementation?

The Virtualization mechanisms can be implemented as hardware and firmware functions or as purely software functions. The choice generally is a trade-off between performance and adaptability. It is being expected that a hardware implementation of the Virtualization layer will run more efficiently, but would on the other hand be less adaptable than a software implementation.

Both implementations are available for the IBM System z, in a non-exclusive manner in that a software driven Virtualization (the z/VM “hypervisor”) can be operated on top of the System built-in hardware Virtualization (PR/SM, “Processor Resource/Systems Manager”) facility.

It is probably worthwhile at this time to emphasize the wealth of experience that IBM has accumulated over years on implementing Virtualization in the mainframe as z/VM was initially released as VM/370 (‘Virtual Machine’) in 1972, and PR/SM made available in the IBM mainframes in 1988.

Performance

Obviously there is a cost to operate the Virtualization layer that ultimately translates into additional processor time consumption and additional management tasks incurred by the exploiting organization. It is then required that the Virtualization benefits offset by far this extra cost.

To achieve optimum Virtualization performance System z uses specifically dedicated hardware mechanisms at the core of its Virtualization implementation that are exploited both by the built-in hardware Virtualization (PR/SM) and the software Virtualization (z/VM). An explanation of the “interpretive-execution” mode of System z is given in “The System z interpretive-execution facility” on page 79.

Virtualized environment security features

Virtualization brings its own set of security issues related to the operations of the Virtualization mechanisms or to the their administration.To name a few:

- The administrative access to virtualization mechanisms configuration and controls should be secure.
- Virtual environments accessing physical resources should be duly authenticated and authorized to do so.
- Virtual environments should be strictly isolated from interferences by other environments.

- ...

Although this looks similar to the Security concerns found when designing an operating system, one must realize that they pertain to the integrity of the virtualized environment. “Integrity” in this case focusing on the capability of the virtualized environment to strictly react as a real one, and to be fully isolated from interferences or penetrations by the other coexisting virtualized environments.

The IBM approach for mainframe virtualization security is to provide a virtualized environment with as much integrity as if it were a real system standing by itself and leave the guest software manage its own security. IBM gets this capability certified by independent laboratories for the PR/SM facility of each new mainframe model and for new releases of z/VM.

The z/VM or PR/SM certification is performed against the Common Criteria (ISO15408) standards. More information on this certification can be found in 4.9, “More on PR/SM security - The certification proof points” on page 102.

4.3 Overview

Virtualization refers to a set of techniques of which purpose is to deliver an abstract view of computer resources, sometimes called an “image” of a computing environment. The virtualized computing environment is provided with virtual resources, behaving, from the users’ standpoint as if they were real ones. Virtual resources can be fully simulated by the virtualization mechanisms or be actually backed by their real counterparts. Programs can be executed in virtualized environments as proper mechanisms are in place to proceed with an initial load of a program and start its execution. Typically the program which is loaded into the virtualized environment is an operating system which, in turn, loads and manages the execution of users’ applications, as if they were running on a standalone physical system.

Most Virtualization implementations allow the creation in a single physical system of multiple virtualized environments that, by sharing physical resources through their virtualized “images”, lead to an optimized utilization of these physical resources. Optimizing the use of physical resources appears today one of the prevalent benefits users find in exploiting Virtualization in a commercial environment. Other benefits of Virtualization are discussed in the next paragraph.

Virtualization can be implemented as purely software functions dedicated to providing the environment image to the users, in that case a special operating system, usually called an “hypervisor”, creates and manages the virtualized environments. The IBM z/VM program product is the System z software hypervisor that creates and manages “virtual machines” into which programs can be loaded and executed.

Virtualization can also be implemented as a set of hardware and firmware mechanisms, as it is the case with the PR/SM (Processor Resource / Systems Manager) facility of System z which is used to create and manage “logical partitions”. PR/SM is described at 4.5, “System z Processor Resource / Systems Manager (PR/SM)” on page 82.
Note that users can run z/VM inside a PR/SM logical partition thus achieving nesting of virtualized environments.

It is obvious that whatever the Virtualization mechanism is it brings some overhead to programs execution as it also consumes physical system resources, however there is today a general recognition of Virtualization benefits offsetting by far the inherent cost of Virtualization.

Virtualization brings also environment integrity and Security concerns of its own that we will discuss in this chapter.

4.3.1 Goals and benefits of virtualization

Virtualization has been available for more than three decades in the IBM mainframes (the VM/370 hypervisor was generally available in 1972, preceded in the late sixties by early implementations of hypervisor concepts such as the IBM System/360 CP-67 operating system). Virtualization has been since constantly in high demand from users because of the strong business drivers that surfaced along the years that can be efficiently satisfied by Virtualizing the computing environment. Although the magnitude of the benefits of Virtualization may vary, depending on the workload context, the specific virtualization technologies used, or the existing IT infrastructure, users can expect to achieve gains in one or many of the following areas:

- Higher physical resources utilization
  Virtualization enables the dynamic sharing of physical resources and resource pools between several virtualized environments, resulting in higher resource utilization, especially for variable workloads where the average needs are much less than an entire dedicated resource.
- Lower management costs
  Virtualization can improve staff productivity by reducing the number of physical resources that must be managed; hiding some of the resource complexity; simplifying common management tasks through automation, better information and centralization; and enabling workload management automation. Note that Virtualization has however its own management costs, that are usually offset by this productivity improvement.
- Usage flexibility
  Virtualization enables computing resources to be deployed and reconfigured dynamically to meet changing business needs.
- Higher availability
  Virtualization enables physical resources to be removed, upgraded, or changed without affecting users operations in the virtualized environment.
- Increased scalability
  Resource partitioning and aggregation enable a virtual resource, depending on the product, to be much smaller or much larger than an individual physical resource, meaning that the user can make scale adjustments without changes to the physical resource configuration.
- Interoperability and investment protection
  Virtual resources can provide compatibility with interfaces and protocols that are unavailable in the underlying physical resources. This is increasingly important for supporting existing systems and however ensuring backward compatibility.
- Improved provisioning
  Virtualization can enable resource allocation to a finer degree of granularity than individual physical units would allow.
- Consolidation
  As virtualization enables multiple applications and operating systems to be supported in one physical system, it can be used to consolidate servers into virtual machines on either a scale-up or scale-out architecture. It also enables systems to treat computing resources as a uniform pool that can be allocated to virtual machines in a controlled manner.

4.3.2 Some theory about Virtualization

We are briefly looking in this paragraph at some concepts underlying the virtualization implementation, we also address the generic integrity and Security concerns that are specific to the operations of virtualized environments.

The virtualized environment - Real and virtual resources

The virtualized environment gives the programs it contains access to all the resources that are needed in a computing environment, however through a virtual "image" of these resources provided by the system's virtualization functional layer. These virtual resources include:

- memory- usually portions of the system's real memory are allocated to the virtualized environments, each one of them with its own set of "virtualized real addresses".
- I/O devices and their access paths: some virtual devices may be fully simulated whereas others can be backed by similar physical devices.
- Networking facilities - They might not need to be backed by real networks if they are used solely to interconnect virtual entities in the same physical system. If this is the case these networking facilities are rather fully simulated in the virtualization layer.
- System controls - They apply to virtualized environments as well which can then be controlled as if they were real machines. One example of such controls is a System Reset or an IPL function that load programs into the virtualized environment and begins their execution.
- Processing units and coprocessors (if any) - They are obviously backed at some degree by real resources, although their degree of dedication to executing the virtualized environments' program varies with the virtualization mechanisms.

Figure 4-2 is a schematic view of virtualized environments created in a real system. Note the terminology here: the system creating and managing the virtualized environment is the "**host**" system whereas the "**guest**" software exploit their virtual environment. The term "**native**" is also commonly used to design any property relating to a real entity as opposed to a virtual one.

You can also note that the figure shows second level virtual environments at the right: a software guest hypervisor is executing in its virtualized environment and it itself creates and manages its own virtualized environments.

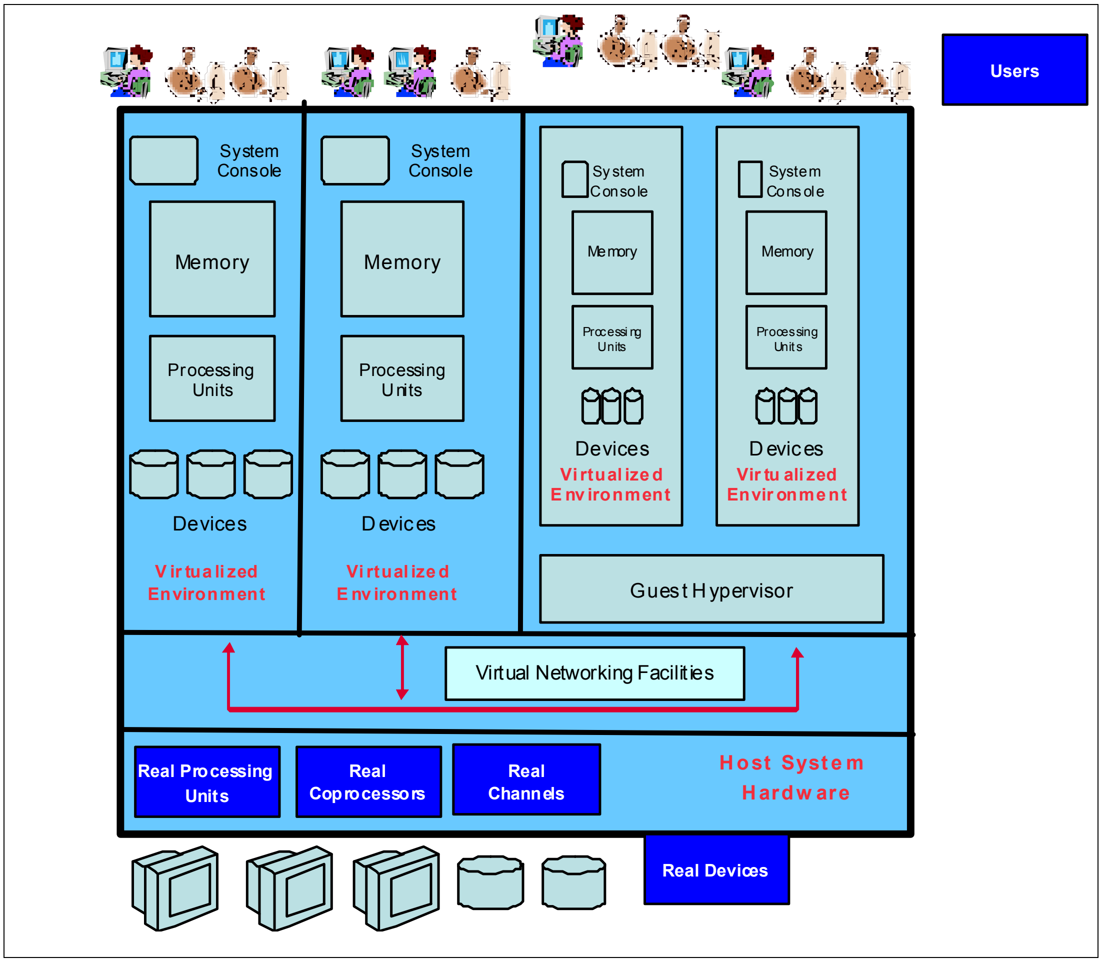


Figure 4-2 Virtualized environments

Virtualization and simulation

Virtualization could be fully achieved by simulation only, as it is the case when the aim of virtualization is to provide an environment architecturally different from the host system. However virtualization in a business context performs more efficiently when the virtualized environments are conforming to the native architecture of the host system. In this latter case the host hardware can be directly exploited, under control of the virtualization mechanisms, by the guest programs without going through a simulation layer. Note however that even there simulation is usually required when a host program is, for instance, to modify the environment, as this modification is to affect the virtualized environment and not the real system.

The System z implementation of Virtualization, either with z/VM or PR/SM, is a good example of this latter approach as it provides zArchitecture compliant virtualized environments. Virtual CPUs are backed by real processing units (PUs) in the machine, so that the instructions flows issued from the guest programs in the virtualized environments are dispatched for execution on real PUs. This avoids as much as possible to proceed with simulation and ensures maximum performance for guest programs execution.

Virtualization and time slicing

Time slicing is the base mechanism used with virtualization when it comes to share physical resources between several virtualized environments. As an example real processing unit execution time can be allocated on a time slice basis so that different virtualized environment can share the same real CPU.

Time slicing in the virtualization context has also issues of its own. For instance:

- The time slicing mechanism should accommodate for asynchronous events that may, for instance, require to immediately swap the current virtual environment execution flow with an instruction flow put on hold waiting for this event.
- The dispatching mechanism, for Security reasons, should also ensure that residual data are cleared before changing the allocation of a shared real resource.
- The dispatching mechanism must also guarantee the integrity of the state information which are saved and restored when execution contexts are switched active or inactive.

In many implementation of Virtualization the time slicing and dispatching algorithms have been refined up to the point that users could assign relative performance goals to the virtualized environments, as it is the case with System z PR/SM or z/VM.

Note also that where there is a strong requirements to achieve very close to native performance, an implementation of Virtualization can offer the capability of dedicating real resources to a specific virtualized environment, that is resources which are not to be shared with other environments. This obviously eliminates most of the time slicing and dispatching overhead. This capability is also offered by System z PR/SM and z/VM.

Configuration of the virtual environment

Configuring a virtual environment is the operation where a system administrator specifies which logical resources will be available to the programs executing in the virtual environment. That is the memory size, the amount of processing units, I/O devices and their access paths, etc. This information is usually kept in a permanent storage area that is used by the Virtualization mechanism when it comes to create the environment or to statically or dynamically modify it.

The configuration of a virtualized environment has also some security connotation:

- Only authorized persons can assign virtual resources to a virtualized environment in agreement with the installation policy addressing workload allocation and assignment of performance goals. It is also of paramount importance that this allocation of resources does not lead to unexpected sharing of real resources or unexpected communications paths between different virtualized environments.
- The Virtualization mechanism should strictly enforce allocating virtual resources to the proper virtualized environment, as per the system administrator’s specifications.

Dynamic reconfiguration of virtualized environments

Being able to reconfigure the environment dynamically implies that the ongoing operations are not disrupted during the change operation and proper synchronization is achieved between ongoing tasks and the activation of the new configuration.

States of a virtualized environment

It is expected that a virtualized environment is not going “live” as soon as defined but that it requires an authorized intervention to be put in the operating state. The same is true as well when it comes to shut down an operating environment.

The state of an environment is important to consider as it also relates to the allocation and use of resources and how the integrity and security mechanisms apply.

Precise definitions of all the possible states that a virtualized environment can be in vary with each implementation, however the following generic states are always, although under different names, provided by each implementation:

- de-activated state - In this state a virtualized environment has been defined but is not created yet, or has been shut down. in the host system. In this state the virtualized environment is not allocated any of its specified virtual resources.
- activated in initial state - the virtual environment is created, that is resources are allocated but operations did not begin yet in the environment. Typically that would be the state of a virtualized environment waiting for the loading of the initial program.
- activated in operating state - the virtual environment is created and the resources are being used by the workload it executes.

Security in a virtualized environment

When addressing this topic it is needed to clearly understand to which entities Security applies in this discussion. In order not to digress from the very subject of this book we will only refer to the System z implementation of Virtualization in this paragraph.

The System z implementation approach, either with PR/SM or z/VM assumes that guests programs are fully in charge of ensuring their own integrity and security, that is it is not expected from the Virtualization mechanisms to interfere or co operate in any way with the guest programs in this area. In other words a poorly-designed program will be running as bad in a System z virtualized environment that it will do in a native environment.

This leaves for discussion the Security of the environment itself.

Integrity of the virtualized environment

The integrity of the environment requires that:

- The specified virtual resources do behave exactly and only as per their architectural definitions and specifications.
- Strict isolation applies to Virtualized environments - This pertains to virtual environments co-existing in the same physical host system. Although the concept of environment isolation is rather easy to grasp, it needs usually to be translated into formal assertions, such as: “an operation executed in a virtualized environment provides effects that can be described only in terms of the virtual resources that are allocated to this environment. These effects shall not be perceivable by other virtualized environments unless these virtual resources are explicitly intended to provide inter-environment communications”.

  This leads to finer considerations such as:

  – It must be impossible for a virtualized environment to gain resources that are not intended to be allocated to this environment.

  – It must be impossible for a user in virtualized environment A to receive or send information from or to virtual environment B unless both environments are properly configured for authorized intercommunications. This item also indirectly addresses the clearing of information in shared real resources before their dynamic re-allocations.

Security of the virtualized environment

We are calling here Security of the virtualized environment the services provided by the physical host system to prevent unauthorized users to affect the definitions or operations of the physical environments and to keep a trace of any event that can potentially produce these effects.

The Security services should therefore provide for:

- Access control to the virtualized environments definitions data
- Access control to state control of the virtualized environment
- Access controlled auditing data

It is also necessary that the implementation of the virtualization mechanisms does not allow operating conditions that might lead to put a virtualized environment in a denial-of-service condition. As it would be the case, for instance, if shared physical resources in a multiple virtual environments configuration were exclusively held by a single virtualized environment.

4.4 Introduction to Virtualization in System z - PR/SM and z/VM

As mentioned in “Overview” on page 73 two virtualization implementations are provided for the IBM System z:

- the z/VM software hypervisor - z/VM is described and explained in Chapter 5. As mentioned, the first release of the hypervisor was available as the VM/370 operating system in 1972.
- The PR/SM facility which is a hardware and firmware implemented Virtualization which comes as a standard feature in IBM System z processors. PR/SM is described and explained in this chapter. The PR/SM feature was initially released in the IBM 3090 processors in 1988.

Although the implementations differ, both offer similar functions in the way the virtualized environments are managed, with however specific features for each one of these products.

The terminology they use also slightly differs. As an example PR/SM refers to **physical** and **logical** resources, whereas z/VM deals with **real** and **virtual** resources. Also PR/SM creates and manages **logical partitions**, whereas z/VM creates and manages **guest virtual machines** (VM).

Both implementations allow nesting of guest environments in that z/VM can execute in a PR/SM logical partition and can also execute as a guest VM in z/VM.

IBM provides as a basic feature of System z a unified core mechanism for both implementations. This mechanism is itself implemented as a hardware and firmware function in System z and is called the “interpretive-execution” which is available for z/VM and PR/SM. z/VM enters the interpretive-execution mode by issuing a Start Interpretive Execution (SIE) machine instruction.

4.4.1 The System z interpretive-execution facility

This System z hardware and firmware facility aims at providing maximum efficiency for the execution of the programs in the PR/SM logical partitions or z/VM virtual machines. These programs’ instructions are then directly executed by the physical CPU of the host system, under control of the System z hardware which “remembers” that it is acting in interpretive-execution mode, as opposed to proceed with an intermediate simulation layer.

Principles of operation

Interpretive-execution can be seen as a mode of a operation of the System z processing unit where it “remembers” that the current flow of instructions is being executed on behalf of a guest program.As explained in “Security enforcement with interpretive-execution” on page 80,

in interpretive-execution mode the system hardware also enforces the allocation of resources to the guest environment.

In interpretive-execution mode, the real host system detects, or “intercepts”, conditions where simulation should take place. Such a condition is, for instance, the execution of a privileged instruction that would affect the execution environment. These guest programs instructions should not be executed as is on the physical host processor as the environment they target is actually the guest environment. When one of these conditions is “intercepted” by the interpretive-execution mechanisms, the interpretive-execution mode is exited and control is given back to the z/VM hypervisor or the PR/SM firmware so that it can proceed with proper simulation of the function.
Note that before exiting the interpretive-execution mode state information on the interrupted guest program’s flow of instructions is gathered by hardware. This state information can be used to re-enter later the interpretive-execution mode at the point of interception. The needs for simulation are kept by design to a minimum as functions that can be performed when in interpretive-execution mode, if conditions allow, include execution of privileged and problem-program instructions, address translation, interruption handling, timing and I/O operations in some cases.

A fine degree of control is provided in the facility so that the conditions that dictate exiting from the interpretive-execution mode can be selected and adjusted by the firmware or the software that invokes the facility depending on the operating conditions that are detected.

One of the design point of the facility is also to provide optimum execution performance for the execution of the guest programs instructions by timely reporting of asynchronous events, such as interruptions, that require to switch execution context.

Security enforcement with interpretive-execution

The System z implementation of interpretive execution provides for the enforcement of resources allocation and authority granted to the guest as explained below.

Enforcement of resources allocation and isolation

A guest environment must at any time be constrained to the portion of the real resources it is specified to receive from the host system. In interpretive-execution mode hardware registers are used in the host system to specify the portion of host real storage that the guest environment is to use. Access to storage locations is based on the memory configuration defined when entering interpretive-execution mode. In other words access to storage is restricted by the environment defined when the interpretive-execution mode was entered.

Likewise hardware registers are used to provide dedicated facilities to the guest environment such as timing facilities, control registers and pointers to the address translation tables. This also adds to proper isolation between guest environments and prevents any interference between the guests operations and the host system’s own operations. The isolation principle also applies to damages that may be incurred by the guest due to a failure in its operations which are then confined to its own environment.

System level information is also kept in the Channel Sub System hardware registers that is used to provide proper I/O connectivity to each guest environment, enforcing the system’s channels allocation as specified by the system administrator.

Enforcement of guest level authority

The interception mechanism of System z interpretive-execution, because it is implemented as a system driven hardware function only, prevents any guest program to gain undue privileges that would allow to interfere with or penetrate into the operations, including administrative operations, of the physical system

The SIE instruction

Detailed information on the START INTERPRETIVE EXECUTION instruction can be found in "IBM System/370 Extended Architecture - Interpretive Execution", SA22-7095.

The START INTERPRETIVE EXECUTION instruction (SIE) is the instruction that the z/VM hypervisor dispatching mechanism uses to enter the interpretive-execution mode for direct execution of a guest virtual machine instructions by a System z real processing unit.

Note: as the hardware mechanisms that the SIE instructions also exploits at the core of the PR/SM feature, it is also commonly said and written that the PR/SM firmware invokes the SIE instruction.

The operand of SIE is the address of a control block called the "State Description". The state description specifies the information that the hardware needs in order to conduct operations in interpretive-execution mode. This information comprises:

- The area of host real storage that is allocated for the guest operations.
- The contents of program-addressable guest registers - That includes the guest virtual processor PSW and control registers.
- The addresses of control tables to be used by the host hardware and firmware functions.
- Miscellaneous options and controls for the interpretive-execution mode
- Areas where information concerning an interception can be retrieved by the host

When entering interpretive-execution mode the next instruction that the processing unit will execute following the SIE instruction is therefore the guest program instruction pointed at by the PSW in the state description.

As explained previously, the processing unit exits the interpretive-execution mode when an interception condition, or other condition qualifying for giving control back to the hypervisor, is met. The SIE instruction is then completed, that is the execution of the guest program is stopped, and the next instruction in sequence after the SIE in the hypervisor program is executed. This process is illustrated in Figure 4-3 on page 82.

Note that SIE instruction is an interruptible instruction which when interrupted, as it is the case when an interception condition is detected, updates parameters in the state description for further analysis and decision by the hypervisor program. Some of these parameters are the ones needed for continuing the execution of the interrupted guest program. Particularly the guest program PSW is updated with the next instruction address, so that a new invocation of the SIE instruction using the same state description will resume the guest program at the point it was interrupted. It must be stressed that this state description update is performed by the System z hardware and cannot therefore be forged by a guest program.

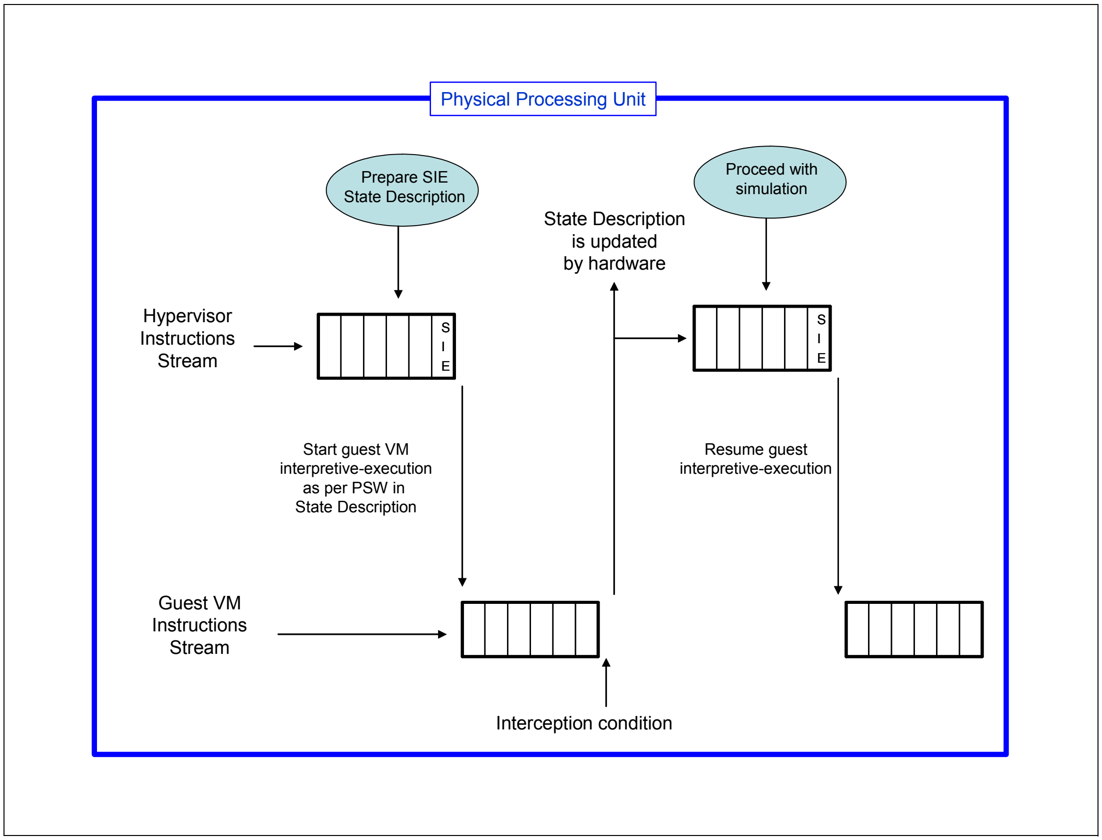


Figure 4-3 SIE execution flow

Nesting guest environments in the interpretive-execution mode

System z allows the nesting of guest environments. Typically a z/VM instance can run in the PR/SM logical partition as a first-level guest, and can create its own guest virtual machines which are then second-level guests. An installation can elect to run another instance of z/VM as a second-level guest which then will create third-level guests and so on.

Although it is possible that installations run with more than three levels of guest environments, it is expected that this be a practical limit and is to be used for non production-oriented purposes such as testing and migrating to new releases of hypervisors.

As of today System z supports two levels of guests operating in interpretive-execution mode. So in our example the z/VM first level guest running above PR/SM is to be run in interpretive-execution mode as invoked by the PR/SM firmware. The z/VM first-level guest can also issue SIE instructions that will put its second-level guests in interpretive mode as well. Assuming that a second-level guest is also a z/VM instance that issues SIE instructions to execute third level guest programs then the second level guest SIE instructions are simulated by the first level z/VM hypervisor, as the host system is already running with two levels of interpretive-execution.

4.5 System z Processor Resource / Systems Manager (PR/SM)

Processor Resource/Systems Manager is a System z hardware and firmware facility that enables the resources of a single physical machine to be divided between distinct, predefined virtualized environments called “logical partitions”. Each logical partition can be seen as a guest environment with proper controls for loading and controlling the execution of programs as if the logical partition were a standalone real system.

As of System z model z990, operating the system in PR/SM mode, as opposed to native basic mode, is not any more an option: the system runs configured with at least one logical partition. As of the writing of this book the IBM System z can accommodate for up to 60 logical partitions concurrently executing in one physical system.

With PR/SM workloads that need to be separated can still run on a single physical system in different logical partitions with the benefits of physical consolidation in terms of economy, management and security. PR/SM also provides the required flexibility when it comes to cope with variations in the workloads characteristics that dictate frequent re-allocation of computing resources.

We will discuss at length in this chapter the integrity and security characteristics of the PR/SM implementation in System z that provide for proper isolation of operations between partitions and controlled access to the logical partitions controls. These characteristics have been evaluated as meeting the Common Criteria (ISO 15408) EAL5 level of evaluation, using a specific protection profile. The certification proof points are also discussed in this chapter.

> Note that the terms “PR/SM” and “LPAR” are used quite often interchangeably, including in IBM documentation. Actually LPAR designate the logical partitioning function and mode of operation, whereas PR/SM is the commercial designation of the system’s feature.

Which reference documents to use

Details on what is covered in this chapter can be found in the following IBM documents:

- “System z10™ Processor Resource/Systems Manager Planning Guide”, SB10-7153.
- “System z Input/Output Configuration Program User's Guide for ICP IOCP”, SB10-7037-07
- “System z10 Support Element Operations Guide”, SC28-6879-00

4.5.1 PR/SM architectural components

We are giving in Figure 4-4 a very high level description of the architectural components of PR/SM implementation.

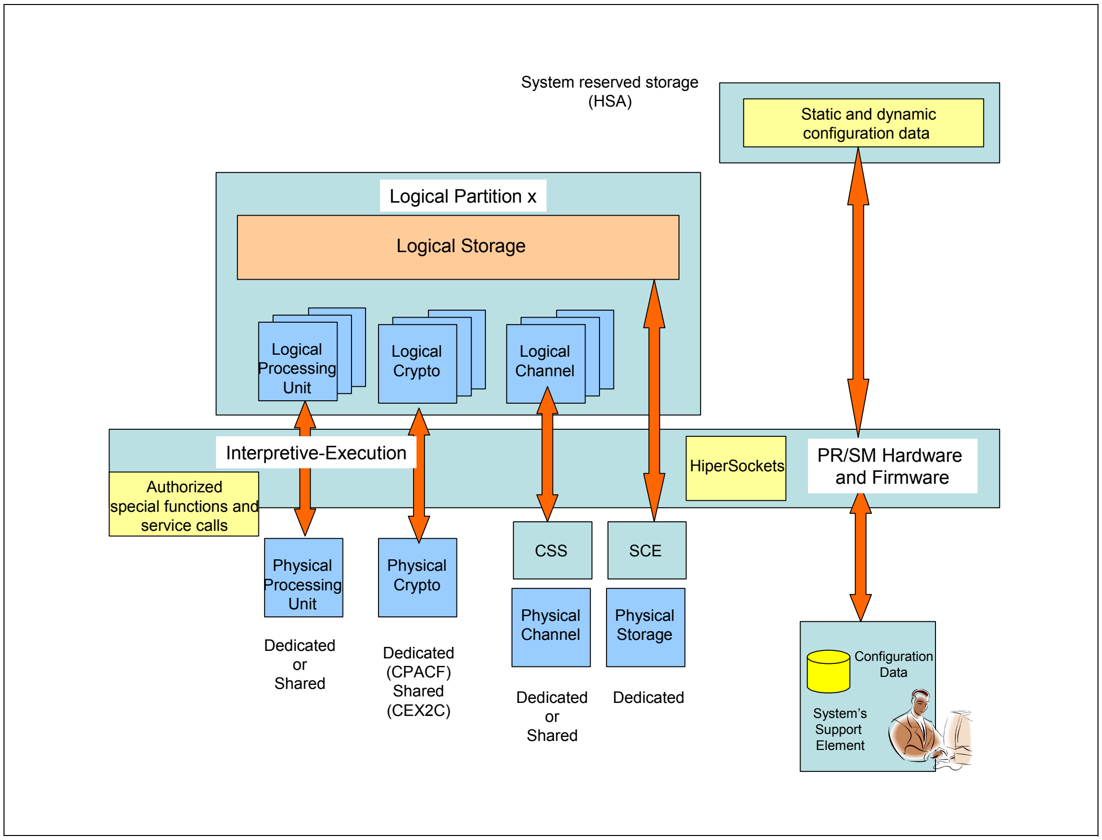


Figure 4-4 PR/SM architectural components

Logical and physical resources

Programs running in the logical partition exploit logical resources such as

- Logical Processing Units - We use the term “processing unit” (PU) here to designate the several types of processors that can be available in System z. The logical partition can provide logical CPs, logical IFLs, logical zAAPs and logical zIIPs.
  The logical partition can provide logical IFCs (if available on the system) if it is to run the Coupling Facility Control Code.

  As shown in the figure, the logical PUs are backed by physical PUs that are to execute the logical partition’s programs in interpretive-execution mode and execute the PR/SM firmware instructions when not in this mode. The instantiation of a logical PU can be seen as the creation, by the PR/SM firmware, of a state description control block to be used when entering or exiting the interpreted-execution mode. See , “The SIE instruction” on page 81 for more information of the state description control block.

  When defining the logical partition, the user can define logical PUs that will share the available physical PUs. The sharing of these physical resources is controlled by the PR/SM firmware which provides for strict allocation of the specified logical PUs and optimized time slicing algorithms.

  The logical partition can also be defined to use dedicated physical PUs. In that case the logical partition is not to share the use of the physical PU with another partition.

  Note that by design an individual logical partition cannot get more logical PUs than the amount of physical PUs available in the system. For a logical partition sharing physical PUs it means that a logical partition cannot be allocated more than the amount of non-dedicated physical PUs still available in the system.

The allocation of logical PUs is specified in the image profile for the logical partition. The contents of a logical partition's image profile is described in 4.5.4, "The logical partition Image Profile" on page 92.

- Logical cryptographic coprocessors - System z comes with integrated hardware cryptography capability. As of the writing of this book there are two types of cryptographic facilities available in System z:
  - The Central Processor Assist for Cryptographic Functions (CPACF)- The CPACF is a facility embedded in the physical PU and is invoked by specific instructions that the executing programs issue in native or interpretive-execution mode. The CPACF provides cryptographic functions performed at a very high speed, however it does not have the capability of preserving secrets such as the value of the encryption key: the requesting program provides this value when needed and is responsible for protecting it.

    There is no specific resource allocation required for the CPACF and the notion of sharing is not relevant as it is integrated in the physical PU: as soon as the PU executes instructions for a guest program, then the program can exploit the CPACF, if properly coded.
  - The Crypto Express 2 (CEX2) - The CEX2 is a physical facility that plugs into slots located in the system I/O cage - There can be as many as 8 CEX2 features installed (for a total of 16 coprocessors). Conversely to the CPACF the CEX2 coprocessor has been designed to securely keep specific installation's secrets known as the "Master Keys". In order to make the CEX2 coprocessor sharable between logical partitions the coprocessor comes with 16 physical "domains". One can think of the "domain" as the set of facilities needed to keep and protect a set of Master Keys.
    As a consequence the CEX2 coprocessor can be shared between up to 16 logical partitions, each one of these partitions being allocated a physical "domain" in the coprocessor by the system administrator. The logical partition users can therefore keep their Master Keys protected in their own coprocessor domain, fully isolated from the other domains.

    This makes the implementation of resource-sharing in the cryptographic coprocessor somehow peculiar in that the domains are not sharable but the imbedded cryptographic engines are. The allocation of logical CEX2 to a logical partition is done through the image profile of the partition.

    More details on the setup required to share CEX2 coprocessors along with the use of the TKE (Trusted Key Entry) workstation in a PR/SM environment are given in 4.8, "A few more words on logical partitions and cryptographic coprocessors" on page 99.
- Logical channels - Here the term "channel" has a very broad meaning in the System z context. It designates ESCON® or FICON® channels, but also the OSA adapter or the Coupling Facility links. These makes approximately twenty channel types the user can choose from to allocate to the logical partition.

  The initial allocation of channels to logical partitions is specified in the IOCDS (I/O Control Data Set) file which is built by the system administrator and resides in the Support Element (SE) of the system. More details on the IOCDS are given in 4.5.3, "Overview of the logical partitions configuration information" on page 89.

  Logical channels are backed by physical channels, except for the IQD type of channel, also known as HiperSocket. HiperSockets consist of a simulated Ethernet LAN that provides inter-partition TCP/IP connectivity within the PR/SM firmware layer. Additional considerations are given to HiperSockets in, "HiperSockets" on page 96.
- Logical storage - The logical partition logical storage is fully backed up by physical storage. Actually they are both one block of contiguous addresses in the system's physical storage. Proper physical address prefixing and physical storage protection is provided by

the System z hardware so that the physical storage block is reserved and completely isolated for the sole use of the logical partition.

There is no such thing as paging or moving the logical partition physical storage contents. However PR/SM allows, with proper setup, a portion of the logical partition storage to be dynamically released. If necessary, the released storage can then be dynamically re-assigned, still with proper setup, to another logical partition. Release and re-assignment of the storage block is performed under control of the guest operating system (z/OS or z/VM) which sends the proper Service Calls to the SE. The PR/SM hardware and firmware verifies that the storage reconfiguration conforms to the releasing and receiving partitions specifications of storage allocation.

Note that the logical partitions do not have their own paths for accessing storage. They use the native storage access components, that is the System z Storage Controller Element (SCE), after proper adjustment and verification of the physical storage address by the PR/SM hardware.

Special functions and service calls

Along the years, specific functions have been implemented in PR/SM to satisfy user requirements for advanced capabilities that require communications between the guest operating system and the PR/SM firmware. Examples of such advanced function is the capability for one guest program to collect performance data pertaining to the complete physical system, or to the capability to have a logical partition programatically de-activating another logical partition.

Obviously these functions, although proven to satisfy users requirements and being access controlled in the requesting programs, conflict with the goal of achieving strict isolation between logical partitions. Therefore these functions can be manually restricted at the logical partition level by the system security administrator.

See “Logical partition security options” on page 93 for more details on these controls.

States of a logical partition

A given logical partition has first to be defined, that is given an image profile by the system administrator.

A defined logical partition can be in either one of the following states:

- Deactivated - The logical partition has not been instantiated yet in the physical system and does not have any access to its defined resources.

  Note that the resources defined to a de-activated partition can be preempted by an active one which has been defined and allocated the same resources.

- Activated - The logical partition has been instantiated in the physical system. Activation succeeds if all the non-sharable resources defined to the partition are not already preempted by another active logical partition. In an activated logical partition, guest programs are “dispatched” onto physical PUs.

  > **Important:** when a resource is allocated to a logical partition, it is set to its architecturally-defined reset state, that is, for instance, channel paths are reset, main storage is zeroed.

- Activated and locked - When in a locked state, controls that pertain to the logical partition and can potentially induce functions disruptive to the partition operations are not accessible to the system administrators. It takes an explicit manual operation by the system administrator to set and reset the locked state at the SE.

IPL in the logical partition

An Initial Program Load (IPL) resets a logical partition to prepare it for loading an operating system, and then loads the operating system using the specified IPL address and IPL parameters.

Reporting logical partitions performance

RMF (Resource Measurement Facility) can be run in a logical partition hosting z/OS. RMF exploits a specific interface with the PR/SM firmware to gather data at the physical system level so that the Partition Data Report can provide an information on how efficiently the logical partition operates in the physical system, in terms such as LPAR management overhead and physical PUs utilization.

4.5.2 The role of the Support Element

The SE is a key component for the administration of the system and of logical partitions. It keeps the configuration information related to logical partitions that are needed at Power On Reset of the system, when activating a logical partition or when dynamically changing the I/O configuration of the system. The service element provides for proper control and serialization of accesses to this information as explained in "User identification and authentication at the SE" below. It also audits security events that include events relevant to the administration of the system and logical partitions.

Although in production environment actions at the SE are actually initiated at an HMC then relayed to the SE, we will consider here, for the sake of simplicity, that interactions are performed at the SE itself.

The SE mediates all requests related to the logical partitions configuration between the system administrator and the PR/SM firmware. It also performs this mediation for guest programs that issue service calls.

User identification and authentication at the SE

All tasks that can be performed at the SE, including those affecting the logical partitions configuration or operating conditions, are submitted to access control after proper authentication of the user. We are reminding below the user authentication and authorization mechanisms implemented in the SE application.

User authentication

Each user who has to access functions of the Support Element must be properly registered in the SE by a person with Access Administrator authority. The user information is kept in a user profile and consists of the user identity, password and authorities, or role, granted to the user.

User roles

Each SE user must be registered at the SE in one or many of the following categories of authorities, or roles:

- Operator - this is the authority to perform basic system operations using the predefined control functions.
- Advanced Operator - this adds to the operator authority the capability of performing some system recovery and maintenance tasks.
- System Programmer - basically this authority allows to customize the system in order to determine its operation.

- Service Representative - this is the authority to access tasks related to the repair and maintenance of the system. Note that the system should be switched to Service Mode prior to performing tasks authorized to the Service Representative. Note also that most of the SE tasks can be performed by the authenticated Service Representative when the system is in Service Mode.
- Access Administrator - this is the authority to create, modify, or delete user profiles in the SE.

The SE comes with initial default user names, roles and password that the Access Administrator should remove when registering real users.

Access control

SE tasks are implemented with a specific requirement about the role the requesting user should be in order to be authorized to perform the task. As an example the tasks related to logical partitions customization and controls require the user to be in the System Programmer role.

> **Note:** "System z10 Support Element Operations Guide"- SC28-6879 is specifying for each SE task what the required authority is.

Mapping the SE roles to organizational responsibilities

Considering the system's management tasks and the roles they require, as specified in the Support Element Operations Guide, the following two categories of administrative authorities can be defined and referred to when considering what privileges are required:

- Security Administrator – a Security Administrator is any SE user who is defined with a role of System Programmer or Service Representative.
- System Administrator - a System Administrator is any user with access to the SE.

Audit data in the SE

The SE automatically keeps a log of security events that occur while the SE application is running. A security event occurs when a resource's operational state or settings are modified and every time a SE user accesses tasks, actions, and resources.

Access to the security log is granted for Administrator, System Programmer or Service Representative user roles.

> **Note:** there is no capability provided to guest programs to get access to the Security Log.

As an example the auditable security events include:

1. The creation or modification of the IOCDS file
2. Any dynamic re-configuration
3. performing a Power-on Reset
4. activating or deactivating logical partitions
5. logging on or off the SE

The log entries are in chronological order and provide an audit log. Entries also include a user (system administrator) identifier when appropriate and a reliable timestamp (timestamps are retrieved from the HMC/SE hardware clock which is periodically synchronized with the other hardware clocks in the system).

Care must be taken however to periodically archive the Security Log as it automatically deletes the oldest records when at its maximum capacity. Proper details on the pruning mechanism and procedure to archive the log contents on removable media are provided in the Support Element Operations Guide.

Locking

For enhanced integrity of execution, locking of partitions is recommended. The partition must then be unlocked before other disruptive operations can be performed on that partition.

Locking or unlocking is achieved by invoking the function at the SE.

Support Element (SE) availability

System z comes with a backup SE. Functions are implemented that permit a quick switch to the backup Support Element when the primary one has a hardware problem. Mirroring functions are performed on a regular basis to communicate any hard disk changes from the primary SE to the alternate SE.

4.5.3 Overview of the logical partitions configuration information

The PR/SM logical infrastructure is shown in Figure 4-5 on page 91 where:

- The physical system is hosting the PR/SM hardware and firmware layer and the activated logical partitions.
- Initial configuration information that is needed by PR/SM is stored on disks in the system's Support Element (SE). There are three files that hold this information:
  - The IOCDS (I/O Control Data Set)
  - The Reset Profile
  - The image profile

The IOCDS file

The IOCDS file contains the description of the physical I/O configuration of the system, in terms of channel paths, control units and I/O devices. The system administrator who builds the IOCDS file using the HCD (Hardware Configuration Definition) or IOCP (Input/Output Configuration Program) program also specifies:

- what are the logical partitions defined in the system - Each logical partition is given a unique name.
- which logical partition is granted access to which physical channel path.

The IOCDS file can be protected from inadvertent or unauthorized write by the system administrator.

The definition of the channel paths in the IOCDS also includes information about the logical sharing of channel paths between logical partition, as explained in "The logical partition Image Profile" on page 92.

The SE can host up to four IOCDS files, the one to use for the system Power on Reset is indicated in the system's Reset Profile.

> **Important:** the description of the I/O configuration in the IOCDS includes the definition of the channels "Candidate List". Proper keywords are used to specify which logical partitions have access to a given channel path. When the system is initialized this information is preserved in several functional elements including the Channel Sub System hardware with the result that no channel path can be allocated to a logical partition if the logical partition is not in the candidate list for the channel path as specified in the IOCDS.
>
> Candidate Lists are discussed in more details in , "Considerations on I/Os sharing" on page 97.

The Reset Profile

The Reset Profile file contains global options and definitions related to the physical characteristics of the system and is accessible through screen menus at the Support Element. One of these definitions is the name of the IOCDS file to use for the system Power on Reset.

One set of options recorded in the Reset Profile specifically pertain to the PR/SM operations. They are:

- The enablement of I/O requests prioritization according to options in the image profile of each logical partition ("Enable global input/output (I/O) priority queuing")
- The programmatic issuance of a reset signal down to the physical I/O interface ("Automatic input/output (I/O) interface reset").
- Logical partitions scheduling options:
  - The user can fix the value of the time slice duration logical partitions can remain dispatched on a physical processing unit or can have PR/SM deciding on the optimum value of this time slice ("Time Dynamically determined by the system" and "Processor running time" options).
  - When fixing the value of the time slice, the user can also force the logical PUs of logical partitions to remain dispatched on physical PUs even if the guest program is in wait state ("Do not end the timeslice if a partition enters a wait state").

> **Note:** letting the logical partitions being dispatched as per the PR/SM built-in optimization algorithms fits most of the workloads. However certain workloads have characteristics that do not lend themselves to efficiently operate when the PR/SM scheduling algorithms are used. An example of such workloads would be a program that enters the wait state for very short periods of time. If PR/SM releases the physical CP at wait state entry and should almost immediately re-dispatch the same program because the wait state ends, this is undue overhead affecting this program throughput. In that case the options will help maintaining the workload throughput of the logical partition.
>
> In the Security terminology these specific types of workloads might be driven in extreme cases to enter a denial-of-service condition, if this capability of manually adapting the PR/SM scheduling algorithms were not here.

The Image Profile

There is one Image Profile file for each logical partition that can be activated in the system. We specifically discuss the image profile contents at 4.5.4, "The logical partition Image Profile" on page 92.

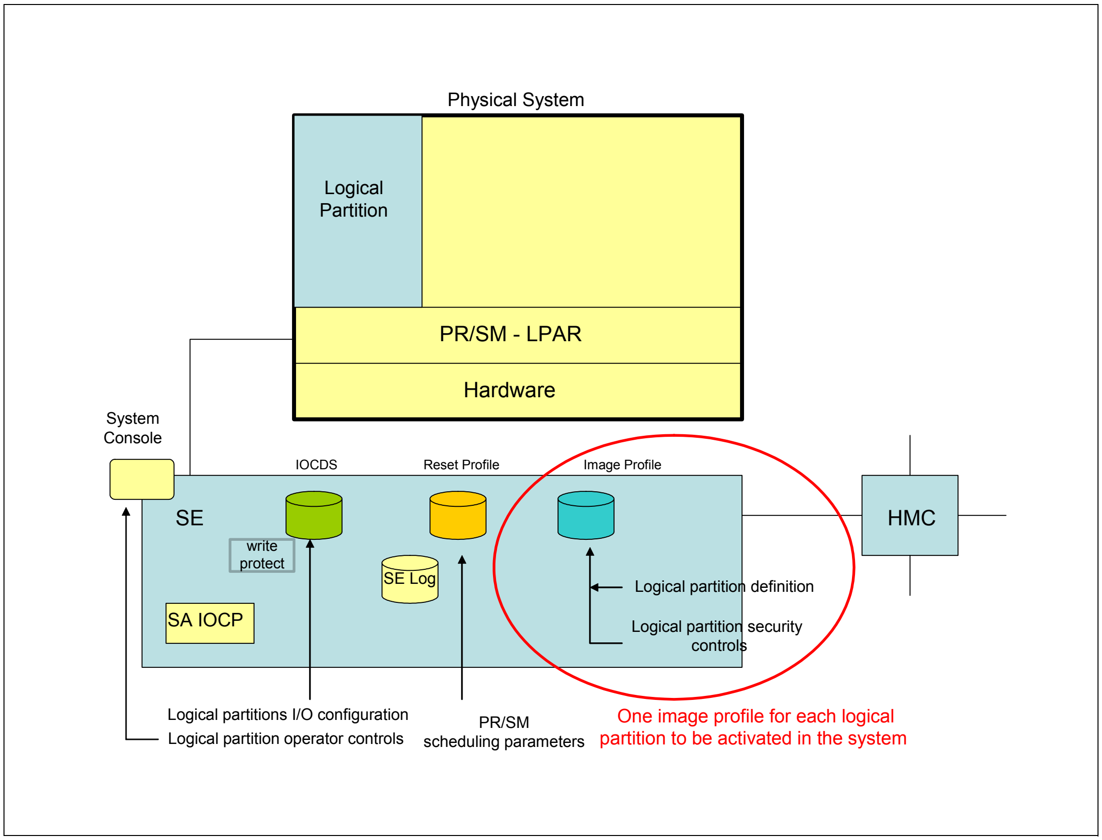


Figure 4-5 PR/SM logical infrastructure

The SE hosts the initial configuration data for the physical system and the PR/SM logical partitions. It also provides the utilities needed for maintaining these data:

- The reset profile and the image profiles can be edited via menus at the SE or HMC display.
- The IOCDS file can be created and maintained using the IOCP program, that comes as a program running under System z IBM operating systems and as a standalone version, that is a program which does not need an operating system to execute. The standalone version of IOCP is used for preparing a not yet initialized configuration and the load module resides in the SE hard disk.
  Installations tend today to use the more versatile HCD program (running under z/OS or z/VM) to administer their I/O configuration once the operating system is running in the logical partition.

  Both the IOCP and HCD programs have a dedicated hardware access path to the IOCDS file in the SE. It is important to note that the HCD program provides the capability of dynamically changing the contents of the IOCDS.

  The following controls are in place to restrict write access to the IOCDS:

  – The file can be put in write-protected state, as specified at the SE by the Security Administrator

  – The logical partition executing IOCP or HCD should have the I/O Configuration Control option enabled in its image profile.

  – If the stand alone IOCP is to be used it requires the Security Administrator authority to control its execution from the SE.

All IOCDSs should be in a write-protected state except for the few minutes during which they are actually updated.

> **Important:** these configuration data are key to the operating integrity of **PR/SM** and must therefore be properly protected. **System z** provides the required **access control** mechanism as explained in **4.5.2, "The role of the Support Element" on page 87**. However it is highly recommended that adequate **physical security** be in place, such as:
>
> - The hardware and any networks used to connect the hardware must be physically secure.
> - Access to **I/O** devices must be restricted to authorized personnel.
> - The **HMC** must be physically protected from access other than by authorized system administrators.

4.5.4 The logical partition Image Profile

The resources, other than I/O facilities, that are planned to be allocated to a logical partition must be specified in an Image Profile file dedicated for this partition - the name of the file is the name given to the logical partition in the IOCDS. The image profile is manually built at the Support Element and maintained using predefined menus on the SE display. It records the logical partition characteristics with in addition options for automatically proceeding with an IPL of the logical partition after its activation.

Logical partition operating characteristics

Many characteristics are assigned to the logical partition in the Image Profile file:

- The identification of the partition - A unique "Partition Identifier" must be entered for he partition. The Partition Identifier is an hexadecimal value (X'00' through X'3F') which is to be used to identify, whenever necessary, whatever data PR/SM produces that relate to this partition.
- The mode, which actually dictates the set of System z architected functions that will be supported in the logical partition.
- How many logical processing units are allocated to the partition. With the option of running in a dedicated mode, that is the logical partition does not share the physical processing units with other partitions.

  > **Note:** dedicating a physical processing unit to a logical partition implies anyway some level of sharing: the physical PU is actually shared between the execution of the guest program and the execution of the PR/SM firmware.

- The logical partition relative share of shared PU resources is expressed in "weight".
- The size of the system's physical storage the logical partition should be allocated.
- The cryptographic coprocessors and the "domains" in the coprocessors the logical partition has access to. It is also specified here what other domains could be controlled from this partition when it operates as a remote host for a TKE (Trusted Key Entry) workstation.

  We are further discussing the allocation of cryptographic coprocessors to logical partition in "A few more words on logical partitions and cryptographic coprocessors" on page 99.
- Security options for the logical partition (see , "Logical partition security options" below).

- Miscellaneous characteristics such as a specific TOD offset to be given to the partition, the IPL parameters if the installation elects to trigger an automatic IPL once the partition is activated.

Logical partition security options

The following options are available in the logical partition image profile to allow or restrict functions that induce specific transfers of information between the physical system and the logical partition.

- Global performance data control
  This option is selected if the logical partition is to view the CPU utilization data and the Input/Output Processor (IOP) data for all logical partitions in the configuration. As it would be needed if the logical partition hosts a z/OS system running RMF to collect overall performance data.
  Not selecting this option only allows the logical partition to view its own CPU utilization data.
- Input/output (I/O) configuration control
  This option allows to access the SE IOCDS file in read or write from the logical partition configuration and to make dynamic I/O changes. Additionally, this parameter allows the OSA Support Facility for z/OS, z/VM, and z/VSE to control OSA configuration for other logical partitions.
- Cross partition authority
  This option allows proper program running in the logical partition to send physical system level architected messages to PR/SM so that actions are taken against other logical partitions. Such actions can be: perform a system reset of another logical partition, or deactivate another logical partition.
- Logical partition isolation
  When this option is selected unshared reconfigurable channel paths allocated to this logical partition cannot be released and moved to another logical partition. Unshared reconfigurable channel paths are explained in.4.7.1, “Allocation of channel paths to logical partitions” on page 94.
- Miscellaneous options are also available that allow or restrict the collection of performance data from specific hardware counters in the physical system.

Security settings are saved by the system across activations for the current configuration. Therefore, if the same configuration is used, Security settings need not be reentered (but should be checked).

4.6 Reconfiguration of logical partitions

The configuration data in the IOCDS and System Profile are used to initialize the system at Power On Reset. The configuration data in the Image Profile are used to initialize the logical partition. Elements of the physical system can be dynamically, that is without disrupting the system’s operations, reconfigured if permitted in the initial setup. As an example PUs or channels could be taken off-line, or on the contrary can be dynamically added to the configuration. The following logical resources can be dynamically reconfigured in a logical partition:

- logical processing units
- logical channels and attached I/O devices
- storage

Initial configuration data that reside in the SE are also copied to a system reserved storage. the Hardware System Area (HSA), at initialization time. When dynamic reconfiguration occurs the configuration data in the HSA are adjusted accordingly and eventually recorded in the SE. This provides for immediate effect of the reconfiguration action and also for the preservation on the SE hard disk, and its mirror in the backup SE, of all reconfiguration changes that occurred since Power on Reset or logical partition activation.

The reconfigurable part of a logical partition comprises the storage and logical PUs resources that may be allocated to the partition.

4.6.1 Logical partition storage reconfiguration

A logical partition is allocated at activation the amount of contiguous physical storage specified in its image profile, unless not enough physical storage is left by the currently active partitions in which case the logical partition activation fails.

It is possible, if supported by the operating system which runs in the logical partition to dynamically deallocate a contiguous piece of the partition's storage. It is also possible to reallocate the released storage to another activated logical partition. Every logical partition has in its image profile a storage range definition consisting of:

- an initial amount of central storage to be allocated to the logical partition at activation.
- a reserved amount which determines how much additional storage can be acquired by the logical partition using dynamic storage reconfiguration.

Important: PR/SM always clears the storage portion that is released.

4.6.2 Reconfiguration groundrules

The reconfiguration of a logical partition can be performed:

- By a user working at the SE with the Security Administrator authority
- By a guest program issuing Service Calls.This is to work under the following conditions:
  - the partition is allocating to itself, or deallocating from itself logical PUs or storage.
  - the logical partition targets another partition for de-activation or reset, and it has the Cross-Partition Control option enabled.

As already mentioned, the PR/SM firmware ensures that the contents of physical processors, storage or I/O utilized by different logical partitions will be cleared of any residual information before being utilized by the receiving logical partition.

4.7 More on PR/SM Logical Partitioning and I/O configuration

In this section we briefly describe the specific characteristics of the System z channel paths that relate to the way they are allocated to PR/SM logical partitions.

4.7.1 Allocation of channel paths to logical partitions

The System z channel paths are allocated to logical partitions as per the I/O configuration definitions in the IOCDS. A channel path can be specified as:

- Shared channel path - The MIF (Multiple Image Facility) feature of System z is then exploited to give each logical partition permitted to access the channel path a logical view of the resource. That is the physical channel path itself appears as a unique virtual resource to each one of the partitions.

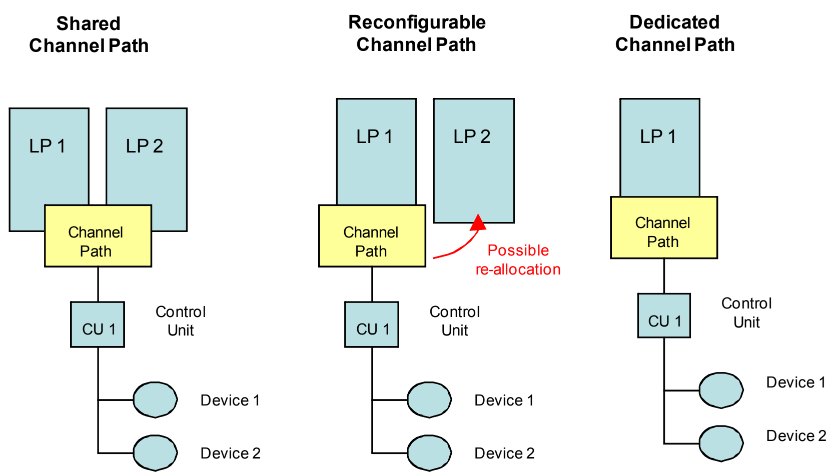
  If the IOCDS does not specify sharing, then no sharing of channel paths will take place.

  Note that accesses to I/O devices through a shared channel path can also be controlled in the IOCDS in terms of which logical partitions that share the channel path have access to which I/O devices.

- Reconfigurable channel path- A reconfigurable channel path is initially allocated to one logical partition but can, via an operator intervention, be moved to another logical partition. Note however that this channel path is allocated to only one logical partition at any point in time.
  The channel path candidate list is used to limit the "mobility" of a reconfigurable channel. PR/SM will only accept reconfiguration commands for channel paths in partitions specified in the candidate list of the target channel path.

- Dedicated channel path - A dedicated channel is initially allocated to one logical partition and this allocation cannot be changed.

We are summarizing how channel paths can be allocated to logical partitions in Figure 4-6. We are also providing in this figure a brief information on how the initial allocation can be modified and how this modification can be controlled. We are explaining farther these controls at 4.7.3, "Consideration on channel paths reconfiguration" on page 98.



Initial allocation	Via IOCDS	Via IOCDS	Via IOCDS
Dynamic reconfiguration w/o changing the IOCDS	possible	possible	Not possible
How the reconfiguration can be performed	Via operator intervention -At the operating system -At the SE	Via operator intervention -At the operating system -At the SE	IOCP or HCD
Logical Partition reconfiguration controls	LPAR I/O Configuration Control Authority	LPAR Isolation Control	LPAR I/O Configuration Control Authority

Figure 4-6 Logical partitions channel paths allocation

OSA considerations

OSA (Open System Adapter) from the logical partition allocation standpoint it is considered as a channel path, of a specific type though, and can therefore be allocated as shared, reconfigurable or dedicated.

Access to an OSA port can be shared among the logical partitions to which the OSA-Express channel path is defined to be shared. Also, access to a port can be shared concurrently among TCP/IP stacks in the same logical partition or in different logical partitions. When ports are shared as described above, the OSA features have the ability to send and receive IP traffic between logical partitions, or between TCP/IP stacks in the same partition, without sending the IP packets out to the physical LAN and then back to the destination logical partition.

HiperSockets

HiperSockets are the simulation, within the PR/SM firmware, of an ethernet network and the physical adapters that would be required to connect to such a network if it were a real one. It actually simulates an OSA-Express link control layer and assigns the adapters MAC addresses from internal tables maintained by the PR/SM firmware.

HiperSockets provide high speed communications between TCP/IP hosts connected to the virtual network as, in reality, these communications end up to be memory-to-memory transfer performed and controlled by the PR/SM firmware.

From the IOCDS definition stand point HiperSocket is also considered as a channel path of a specific type (Internal Queued Direct Communication - IQD - channel path) that can be allocated to logical partitions as shared, reconfigurable or dedicated.

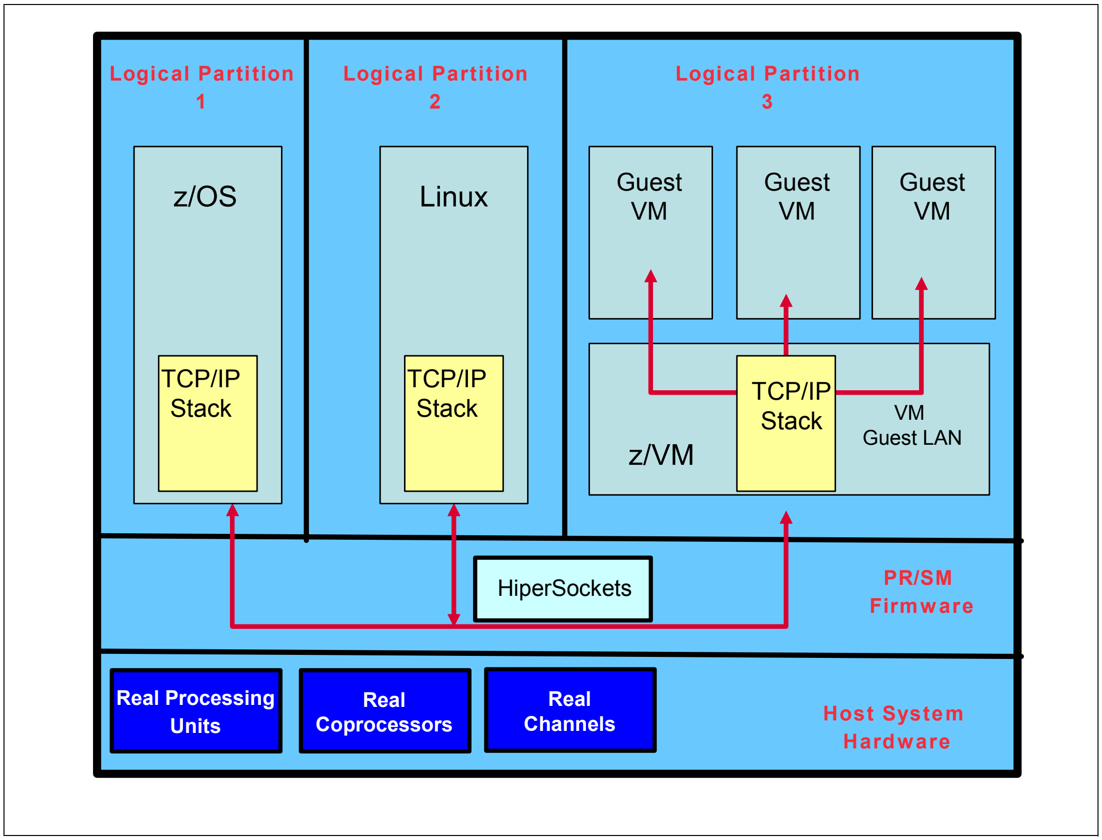


Figure 4-7 System z PR/SM and HiperSockets

Note: HiperSockets provide an interesting networking approach from the security standpoint when it comes to interconnect logical partitions all residing strictly in the same physical system: the network segments do not exist (as they are simulated by PR/SM) and as such cannot be attacked by any physical means.

It is true obviously that they remain vulnerable to any programmatic attack that would be launched from one of the connected logical partitions. One must also realize that conventional protections that TCP/IP hosts use to run are also effective with HiperSockets as they do not differentiate from any other network as seen by the TCP/IP host.

Considerations on I/Os sharing

Use of a shared channel path allows the possibility of two partitions having access to the same I/O control units and devices. Although a shared channel path is defined to be shared, none of the devices that are connected to it need to be shared among logical partitions. Which partitions share the device is indicated in the IOCDS. What I/O devices are shared is highlighted in the IOCP configuration report.

When devices are assigned to a single logical partition, they cannot be accessed by any other logical partition.

4.7.2 Candidate List and Access List

The information in the IOCDS statements is also used to build system internal lists, called the channel path Candidate List and Access List, that affect the allocation of resource, at logical partition activation:

- The Candidate List defines which logical partitions can configure on-line the channel path following the partition activation. In other words the candidate lists specify which physical channel paths a logical partition has potentially access to.
- The Access List defines which logical partitions will have the channel path configured online at the activation of the partition following an initial power-on reset of the system with this very IOCDS.
- The Device Candidate List defines which logical partitions can access the device following their activation. It is used for devices attached to shared channel path. It allows to select among the logical partitions that share the channel path which ones have access to the device.

It is relevant to note here that a control unit is allocated to a partition if a channel path to which it is attached is allocated to the partition.

4.7.3 Consideration on channel paths reconfiguration

Even if a channel path has been designated as reconfigurable, that channel path cannot be removed from a logical partition unless the channel path has first been taken offline from within that logical partition. If the partition executes z/OS this is done using a z/OS operator command (CONFIG). For partitions executing other operating systems, the channel path is removed by direct operator intervention at the SE.

When a channel path is deconfigured from a logical partition, each subchannel (an internal structure that provides the logical appearance of an I/O device, and is uniquely associated with one I/O device) for which this channel path was the only remaining online path, is removed from the logical partition. Before the subchannels are removed, they are drained and disabled. Subsequently the channel path is reset. If the channel path being deconfigured is the last channel path to a physical device, that device is also reset.

At that very first use of a newly created IOCDS, activation configures all channel paths to the logical partitions as defined by the IOCDS. The subsequent movements of reconfigurable channel paths, from one logical partition to another, are remembered by the system. During subsequent activations, as each logical partition is activated, if a channel path was (previously) moved out of a logical partition, the channel path is taken offline to that logical partition; if a channel path was moved into a logical partition, the channel path is brought on line to that logical partition.

4.7.4 Summary of ground rules and restrictions

The System z PR/SM implementation enforces the following ground rules and restrictions:

- a channel path can only be allocated to a logical partition if that partition has candidate access to the path
- a logical partition can be prevented from using a shared channel path.
- a channel path can be allocated exclusively to one logical partition either by identifying the channel path as dedicated, or by designating the owning partition as isolated.
- a reconfigurable or dedicated channel path is never shared.

- a control units and I/O devices cannot be allocated independently of the channel path to which they are attached.
- A reset signal is sent to a non-shared channel path and its attached I/O devices before allocation of the channel path to a logical partition.
- A channel path, shared or not shared, cannot be used if it appears off-line to the guest programs.

4.8 A few more words on logical partitions and cryptographic coprocessors

As already seen, each CEX2 coprocessor is hosting 16 physical “domains”, or sets of physically and logically secure registers where the Master keys of each sharing logical partition can be safely kept. the logical partition Image Profile specifies the coprocessor and the domain the logical partition has access to.

As by construction a coprocessor can be used by no more than 16 logical partitions, if more than 16 partitions need access to cryptographic services, then additional coprocessors must be installed in the system to guarantee that each activated logical partition that uses coprocessors will have its own dedicated physical domain.

Once an activated logical partition has been allocated a specific domain in a specific coprocessor, as specified in its Image Profile, the same domain in the same coprocessor cannot be allocated to another activated logical partition.

This approach is illustrated in Figure 4-8 where 3 logical partitions have been activated that run operating systems supporting the System z hardware cryptographic facility.

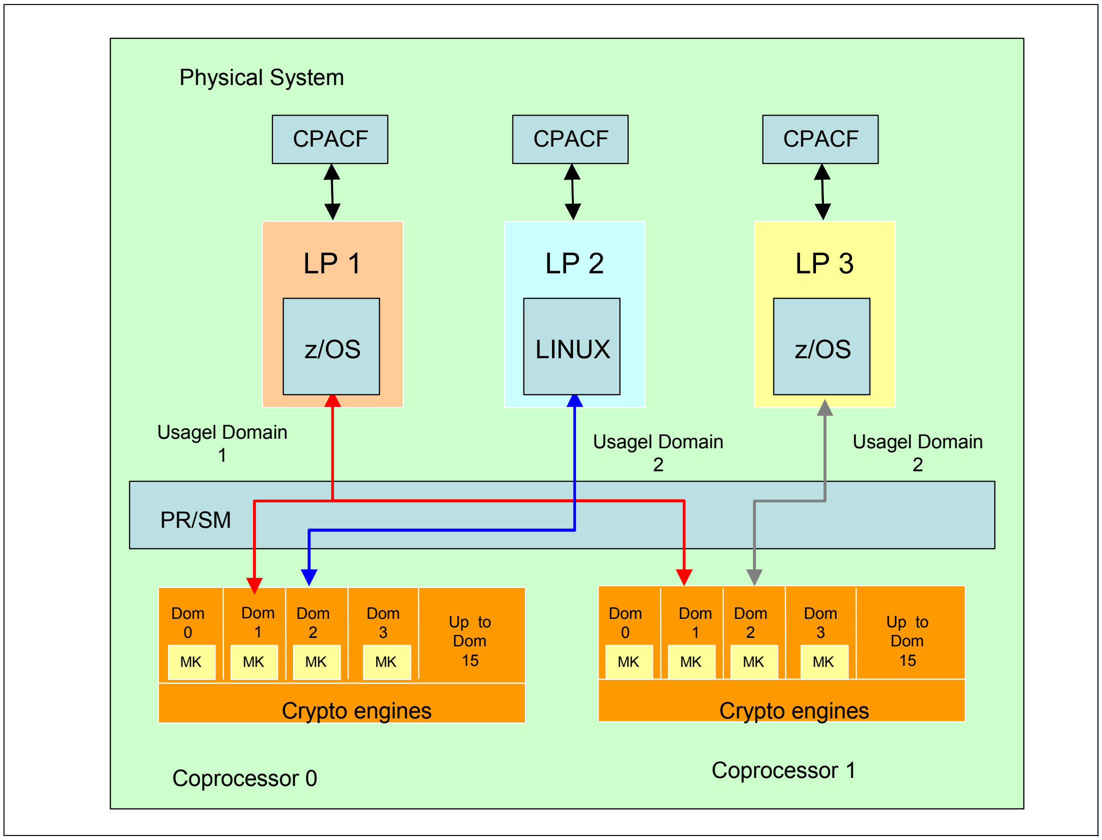


Figure 4-8 Allocation of CEX2 domains to logical partitions

As already mentioned logical partitions have de facto access to the CPACF as it is part of the physical PU. In this example the uniqueness requirement for the combination of coprocessor number and domain number allocated to the activated partitions is satisfied:

- The image profile of logical partition 1 specifies that the logical partition is to be given access to domain 1 in the physical coprocessors 0 and 1.
- The image profile of logical partition 2 specifies the allocation of domain 2 in coprocessor 0.
- The image profile of logical partition 3 specifies the allocation of domain 2 in coprocessor 1

Note that figure refers to “usage domain”. We are explaining below what usage domains and control domains are.

Usage domains and control domains

Usage domains are what we just described: the domains in the CEX2 that logical partitions can use to safely keep their secret Master Keys.

Control domains refer to the use of a special cryptographic feature of System z called the TKE (Trusted Key Entry) workstation. The TKE workstation is used to centrally administer secrets in the CEX2 coprocessors that are connected via TCP/IP to the TKE. All operations at the TKE and network communications are highly secured. The infrastructure layout is represented in Figure 4-9.

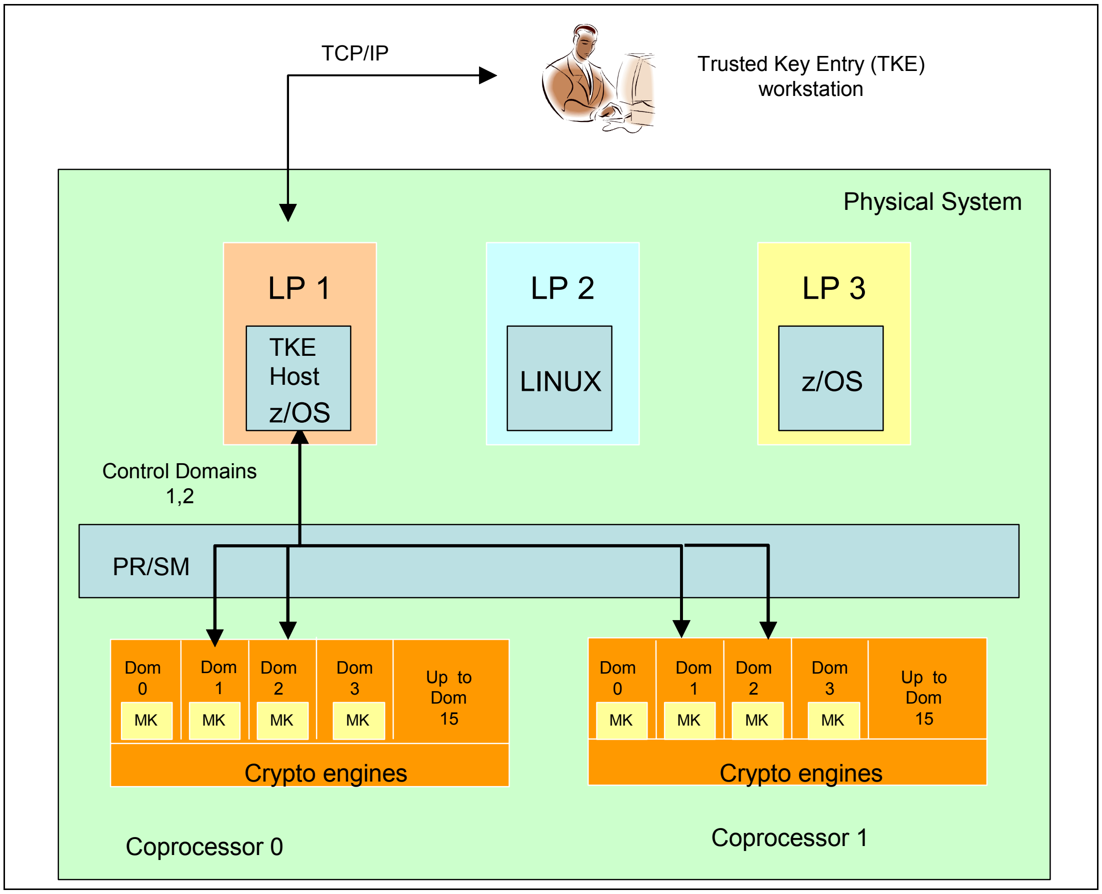


Figure 4-9 TKE infrastructure and control domains

The TCP/IP communications with the TKE are handled by a logical partition designated as the “TKE host”.

The domains in the in the physical coprocessors that can be administered from the central TKE are specified in the image profile of the TKE host logical partition. They are called the “control domains”, these are domain 1 and domain 2 in Figure 4-9.

Cryptographic coprocessors Online and Candidate lists

Similarly to the channel paths, the cryptographic coprocessors allocation to logical partitions is recorded into Candidate and Online lists. There is a coprocessor Online List and Candidate List that have to be specified in the image profile of partitions using the CEX2 coprocessors (contrary to the channel path candidate list in the IOCDS the partitions are not assigned to the resource, instead the resources are assigned to the partitions).

The Cryptographic Candidate List identifies the CEX2 coprocessors that are eligible to be accessed by this logical partition.
Note that when the partition is activated, an error condition is not reported if a coprocessor in the Candidate list is not installed in the system. Selecting a coprocessor that is not installed prepares the settings in the active partition for a future nondisruptive install of the coprocessor.

The Cryptographic Online List identifies the CEX2 coprocessors that are automatically brought online during logical partition activation. The coprocessors in the Online List must also be selected in the Candidate List.

PR/SM therefore allow logical partitions to access only the coprocessors that are listed in the Cryptographic Candidate List, using the domain specified as Usage Domain in the logical partitions image profiles.

4.9 More on PR/SM security - The certification proof points

IBM has been proceeding for many years with Security certification of the PR/SM hardware and firmware implementation for every new mainframe models. As of the writing of this book System z model z10 EC has been evaluated as meeting the Common Criteria (ISO 15408) EAL5 level of certification, using a specific protection profile.

The public version of the Security Target for this evaluation can be found on the BSI (Bundesamt für Sicherheit in der Informationstechnik") internet web site: www.bsi.bund.de

The IBM documentation "System z10 Processor Resource/Systems Manager Planning Guide" - SB10-7153-02 dedicates an appendix to "developing, building and delivering a certified system", which provides an information on what is covered by this certification and how to configure a system so that the PR/SM setup meets the certification criteria.

The objective of the evaluation

The objective of the evaluation was to assess whether the PR/SM implementation provides the Security Administrator the ability to define a completely secure system configuration. That is a system defined in such a manner that total separation of the logical partitions is achieved thereby preventing a partition from gaining any knowledge of another partition's operation.

Note that only functions related to logical partition isolation, physical resource allocation, access control and audit were the subject of this evaluation. Additional functions provided by PR/SM related to normal operations and maintenance of the system were not considered as security enforcing functions as the system will be configured to provide a configuration consistent with secure isolation such that these operations cannot be in conflict with the security policy of PR/SM.

The security objectives that were the target of the evaluation can be read in the "Common Criteria for Information Technology Security Evaluation - Public Version of the Security Target for PR/SM for the IBM System z10 EC" - This document is available at https://www.bsi.bund.de/cln_134/ContentBSI/english/topics/Certification/CertificationReports/reports.html#doc471312bodyText9

Isolated Logical Partition

The PR/SM security evaluation introduces the concept of Isolated Logical Partitions, that is logical partitions that can be configured so that one partition cannot gain knowledge about any other partition's available I/O resources or performed operations. Isolated Logical Partitions were the target of the Common Criteria certification.

Conversely, a non-Isolated Logical partition has explicitly or implicitly some level of sharing with other logical partitions that results, or may result, in communications flowing between the two partitions. A single physical System z allows any combination of Isolated and non-Isolated logical partitions to be configured. The non-Isolated logical partitions can be configured in any manner supported by System z. Any level of sharing or cooperation among the non-Isolated logical partitions (e.g. Parallel Sysplex®) is permitted and will not have any impact on the Isolated logical partitions.

The following System z functions are not covered by the Common Criteria certification as an inappropriate use of them may yield potentially non-isolated logical partitions:

- Dynamic I/O Configuration
- Dynamic CHPID management
- Reconfigurable channel paths (CHPIDs)
- I/O resource sharing using Multiple Image Facility (MIF)
- Intelligent Resource Director (IRD)
- Interconnection of logical partitions via HiperSockets or shared OSA port

Setting up Isolated Logical Partitions in the system

The following conditions should be met to qualify for the status of Isolated Logical Partition and establish a certified environment:

- The hardware and any networks used to connect the hardware must be physically secure. Access to I/O devices must be restricted to authorized personnel. The Hardware Management Console must be physically protected from access other than by authorized system administrators.
- The remote support facility must be disabled. Devices must be configured so that no device is accessible by any partition other than the partition to be isolated (although they may be accessible by more than one channel path).
- Each I/O (physical) control unit must be allocated only to an Isolated logical partition in the current configuration.
- The Security Administrator must not reconfigure a channel path owned by an Isolated partition unless all attached devices and control units are attached to that path only.
- The Security Administrator should ensure that all devices and control units on a reconfigurable path owned by an Isolated partition are reset before the path is allocated to another partition.
- No channel paths may be shared between an Isolated partition and any other partition(s).
- Although the system will help ensure that the total number of dedicated and shared processors are not over allocated, the System Administrator must ensure that the number of processors dedicated to activated partitions is less than the total number available. This is important so that some processors are available for partitions that do not have dedicated access.
- Dynamic I/O Configuration changes must be disabled.
- If I/O Priority Queuing is enabled for the system an Isolated partition's minimum and maximum I/O Priority values must be equal.
- For Isolated partitions, Workload Manager must be disabled so that CPU and I/O resources are not managed across partitions.
- An Isolated partition must not be configured to enable HiperSockets (Internal Queued Direct I/O).
- Partitions must be prevented from receiving performance data from resources that are not allocated to them (Global Performance Data Control Authority must be disabled).At most one partition can have I/O Configuration Control Authority (i.e. No more than one partition must be able to update any IOCDS.) and this partition must be administered by a trustworthy administrator (i.e. the administrator of this partition is considered to be the Security Administrator). I/O Configuration Control should be enabled for a single, specific logical partition only during the short period of time when it is permitted to write a new IOCDS.
- The Security Administrator must ensure that write access is disabled for each IOCDS, unless that IOCDS is to be updated (the current IOCDS must not be updated).The Security Administrator must verify any changed IOCDS after a power-on reset with that

IOCDS, before any partitions have been activated (the Security Administrator may determine whether the IOCDS has been changed by inspecting the date of the IOCDS).

- No partition should have Cross-partition Control Authority (i.e. No partition should be able to reset or deactivate another partition).
- No Isolated partition may have coupling facility channels which would allow communication to a Coupling Facility partition.
- The “Use dynamically changed address” and “Use dynamically changed parameter” check boxes must not be selected in the Image or Load profile.
- No Isolated partition should have the following Counter Facility Security Options enabled:
  - Crypto activity counter set authorization control
  - Coprocessor group counter sets authorization control

Disabling these options will ensure that its crypto and coprocessor activities are not visible to any other partitions.

-

5

z/VM Security

This chapter discusses the security aspects of z/VM facilities and introduces how z/VM virtualization can provide a secure isolation between guests on System z. The chapter also discusses Resource Access Control Facility(RACF) and Ligthweight Directory Access Protocol (LDAP) on z/VM and how you can use them to allow an enterprise wide point of control.

z/VM has the System z hardware resources virtualization architecture built in the software through the concept of the virtual machine. z/VM Virtual Machine is also referred to as user IDs, guests or hosts.

z/VM can host multiple operating systems as guests of the virtual machines by emulating the System z hardware within that same hardware, while providing extra features and benefits that are implemented in software in a more cost effectively way.

Virtualization of the machine enables you to do the following:

- Manage many servers using universal management tools
- Reduce management costs
- Maximize the utilization of existing hardware

Like all System z operating systems, z/VM can run alone or share the mainframe with others by using the LPAR capabilities of System z. z/VM can either virtualize the same System z server it is running on, or emulate hardware features that are not necessarily installed on that particular model of server, and it can provide a customized environment for each guest.

z/VM provides System z features to the guest operating systems transparently and simultaneously, without the need of having a physical server per guest. At the same time, z/VM can isolate each guest operating system and handle access to real devices as needed.

The Start Interpretive Execution Instruction (SIE) available on the System z can create the environment for Virtual Machines (user IDs). Most of the of instructions are executed by the hardware with little z/VM intervention.

In addition to SIE, some of the newer System z provide I/O assist functions to QUEUE DIRECT I/O(QDIO), used by System z adapters such as Networking and Fibre Channel

Protocol (FCP) to assist reducing the amount of interrupts passed to the z/VM Control Program.

The number of guests that z/VM can operate concurrently is limited only by the amount of resource available to the System z. This is in contrast to LPARs which have a fixed limit to the number of LPARs, dependent on the System z used.

Guests of a z/VM host run within user IDs, or just “ID”, for short. Typically, people logging on to z/VM are called “users”, whereas automated user IDs are known as “service machines” and run in a disconnected state, that means without a console or display terminal attached.

5.1 z/VM and the IBM Security Blueprint

The IBM Security Blueprint uses a product-agnostic and solution-agnostic approach to categorize and define security capabilities and services that are required to answer business security requirements or areas of concern categorized by the IBM Security Framework. It also defines a common vocabulary to use in further discussions.

In the blueprint, IBM aims to identify architectural principles, that are valid across all domains, and fundamental services within and across the domains. It also highlights applicable best practices and IT standards.

This part will describe which z/VM areas are related to the IBM Security Blueprint on the Security Services and Infrastructure layer, as highligthed on Figure 5-1.

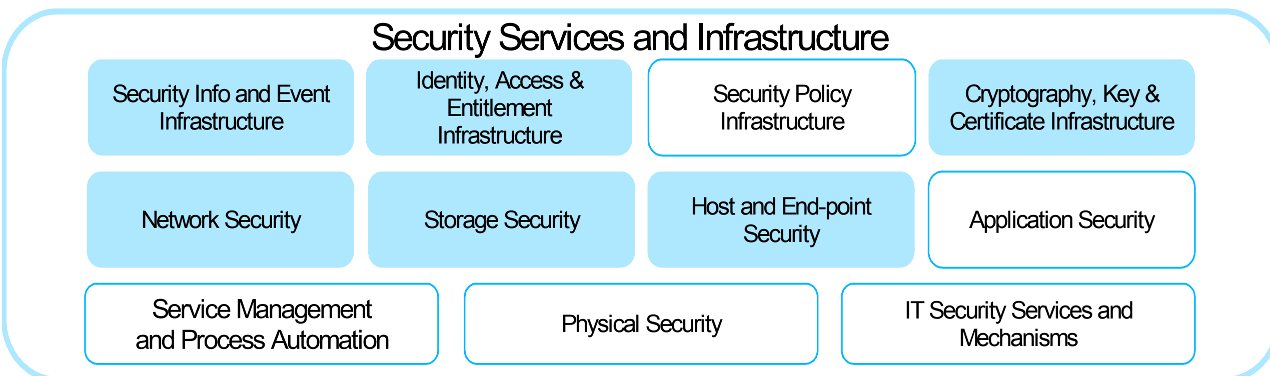


Figure 5-1 IBM Security Blueprint disciplines on z/VM

Security Information and Event Management Infrastructure

On z/VM, Security Information and Event Management disciplines are provided by components described under following headings:

- *"Intrusion detection"*

  The Intrusion detection creates journal to identify attempts to logon to a virtual machine or link to minidisk using invalid password

- *"Accountability"*

  z/VM *Logon By* feature creates an audit trail of the actual user to ensure user authority is validated and to provide accountability.

- *"The Resource Access Control Facility"*

  RACF on z/VM can be used for logging and reporting that allow resource owner to identify who attempted to access the resource.

Identity, Access and Entitlement Infrastructure

On z/VM, Identity, Access and Entitlement Infrastructure disciplines are provided by components described under following headings:

- *"The Resource Access Control Facility"*

  RACF on z/VM can be used to verify the user and control access to protected resources, like minidisks, terminals and shared file system (SFS) files and directories

- *"LDAP"*

LDAP can be used to have information stored about users, applications, files and other resources in directory format accessible from network and being able to synchronize information on other directories.

Security Policy Infrastructure

On z/VM, Security Policy Infrastructure discipline has no role

Cryptography, Key and Certificate Infrastructure

On z/VM, Cryptography, Key and Certificate Infrastructure disciplines are provided by components described under following heading:

- *"System z cryptographic solution"*

  z/VM does not provide a direct interface to cryptographic hardware, nor does it require cryptographic services itself. z/VM provides means to share hardware cryptographic services among the hosted guests.

Network Security

On z/VM, Network Security discipline is provided by components described under following heading:

- *"Secure communication to the z/VM System using SSL"*

  On z/VM the TCP/IP connections to z/VM guests can be protected by Secure Socket Layer (SSL) to avoid sending passwords and other confidential information in clear text.

- *"z/VM virtual networking"*

  z/VM virtual networking has highly secure communication paths using z/VM TCP/IP stack, Guest Lan and support for VLAN and Virtual; Switch.

Storage Security

On z/VM, Storage Security discipline is provided by components described under following headings:

- *"z/VM security features"*

  z/VM Control Program uses disk storage partitioning, known as *minidisks*, to share one physical disk among several guests.

- *"z/VM system integrity definition"*

  z/VM protects auxiliary storage disk extents implementing isolation for minidisks and virtual disks. Minidisks can be also password protected.

- *"The Resource Access Control Facility"*

  RACF on z/VM can be used to control access to protected resources, like minidisks and shared file system (SFS) files and directories

Host and Endpoint Security

On z/VM, Host and Endpoint Security disciplines are provided by components described under following headings:

- *"Introduction to z/VM virtualization"*

  z/VM has System z hardware resources virtualization architecture built into the software and provides System z features to the guest operating systems transparently without having a physical server per guest.

- *"z/VM security features"*

  The core of the z/VM is the *Control Program* which creates and maintains virtual environments for virtual machines (guests).

- *"The SIE instruction"*

  z/VM does not allow guest operating systems to be aware of each other by using the *Interpretative Execution Facility* (IEF) which is built in specially for this purpose. IEF executes an entire virtual machine instruction stream as a single instruction called SIE.

- *"z/VM system integrity definition"*

  The z/VM control program system integrity is the inability of any program running in a virtual machine circumvent or disable the control program real or auxiliary storage protection or obtain control in real supervisor state

- *"DirMaint system integrity"*

  DirMaint™ uses standard z/VM facilities to isolate users from each other and from the system and protect files from outside interference or contamination.

- *"Compliance to policy"*

  A z/VM system is secured using security features of the System z hardware by maintaining compliance to security policy within operating practices, thus making use of the *user directory* that contains a list of users of the system. In the z/VM system of privilege, a user either can have no privileges or can be assigned to one or more privilege classes. Each privilege class represents a subset of Control Program commands that the system permits the user to enter.

Application Security

On z/VM, Application Security discipline has no role

Service Management and Process Automation

On z/VM, Service Management and Process Automation disciplines has no role

Physical Security

On z/VM, Physical Security discipline has no role

IT Security Services and Mechanisms

On z/VM, IT Security Services and Mechanisms disciplines has no role

5.2 Introduction to z/VM virtualization

z/VM has the System z hardware resources virtualization architecture built into the software through the concept of the *virtual machine*. z/VM virtual machine is also referred to as *user IDs*, *guests*, or *hosts*. z/VM can host multiple operating systems as guests of the virtual machines by emulating the System z hardware within that same hardware, while providing extra features and benefits that are implemented in software in a more cost effective way.

Virtualization of the machine enables you to:

- Manage many servers using universal management tools
- Reduce management costs
- Maximize the utilization of existing hardware

Like all System z operating systems, z/VM can run alone or share the mainframe with z/VM, Linux for System z or z/OS using the LPAR capabilities of System z. z/VM can either virtualize the same System z server on which it is running or can emulate hardware features that are not necessarily installed on that particular model of server. It can also provide a customized environment for each guest.

z/VM provides System z features to the guest operating systems transparently and simultaneously, without having a physical server per guest. At the same time, z/VM can isolate each guest operating system and handle access to real devices as needed.

The *Start Interpretive Execution Instruction* (SIE) that is available on System z can create the environment for virtual machines (user IDs). Most of the of instructions are executed by the hardware, with little z/VM intervention. The SIE is implemented on the System z by the *Interpretive Execution Facility*. The z/VM Control Program gets an interruption when the hardware detects conditions such as Timer expiration, unassisted Input Output operation (I/O), instructions that require some privileges, and Program Interrupts. SIE also handles the use of region, segment, and page tables that were set up previously by the z/VM Control Program for the user ID. SIE instruction is available only on System z, and it is exploited by z/VM to decrease Control Program overhead.

In addition to SIE, some of the newer System z provide I/O assist functions to QUEUE DIRECT I/O (QDIO) that is used by System z adapters such as Networking and Fibre Channel Protocol (FCP) to reduce the amount of interrupts that are passed to the z/VM Control Program. Resources that can be shared between guests are CPU, real memory, disk volumes, network adapters, printers, and devices specific to System z such as cryptographic cards. It also supports the latest Storage Area Network (SAN) and virtual tape library systems. For example, in the case of disk storage, z/VM is capable of partitioning a disk volume and assigning portions to each guest. Control of read-only and read-write access, or none at all, is at the discretion of the z/VM administrator.

The number of guests that z/VM can operate concurrently is limited only by the amount of resource that is available to the System z. This is in contrast to LPARs which have a fixed limit to the number of LPARs that are dependent on the System z used.

Guests of a z/VM host run within user IDs, or just *ID* for short. Typically, people logging on to z/VM are called *users*, whereas automated user IDs are known as *service machines* and run in a disconnected state (which means without a console or display terminal attached).

z/VM user definition and part of z/VM security is accomplished by the z/VM *directory*. The directory is encrypted on the DASD to avoid hacking its contents. To allow multiple users to change and maintain the directory, your installation can use a directory Control Program such as *Directory Maintenance Facility* (DirMaint). You can configure DirMaint to allow specific levels of change control as needed to specifics administrators and to the users.

z/VM isolation between users and security can be enhanced using RACF. RACF allows a deep and more flexible security implementation than z/VM controls. You can configure RACF to control z/VM commands and resources and to define who can access these resources and at what level of authority.

5.3 z/VM security features

z/VM is today's version of a hypervisor operating system. Virtual machine design began in the early 1960s, when IBM was exploring how to meet customer expectations using virtualization. The development of virtual machine was closely tied to the development of virtual storage, because they needed to operate together. The core of z/VM (that is, the hypervisor), is actually the *Control Program*. The Control Program creates and maintains virtual environments for virtual machines (guests).

Figure 5-2 represents a z/VM system with multiple guests.

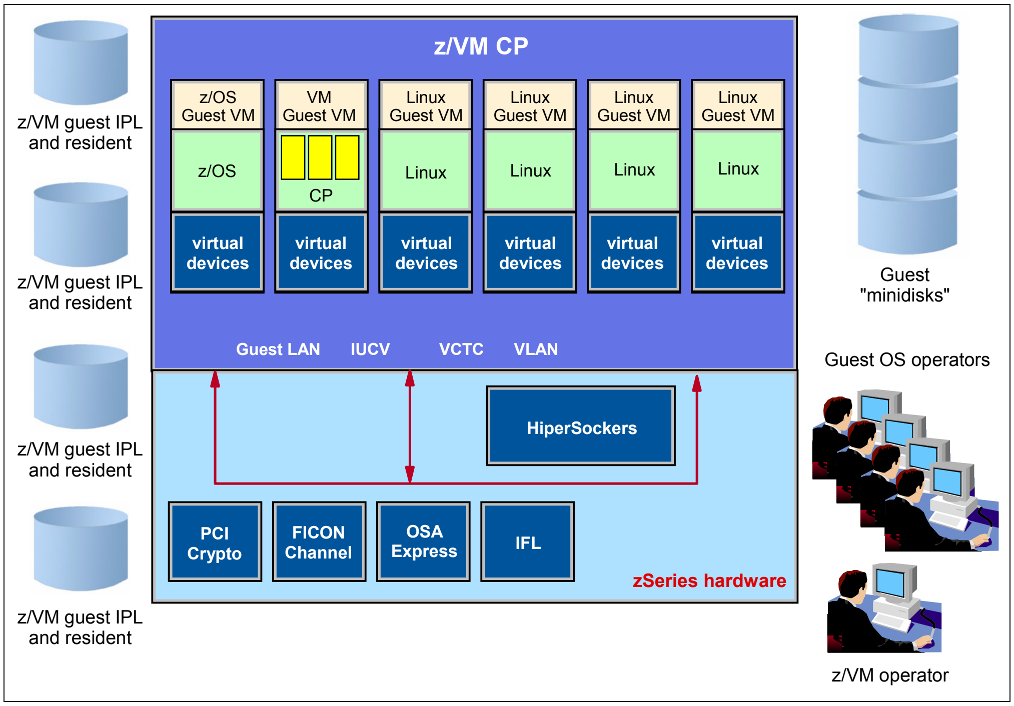


Figure 5-2 z/VM implementation with multiple guests

Note the following points:

- Only the Control Program is IPLed using the actual hardware IPL sequence. The guest IPL sequence is simulated by the Control Program.
- The Control Program operator console is actually providing Control Program emulated *hardware consoles* for the guest virtual environment. Control Program commands, on top of the guest IPL command, are providing, for example, the equivalent of a *System reset* or *Restart* hardware functions for the guest machines.
- The operating systems running in each guest have their usual operator console, which are physically connected through I/O channels to their respective operating systems.
- z/VM has its own scheme of disk storage partitioning, known as *minidisks*, that the Control Program uses to share one physical disk among several guests.
- z/VM can itself be running in a guest virtual machine, thus creating *second level* guest z/VMs.
- The security requirements, as seen from the z/VM software perspective, ensure that the guest z/VMs have access only to the physical resources to which they are entitled and to guarantee inter-guest isolation. Each guest operating system is then responsible for its security environment as seen by its own users.

Note: These Control Program and guest software layers created in the z/VM exploit the z/Architecture and use the physical CPUs by switching instruction flows and address spaces between Control Program, guests operating systems, and user programs. They all exploit the same System z instruction set, with the exception of Control Program. Control Program uses a very specific instruction that is designed for virtual machine use called the *Start Interpretive Execution* (SIE).

5.3.1 The SIE instruction

By default, z/VM does not allow guest operating systems to be aware of each other. It achieves this by using the *Interpretive Execution Facility* (IEF) of the System z that is built in specifically for this purpose.

IEF executes an entire virtual machine instruction stream as a single instruction called SIE. The virtual machine is dispatched to run by the Control Program in a way that makes the System z firmware aware of the virtual machine details. The guest runs on the hardware until its time slice expires or until an instruction that cannot be virtualized is attempted, such as reference to a real address. At that point, the Control Program regains control to simulate the operation. Should a guest operating system fail, control is always returned to the Control Program for error recovery. In this way, guests are isolated and protected from others. This results in very low overhead for a virtualized environment and delivers confidentiality and integrity among virtual servers.

This feature is described in greater detail in *z/VM Security and Integrity,* GM13-0145.

The operating system running as a guest can perform its own address translation and maintain its own tables of *real* and *virtual* memory without awareness that its entire address space is virtual. What appears to be page or swap space to an operating system can be, in reality, real memory. Performance improvements can be realized in this way with certain operating systems. Conversely, a poorly behaving application will be paged out by z/VM and will not drag down the entire system. In such a protected environment, a guest operating system that is running on z/VM can communicate over a virtual network with another guest of the same mainframe, or with another server externally, and not notice the difference.

Clearly, not all devices can be shared simultaneously as can real memory, network interfaces, and direct access disks. Tape drives, for example do not lend themselves to being used by multiple programs concurrently. Thus, serial devices are attached to a guest for the duration of use, while random access devices are controlled by the Control Program and instructions for each guest interleaved in an efficient manner. Printers and other unit record devices are, in general, owned by the Control Program and the spooling subsystem controls which output set is printing at any given time. In certain circumstances, a serial device is dedicated to a guest with the **attach** command and is unavailable to other guests during that session.

5.3.2 z/VM system integrity definition

The z/VM control program system integrity is the inability of any program running in a virtual machine not authorized by z/VM control program mechanism under the customer's control or a guest operating system mechanism under the customer's control to:

- Circumvent or disable the control program real or auxiliary storage protection.
- Access a resource protected by an external security manager (ESM), such as RACF. Protected resources include virtual machines, minidisks and terminals.
- Access a control program password-protected resource.
- Obtain control in real supervisor state or with privilege class authority or directory capabilities greater than those it was assigned.
- Circumvent the system integrity of any guest operating system that itself has system integrity as the result of an operation by any z/VM control program facility.

Real storage protection refers to the isolation of one virtual machine from another. CP accomplishes this by hardware dynamic address translation, start interpretive-execution guest storage extent limitation and the Set Address Limit facility.

Auxiliary storage protection refers to the disk extent isolation implemented for minidisks and virtual disks through channel program translation.

Password-protected resource refers to a resources protected by CP logon passwords and minidisk passwords.

Guest operating system refers to a control program that operates under the z/VM control program.

Directory capabilities refers to those directory options that control functions intended to be restricted by specific assignment, such as those that permit system integrity controls to be bypassed or those not intended to be generally granted to users.

5.3.3 DirMaint system integrity

DirMaint uses standard z/VM system facilities to:

- Protect the DirMaint service machines (DIRMAINT, DATAMOVE, DIRMSATs) from subversion
- Protect files from outside interference or contamination.
- Isolate users from each other and from the system.
- Exploit hardware protection mechanisms.
- Identify the originating user ID (and node ID), for all incoming requests.
- Record auditable information.

The DIRMAINT and DIRMSAT service machines require the appropriate CP privilege class to use CP commands and DIAGNOSE codes. These machines benefit from use of the OPTION D84NOPAS directory statement, and security is enhanced with the D8ONECMD FAIL LOCK directory statement. Data integrity is enhanced when the optional DATAMOVE service machines have LNKSTABL and LNKEXCLU specified on the OPTION statement in their directory entries.

5.3.4 Intrusion detection

One element of z/VM intrusion detection capabilities is that if a login is denied, the event is tracked and a security journal entry made when the number of denials exceeds an installation-defined maximum. When a second maximum is reached, log on to the user ID is disabled, an operator message is issued, and the terminal session is terminated.

Journaling is supported on z/VM. Virtual machine logon attempts and linking to other virtual machine’s minidisks are detected and recorded. Using the recorded information, you can identify attempts to log on to a virtual machine or to link to minidisks using invalid passwords.

5.3.5 Accountability

A special capability available with z/VM is *Logon By*. When users log on to the shared user ID using this option, they provide their own user ID and password. An audit trail is maintained of who is actually logged into a shared user ID, so the problems inherent in sharing passwords are avoided. This audit trail tracks the identity of the user of a shared user ID, ensures user authority is validated, and provides *accountability*.

5.3.6 Compliance to policy

A z/VM system is secured using security features of the System z hardware by maintaining compliance to security policy within operating practices, thus making use of the *user directory* that contains a list of users of the system. The main control point of z/VM security is the user directory file, called *USER DIRECT*. For that reason, it is also called the *Control Program directory*. This directory file is owned by the system administrator, and every user of the system is identified within this file, including the system administrator. Also, each user's resources are defined in the file. The directory and the ability to promote a directory into active state must be the most protected assets of a z/VM system. The system administrator must lead the way in following security standards and guidelines if the community is to be safe. When DirMaint is installed, it owns the directory, and controls each part of the directory that can be changed by a given user in the system.

When Resource Access Control Facility feature for z/VM (RACF) is installed, it can be configured to control functions normally being checked in the directory for authorization. At a minimum RACF controls the password field but can be used to control minidisk access, spool files, and commands privileges.

In z/VM, a subject is a virtual machine, which is one of four types:

- General user
- Privileged user
- Trusted server
- System operator

Each role has approximately the same logical structure. A general user is defined as a virtual machine that has, at the most, the set of the Control Program commands available in the IBM-defined privilege class G.

In addition, the general user does not have the following characteristics:

- SPECIAL
- Group-SPECIAL
- CLAUTH
- Group-CLAUTH
- OPERATIONS authority to RACF
- No OBEY authority for VM TCP/IP
- No access to the z/VM directory (source or object forms)
- No read-write access to the PARM disk(s), or other system areas of CP-owned volumes
- No read-write access to the source or object code of CP, CMS, RACF, or z/VM TCP/IP
- No read-write access to the RACF database
- No read-write access to the RACF audit trail

z/VM privileges

In the z/VM system of privilege, a user either can have no privileges or can be assigned to one or more privilege classes. Each privilege class represents a subset of Control Program commands that the system permits the user to enter. Each privilege class, sometimes called *CP privilege class*, is defined around a particular job or set of tasks, thereby creating an area outside of which the user cannot go. Of course, it is commonplace for a user to be assigned to more than one CP privilege class. Users are unable to enter commands in privilege classes to which they are not assigned.

> **Note:** Any user, except those with either NO PRIVILEGE or CP privilege class G, is considered part of the configuration but is not necessarily considered trusted.

Here is a summary of CP privilege classes, their associated users, tasks, and security implications:

- **Privilege class A**

  The primary system operator. The system operator is among the most powerful and privileged of all z/VM users. The system operator is responsible for the system's availability and its resources. The system operator also controls accounting, broadcasts messages, and sets performance parameters.

- **Privilege class B**

  The system resource operator. The system resource operator controls the allocation and de-allocation of real resources, such as memory, printers, and DASD. Note that the system resource operator does not control any resource already controlled by the system operator or the spooling operator.

- **Privilege class C**

  System programmer. The system programmer updates the functions of the z/VM system and can change real storage in the real machine.

- **Privilege class D**

  Spooling operator. The spooling operator controls spool files and real unit record devices, such as punches, readers, and printers.

- **Privilege class E**

  System analyst. The system analyst has access to real storage and examines dumps to make sure that the system is performing as efficiently and correctly as possible.

- **Privilege class F**

  IBM service representative. The representative of IBM who diagnoses and solves problems by examining and accessing real input and output devices and the data they handle.

- **Privilege class G**

  General user. This is the most prevalent and innocuous of the CP privilege classes. The commands that privilege class G users can enter effect only their own virtual machines.

- **Privilege class ANY**

  The commands in this privilege class are available to any user.

Privilege classes A, B, C, D, E, and F require individuals worthy of significant trust and whose activities require careful auditing. For example, users with privilege class B or C can modify an installation's system of CP privilege. Because this modification violates the CAPP security policy, system programmers and similarly privileged users must be trusted to not tamper with the system of CP privilege (and auditing must confirm this trust).

As another example, privilege class C users can enter the `cp store host` command that allows them to alter real storage. This command makes it possible for users to negate the CAPP classification.

Privilege class G users have no influence outside their own virtual machines. So, with the exception of access to storage objects, they have very little security relevance.

The ANY privilege class commands cannot violate the security policies of the system. This is because all commands in the ANY privilege class are auditable and subject to either Discretionary Access Control (DAC) or Mandatory Access Control (MAC). Therefore, class ANY users, together with class G users, cannot violate the security policy. In the Control Program, each level of privilege is discrete and not predicated on others. Furthermore, each privilege class (a subset of commands) is related to one or more function types (subsets of users).

The Control Program directory

The Control Program directory is the reference repository which z/VM uses to perform its access control. By default, each z/VM user's address space, DASD, vswitch, and all files are private to the user or virtual server.

Special action is required to expose data to another user, although, as with all platforms, the OPERATOR or superuser is able to gain all access rights to them if they have a need. In this directory, each guest's *privilege class* is assigned.

The privilege class determines their rights to issue certain CP commands and program instructions (diagnose codes) that reference the world outside their own virtual machine. Although a default set of classes is defined when you initially install z/VM, you can also create your own custom classes. You can define up to 32 classes. The standard class for general users is class G. Class G users cannot affect other users or CP operation, although by using certain query commands, they can become aware of other guests.

It is possible, even desirable, to create classes with less than general user privilege which allow certain virtual machines only the minimum functionality required to perform their assigned duty.

The CP directory has basically two forms:

- Human readable file called USER DIRECT.
- Machine readable in object form, placed on a reserved area of disk by the privileged CP **directxa** utility command.

In z/VM, access rights can be granted in two ways:

- Using the privilege class.
- Granting the access to resources by the owner or by another authorized user. For example, you can grant a user read or write access to virtual disks that are owned by a different user. The resource's owner or a system administrator can grant the permission.

System user IDs involved in security

There are some users that are defined as part of the installation process. The standard user IDs for the default system installation are:

- OPERATOR: System operator and high privilege user ID. Equal to root on UNIX systems.
- MAINT: Used by a person for system administration and maintenance. Similar to OPERATOR from an authorization point of view.
- EREP: EREP stands for *Environmental Record Editing and Printing Program*. Hardware anomaly detection and predictive failure system.
- DISKACNT: Records events such as logon and logoff.
- OPERSYMP: Used to retrieve symptom records. System dump analyzer and problem tracking system.

Conversational Monitor System

VM consists of the Control Program (CP), which is the virtual machine hypervisor that interfaces between real devices and the guest, and the Conversational Monitor System (CMS) which is a single-user operating system for use within a CP virtual machine.

CMS provides a text-based environment much like PC-DoS. Using CMS, you can create and edit files, and build applications to automate routine tasks. Many programming and scripting languages are supported under CMS. CMS does not recognize the concept of a user ID; instead, that distinction belongs to CP.

Within CP you have your very own mainframe to operate, and you can use it to create files that contain data or programs, share them with other CMS users, compile them, execute them, or send them to other virtual machines running other operating systems. But in order to perform those tasks, you must first LOGON to CP with a user ID.

CMS is often loaded into a CP virtual machine first in order to prepare the virtual machine for loading a second, batch-oriented, operating system.

5.3.7 Secure communication to the z/VM System using SSL

Depending on the security policies in an enterprise, and depending on the network environment, clients might want to secure (encrypt) the communication for their connections to the z/VM guest user IDs to avoid sending passwords in clear text over a network and to protect the content of the communication.

With z/VM you can setup the TCP/IP connections to the z/VM guests to be protected by Secure Socket Layer (SSL).

Like in other areas, the implementation of SSL is done in z/VM by the usage of a virtual server to handle the work. The SSL server, which runs in the SSLSERV virtual machine, provides processing support for encrypted communication between remote clients and z/VM TCP/IP server that listen on secure ports. The SSL server manages the database in which the server authentication certificates are stored. The TCP/IP stack server routes requests for secure ports to the SSL server. The SSL server, representing the requested application server, participates in the handshake with the client in which the cryptographic parameters are established for the session. The SSL server then handles all the encryption and decryption of data.

Figure 5-3 illustrates the principal setup and the information flow.

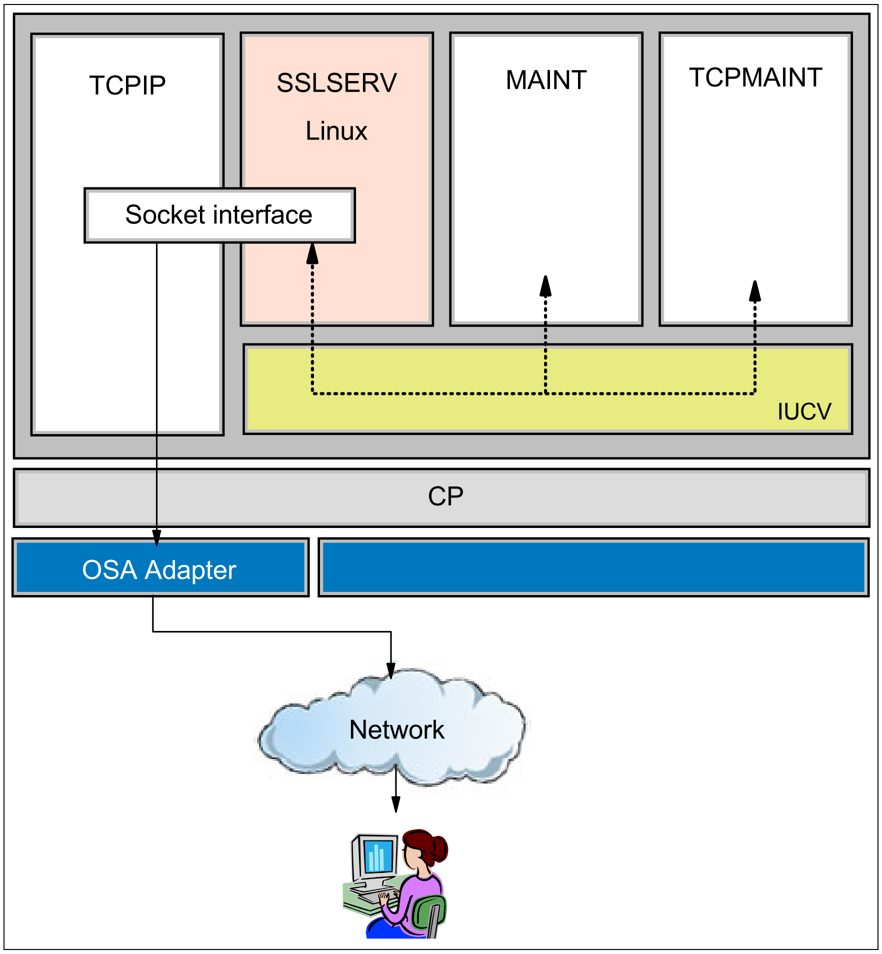


Figure 5-3 SSL implementation architecture in z/VM

The SSL Server is a virtual service machine, with a special Linux server installed and configured for exclusive use of SSL. Only specific Linux distributions and kernel levels are supported. For a list of supported Linux distributions and kernel versions, and also for detailed setup instructions, see the z/VM Web page:

http://www.vm.ibm.com/related/tcpip/

On this page you can find a link to SSL Server Configuration where all related information is available:

http://www.vm.ibm.com/related/tcpip/vmsslinf.html

Verify whether there are important service updates that might contain new information such as necessary PTFs or APARs. Check this information before you begin your implementation.

During the SSL handshake the client and the SSL server negotiate which encryption algorithms will be used to secure the connection. A cipher suite is selected by them which is common for both parties. Using the **ssladmin query status** command in the TCP/IP server, you can verify which cipher suites are allowed to be used by the SSL connection. Note that not all cipher suites provide a high degree of security. We recommend that you carefully consider which suites to allow. You might want to exempt individual cipher suites, such as NULL, NULL_SHA, or NULL_MD5, or you might want to instruct the SSL server to operate in Federal Information Processing (FIPS) mode.

The SSL support of z/VM is an easy method to protect the communication to the z/VM server especially for administration tasks. This increases the total security and protection of the z/VM system, as sensitive information such as passwords from administrators are protected independent from the network.

Enhanced SSL server on z/VM V5R4

With the PTF for APAR PK65850 (for additional required service, see the VM540 subset of the TCPIP540 PSP bucket), z/VM V5.4 provides an SSL server which runs in a CMS environment, replacing the Linux-based SSL server provided in previous z/VM releases. This change removes the requirement of using a Linux distribution to host the SSL server. The CMS-based SSL server might enable encryption services to be deployed more quickly and can help make installation, service, and release-to-release migration simpler. Other enhancements to the z/VM SSL server include:

- Network-free SSL server administration

  The SSL server can be managed without requiring a network connection between the SSL server administrator and the SSL server.

- New encryption and decryption engine

  The SSL server uses z/OS V1.10 System SSL technology for encryption, decryption, and certificate management.

- New certificate-management services

  The System SSL **gskkyman** utility is now used to manage the SSL server certificate database. New services available for the SSL server include certificate renewal, certificate signing, and certificate exportation with or without the private key. The gskkyman application also manages certificates for the z/VM LDAP server.

5.4 Additional features

This section introduces briefly that additional feature and products that are available in the z/VM system and that are used to build the security solution.

5.4.1 The Resource Access Control Facility

The Resource Access Control Facility (RACF) licensed program can satisfy the preferences of the user without compromising any of the concerns raised by security personnel. The RACF approach to data security is to provide an access control mechanism that offers effective user verification, resource authorization, and logging capabilities. RACF supports the concept of *user accountability*. It is flexible, has little noticeable effect on the majority of end users, and little or no impact on an installation's current operation.

RACF controls access to and protects resources on both multiple virtual storage (z/OS) and virtual machine systems. For a software access control mechanism to work effectively, it must be able to first identify the person who is trying to gain access to the system, and then verify that the user is really that person.

With RACF, you are responsible for protecting the system resources, such as minidisks, terminals, and shared file system (SFS) files and directories, and for issuing the authorities by which those resources are made available to users. RACF records your assignments in *profiles* stored in the RACF database. RACF then refers to the information in the profiles to decide if a user should be permitted to access a system resource.

The ability to log information, such as attempted accesses to a resource, and to generate reports containing that information can prove useful to a resource owner, and is very important to a smoothly functioning security system. Because RACF can identify and verify a user's user ID and recognize which resources the user can access, RACF can record the events where user-resource interaction has been attempted. This function records actual access activities or variances from the expected use of the system.

RACF has a number of logging and reporting functions that allow a resource owner to identify users who attempt to access the resource. In addition, you or your auditor can use these functions to log all detected successful and unsuccessful attempts to access the RACF database and RACF-protected resources. Logging all access attempts allows you to detect possible security exposures or threats. The logging and reporting functions are:

- Logging: RACF writes audit records in a file for detected, unauthorized attempts to enter the system. Optionally, RACF can also writes records for authorized attempts or detected, unauthorized attempts to:
  - Access RACF-protected resources
  - Issue RACF commands
  - Modify profiles on the RACF database.
- Sending Messages: RACF sends messages to the security console for detected, unauthorized attempts to enter the system and for detected, unauthorized attempts to access RACF-protected resources or modify profiles on the RACF database.
- Keeping Statistical Information: Optionally, RACF can keep selected statistical information, such as the date, time, and number of times that a user enters the system and the number of times a single user accesses a specific resource. This information can help the installation analyze and control its computer operations more effectively. In addition, to allow the installation to track and maintain control over its users and resources, RACF provides commands that enable the installation to list the contents of the profiles in the RACF database.

Some features introduced with z/VM Version 5.3 and RACF Feature Level 5.3 include:

- Mixed-case 8-character passwords
- Mixed-case password phrases up to 100 characters, including blanks
- No longer possible to reset password to default group name
- Audit trail can be unloaded in XML format
- Remote authorization and audit through z/VM new LDAP server and utilities

Note: For references on security implementation on z/VM, including RACF, refer to *z/VM Secure Configuration Guide,*SC24-6138.

5.4.2 LDAP

Today, people and businesses rely on networked computer systems to support distributed applications. These distributed applications might interact with computers on the same local area network, within a corporate intranet, within extraneous linking up partners and suppliers, or anywhere on the worldwide Internet. To improve functionality and ease-of-use and to enable cost-effective administration of distributed applications, information about the services, resources, users, and other objects accessible from the applications needs to be organized in a clear and consistent manner. Much of this information can be shared among many applications, but it must also be protected in order to prevent unauthorized modification or the disclosure of private information.

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

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

z/VM V5.3 introduces a z/VM LDAP server and client. It is a subcomponent of TCP/IP. Thez/VM LDAP server has been adapted from the IBM Tivoli® Directory Server for z/OS (delivered in z/OS V1.8).

z/VM LDAP can help to simplify administration tasks when a Linux farm is installed on z/VM.The z/VM LDAP server helps administrators:

- Improve Linux logon security using RACF services to validate user and password.
- Reduce repetitive tasks, such as defining the same Linux user in multiple Linux images.
- Implement the RACF database sharing coupled with a z/VM LDAP server using native authentication permits to have a single point to administrate multiple z/VM and Linux servers.
- Gives a better business continuity solution when using multiple mainframes in conjunction with the hiperswap function.

LDAP upgrade and RACF password change logging

In order to help maintain cross-platform consistency, the z/VM LDAP server introduced in z/VM V5.3 has been upgraded to the function level of the z/OS V1.10 IBM Tivoli Directory Server for z/OS. The z/VM RACF Security Server feature has been enhanced to create LDAP change log entries in response to updates to RACF group and user profiles, including changes to user passwords and password phrases. This update enables password changes made on z/VM to be more securely propagated to other systems, including z/OS, using applications such as the IBM Tivoli Directory Integrator.

5.4.3 System z cryptographic solution

System z is well equipped to address modern cryptographic security needs. Since early in the 1990s, the hardware has shipped with one form or another of cryptographic processor included, and upgrades were available shortly after to allow you to customize and expand that solution. All the System z operating systems (z/OS, Linux on System z, z/VM, and z/VSE) provide the software to implement the necessary solutions. Today, the mainframe offers everything that you need for an effective cryptographic solution in your environment.

Cryptographic software support: z/VM

Virtual machines enable the sharing of System z hardware among many operating systems. As a virtualization solution, the z/VM operating system does not provide a direct interface into any of the cryptographic hardware, nor does it require cryptographic services itself. z/VM provides a means of sharing the hardware cryptographic resources among the operating systems that are hosted (known as *guests*). Cryptographic resources can be shared through z/VM as follows:

- For all available cryptographic accelerators, z/VM provides unlimited access to all guests. This includes access to the CP Assist Cryptographic Function (CPACF) and the CEX2C when configured as a pure accelerator (CEX2A).
- The secure key *Hardware Security Module* (HSM), also sometimes referred to as a *Tamper Resistant Security Module* (TSRM), is comprised of a highly specialized piece of equipment that is designed to be the basis of your cryptographic security solution. These devices are the *strongboxes* that protect your symmetric keys and our asymmetric private keys. For HSM processors (CEX2C), z/VM can assign them to guests as well, but like logical partitioning, z/VM must assign the domains to each guest. A guest can have more than one domain. Also, multiple guests can be assigned the same domain, but only one guest can be active in a domain at any time.

For z/VM TCP/IP prior to z/VM Version 5.3, z/VM TCP/IP provides Secure Sockets Layer (SSL) support through a program interface called *VMSOCK*. Calls are redirected to a Linux on System z guest under z/VM, which contains special code provided by z/VM. This implementation was available for TN3270 or programs that were written to take advantage of the interface.

With z/VM Version 5.3, the usage of the SSL was extended to Transport Layer Security (TLS) and now provides full support for TN3270, SMTP and FTP.

z/VM definitions

When an LPAR has been configured to benefit from hardware cryptography support, z/VM running in such an LPAR can use the hardware support for cryptographic operations to provide it to its guests. The way how z/VM provides this support is by gaining access to the Adjunct Processor (AP) queues to the guests. From a system implementation perspective, an AP of a Crypto Express2 feature is one of its internal cryptography engines (cryptography coprocessor units). Note, that AP designates to the processor, while AP ID specifies the number associated with it.

To make use of the accessible hardware by z/VM and to provide it to the guests, note the following rules:

- Each Adjunct Processor (AP) can have up to 16 usage domains assigned to it.
- Each usage domain:
  - Has a separate set of master keys for secure key operation stored in the CEX2C
  - Is associated with a separate AP queue
- The AP queues reside in the Hardware System Areas (HSA) and provide access to an AP.
- An AP queue can be identified by the AP number and the usage domain index.
- The AP numbers are assigned to the Cryptographic Candidate List or Cryptographic Online List in the LPAR activation profile.
- Each LPAR is assigned at least one usage domain which apply to all of the APs configured to this LPAR.
- An AP can be shared among 16 LPARs.

- The combination of usage domain and AP must be unique among active LPARs.

According these rules, a z/VM system can have up to 256 AP queues, which can be used by the z/VM guests.

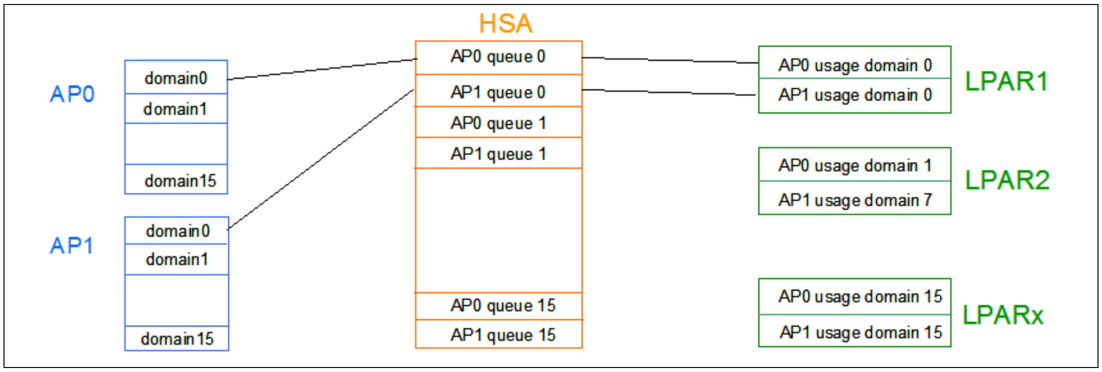


Figure 5-4 Mapping of AP queues to LPARs

In a z/VM environment it is expected, that the LPAR running z/VM will have access to multiple AP queues. There are two ways z/VM can provide access for the guests to the AP queues:

- Shared queue support
- Dedicated queue support

The shared queues support is the choice to provide for one or more Linux guests hardware encryption support for clear key operation (for example for SSL).[1] With shared queue support, z/VM intercepts and simulates AP instructions for all shared-queue guests, and then issues the instructions on behalf of the guest. If there are multiple queues available, z/VM decides which AP queue is actually used under the cover.

The dedicated queue support for a guest must be used, if the guest needs secure key support and relies on stored encryption keys in the hardware coprocessors. For guests using dedicated-queue support, z/VM does not intercept and instead allows the guest to execute the AP instructions under SIE. In this case, no virtualization of AP queues is done.

Cryptographic software support: z/OS

z/OS has provided cryptographic solutions on System z longer than any other operating system. The Integrated Cryptographic Services Facility (ICSF) solution is available.

In the z/OS world, the Integrated Cryptographic Services Facility (ICSF), an unpriced optional feature of the Cryptographic Services (a base element of z/OS), is considered to be synonymous with the cryptographic solution. In addition to providing the Common Cryptographic Architecture (CCA) APIs and interfacing with the hardware, ICSF interfaces with the External Security Manager to ensure that requesters are authorized to access the cryptographic services and resources that they are requesting.

Cryptographic software support: Linux

The System z platform with its high availability, connectivity, scalability, and virtualization, provides an ideal platform on which to host Linux servers. Linux implementations have access to all of the cryptographic packages that any other Linux deployment might have, which would typically include many software cryptographic toolkits and products. What makes Linux

[1] This applies also for z/OS and z/VSE guests, if only clear key support is necessary.

distinct in this area is that it also has access to the System z hardware cryptography environment.

Cryptographic software support: z/VSE

The z/VSE operating system also supports cryptographic processing to support SSL for its TCP/IP-based processes. If cryptographic accelerator hardware is available, z/VSE uses it automatically. This includes CPACF and CEX2C/CEX2A hardware. If there is no cryptographic hardware available, z/VSE performs the necessary cryptography in software.

5.5 z/VM virtual networking

In this chapter we are describing at a high level the networking facilities that z/VM provides to its guest Virtual Machines. We namely address here:

- The z/VM TCP/IP stack
- The Guest Lan
- The z/VM support of VLAN (IEEE 802.1Q)
- The z/VM Virtual Switch (VSWITCH)

Each of these options provides a highly secure communication path, under control of CP, that is not detectable or in any way “sniffable” by other virtual machine (that is, no other virtual machine can eavesdrop on the data moving between virtual machines). Of course, these virtual network connections are only as secure as the guests connected to them.

5.5.1 The z/VM TCP/IP Stack

The z/VM TCP/IP stack is an optional service machine executing the TCP/IP stack task under CMS, that enables virtual machines hosted by the z/VM instance to be connected to external networks with some degree of protection against denial-of-service (DoS) attacks. The z/VM stack can be seen, from the networking standpoint, as a dual-homed TCP/IP stack: one network adapter being a physical adapter connected to the physical network, the other adapter being a virtual one connected to a virtualized guests network or point-to-point connections (e.g. Virtual Channel-to-Channel). Figure 5-5provides a simplified view of the topology, where the z/VM TCP/IP stack acts as a router between the external physical network and the virtualized network(s).

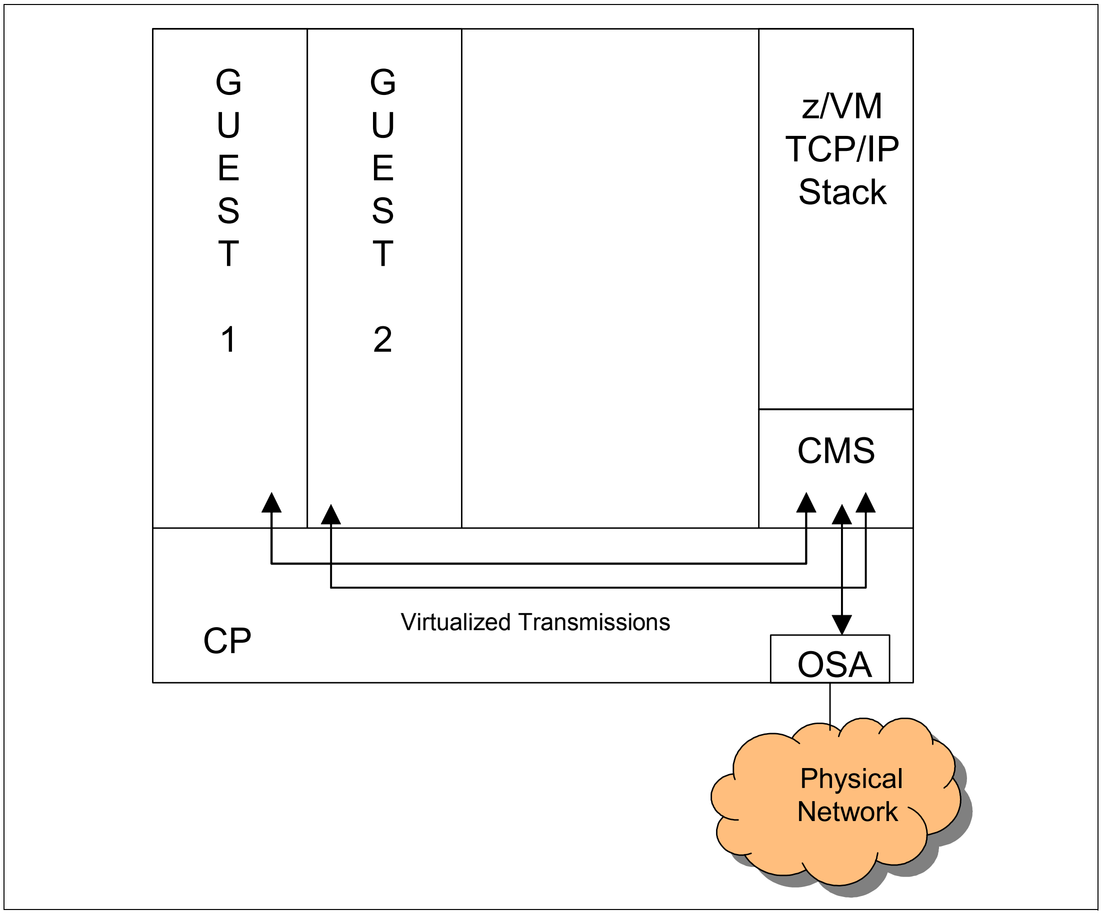


Figure 5-5 z/VM TCP/IP stack topology (simplified)

Protection of the z/VM TCP/IP stack and the connected virtual networks

The z/VM stack is the single application running in the CMS service machine, and as such benefits of the virtual machine environmental security, however it is physically connected to networks that could be by essence non-secure.

The stack hosts built-in protections against known denial-of- service (DoS) attacks such as Smurf, Fraggle, Ping-o-Death, Kiss of Death (KOD), Blat, SynFlood,Stream, R4P3D, etc... In case of DoS conditions detected, the received packets are discarded with proper alert messages at the guest VM console. DoS detailed information can be displayed or reset with the VM NETSTAT command.

The TCP/IP stack audits all changes to the IP or z/VM system configuration made using the IPCONFIG, OBEYFILE or NETSTAT command.

5.5.2 The z/VM Guest LAN

the z/VM control program (CP) provides a feature known as Guest LAN. This feature enables the users to create multiple Virtual LAN segments within a z/VM environment. Guest LANs do not have a physical connection to the external network. Instead, they must use a router (z/VM TCP/IP stack for instance, or a guest machine acting as a TCP/IP router) or a Virtual Switch which is itself connected to the external network. The “Virtualized Transmissions” mentioned in Figure 5-5 can therefore be implemented as Guest LANs.

A virtual machine accesses a Guest LAN using a virtual Network Interface Card (NIC), which emulates either a System z HiperSockets adapter or a QDIO Ethernet adapter. The most recent development is the VSWITCH which adds connection of a Virtual QDIO Ethernet NIC to a LAN Segment that is connected to a OSA card, eliminating the need for routing. IPV6 support was added to the HiperSockets and Ethernet virtual NICs. We are addressing the VSWITCH in 5.5.4, “The z/VM VSWITCH” on page 128.

Figure 5-6 is an example of Guest LANs topology in a z/VM system hosting several Linux for System z guest instances. Note that only one Linux guest connects to the external network, this guest provides external routing and firewall services for the other Linux guests connected to their individual Guest LAN(s).

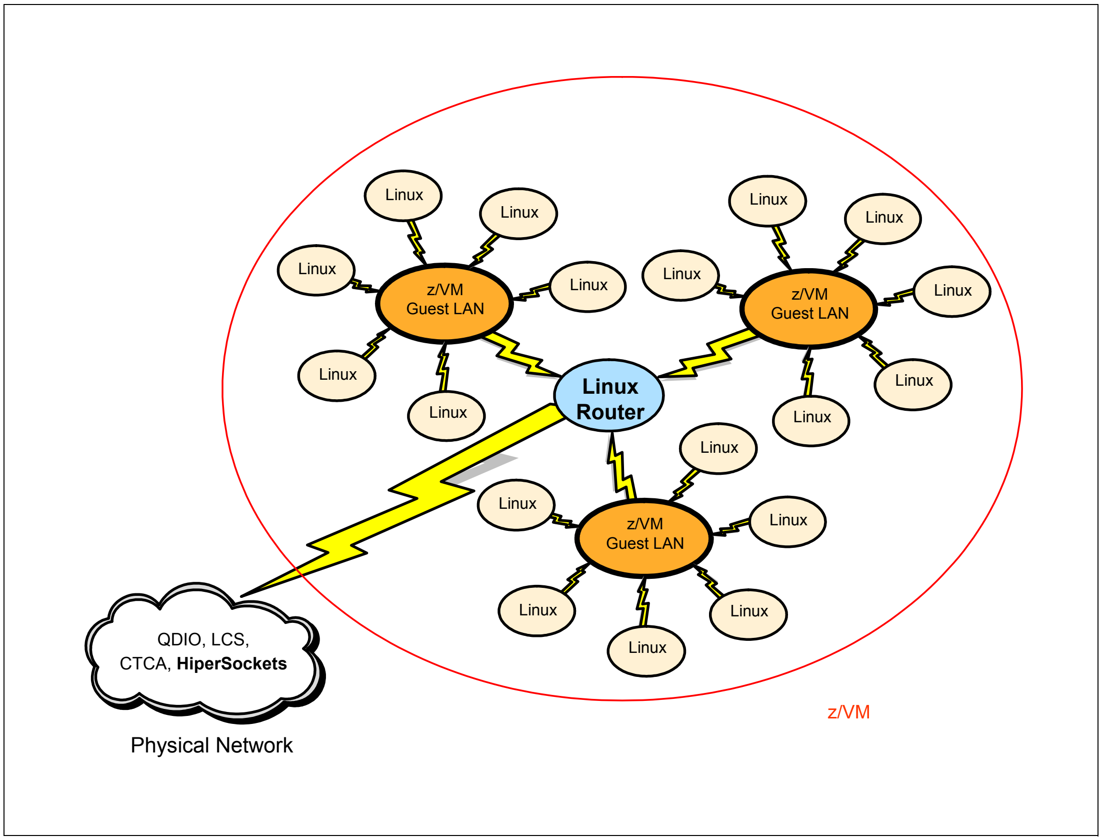


Figure 5-6 Guest LAN topology

Security of the Guest Lan topology

As already mentioned the Guest Lan is simulated by CP which takes care of the proper isolation between the guest machines connected to the Guest LAN. The security of the virtualized network topology is to prevent any actions by users that would inadvertently or on purpose modify the topology of the LAN and would give unexpected connectivity to guest VMs. This is covered as follows.

Configuration of physical devices

The physical devices are defined in the System’s IOCDS in terms of channel paths and control units they are connected to, along with the specifications of channel paths (and devices if the channel path is shared) allocation to logical partitions. This requires proper privileges at the Support Element to run the standalone IOCP, if it is used to build the IOCDS, and to put the IOCDS file in write-enabled state.

If the IOCDS is build using the regular IOCP or HCD, that is running in VM, or under a z/OS guest operating system, the host operating system requires the users to have proper privileges to perform these functions. In addition to this, the logical partition where z/VM runs can also be prohibited at the SE to write a new IOCDS.

The defined devices should then be ATTACHed or LINKed to CP by explicit CP commands.

There is no API available in the guest VM that would enable the VM to become in control of these configuration data.

Configuration of the Guest LAN

The following process has to be followed in order to configure a Guest Lan:

1. The Guest LAN is defined to CP using the CP DEFINE command or the definition can be part of the SYSTEM CONFIG. The LAN is identified to CP by the combination of an owning VM name and a LAN name. All commands aimed at controlling this very LAN should specify this combination.
2. Virtual Network Interface Cards (NIC) are created that will be later attached to guest VMs. This is also achieved via the CP DEFINE command.
3. Finally the CP COUPLE command is used to attach the NICs to the guest VMs and the specific Guest LAN.

Guest LANs can be owned by a virtual machine or the SYSTEM, that is CP. Guest LANs can be restricted or non-restricted. From a security point of view the Guest LANs should always be restricted to keep virtual machines from using them without explicit authorization. When Guest LANs are restricted, RACF takes care about authorization. VMLAN is the RACF profiles class which is used in this case.

5.5.3 z/VM VLAN support

VLAN is a networking technique where the low level Ethernet data frames conveyed by a physical network can be “tagged”, that is additional information is added to each frame, as belonging to a specific “virtual LAN” (VLAN). With this technique, multiple virtual LANs subnets can coexist on the same physical network. VLAN-aware adapters at the TCP/IP hosts connected to the physical network only respond to the traffic tagged for the VLAN they are connected to. The VLAN specifications can be found in the IEEE 802.1Q standard.

Figure 5-7 is providing a very high level view of a network topology when the VLAN technique is used. Using VLANs requires to have VLAN-aware routers, switches an adapters connected to the network. In this figure the switch accepts all regular or VLAN-tagged frames through the “hybrid” trunk port, then it routes or switches the tagged frames to the proper VLAN-aware devices on the basis of the VLAN numbers the frames and the devices belong to.

Virtual LANs (VLANs) facilitate easy administration of logical groups of machines that can communicate as though they were on the same local area network (LAN). However beyond this implicit simplification of physical network implementation and administration, they are also providing a degree of low-level security by restricting direct contact with a server to only the set of stations that comprise the VLAN.

On System z, where multiple stacks, in logical partitions or guest VMs, may exist potentially sharing one or more OSA-Express features, VLAN support is designed to provide a greater degree of isolation.

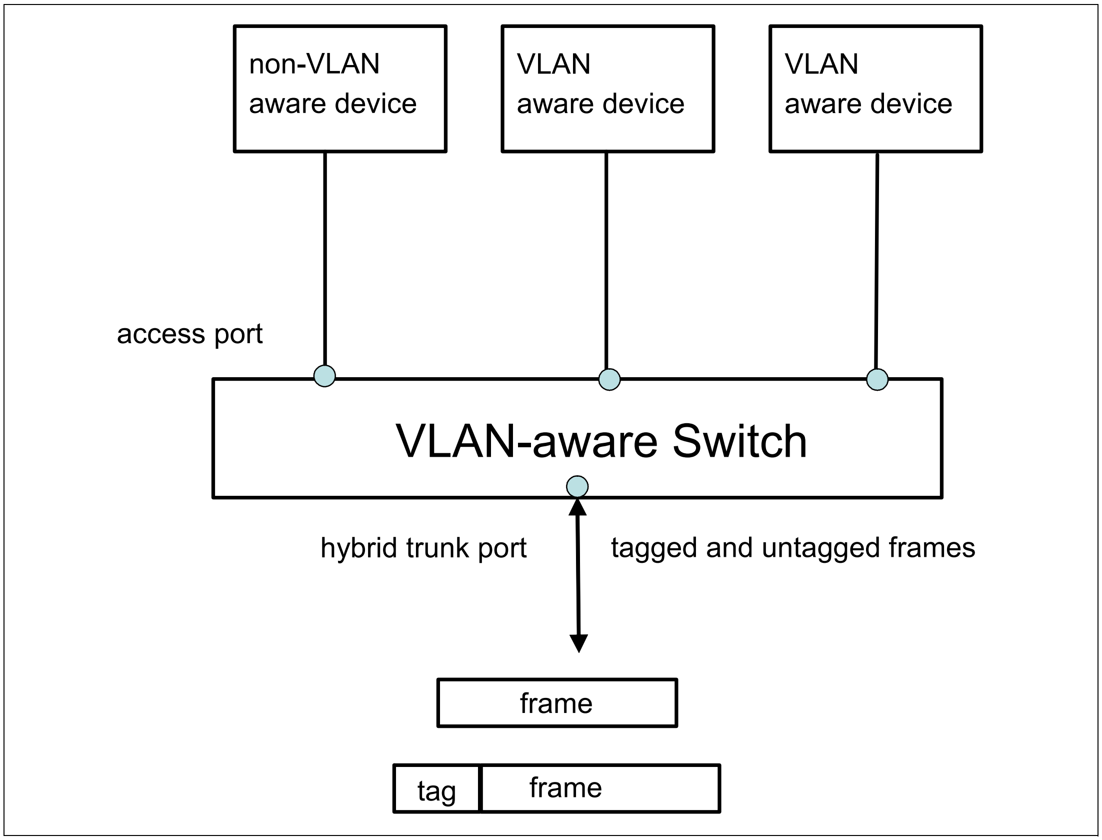


Figure 5-7 VLAN topology

z/VM supports the IEEE 802.1Q standard for z/VM Virtual Switch, z/VM QDIO Guest LAN, and z/VM HiperSockets Guest LAN, allowing z/VM guests to create and participate in virtual LAN configurations.

5.5.4 The z/VM VSWITCH

The z/VM Virtual Switch (VSWITCH) builds on the Guest LAN technology. VSWITCH connects a Guest LAN to an external network using a physical OSA-Express interface. When defining a VSWITCH additional OSA-Express physical devices can be specified as backups adapters. To summarize, the z/VM VSWITCH is:

- A special-purpose Guest LAN Ethernet IPv4 and IPv6
- With a built-in bridge to outside network
- IEEE VLAN capable
- Each Virtual Switch has up to 8 separate OSA-Express connections associated with it
- It is created in SYSTEM CONFIG or by the CP DEFINE VSWITCH command

Because the VSWITCH is essentially connected to the physical LAN, the requirement for an intermediate router between the physical and (internal) Guest LAN segments is removed. Beside reducing network latency and the overall CPU consumption, the VSWITCH also removes the need for specialized skills to configure and administer z/VM-based routers.

Figure 5-8 describes at a high level the topology of a network implemented with a z/VM VSWITCH. The figures also shows the capability of the VSWITCH to switch packets on a VLAN tagging basis.

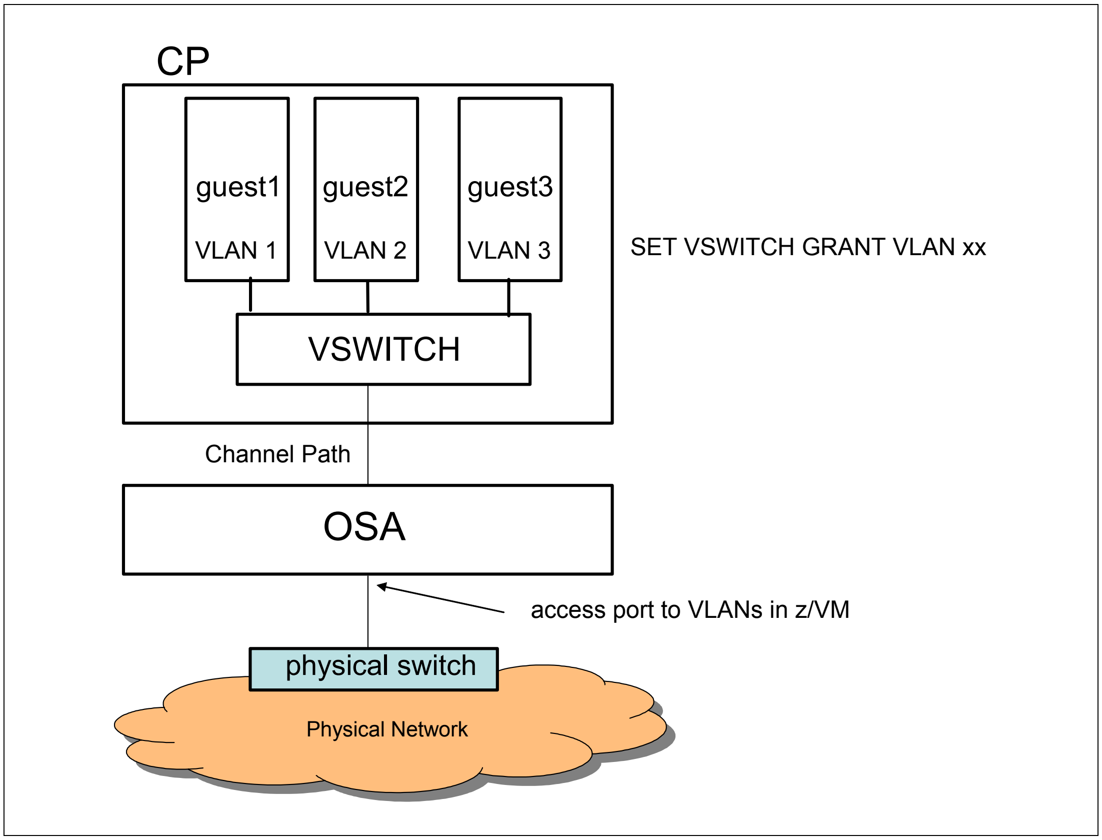


Figure 5-8 z/VM VSWITCH network topology.

The z/VM VSWITCH can operate at Layer 3 (network layer) or Layer 2 (data link layer) of the OSI model.

Security considerations on the VSWITCH

A VSWITCH is always owned by the SYSTEM, that is CP, and is always restricted by CP for coupling by a virtual NIC.

For maximum isolation between guests the user should configure the VSWITCH as VLAN AWARE. Virtual LAN segments are controlled by the virtual switch, not the guests, and an external security manager (ESM), such as RACF can control access to the virtual switch and VLANs. VLAN in conjunction with the port type designated for a guest, defines the ingress and egress rules that are applied to this connection. The guest VLAN IDs are specified through the grant option of the SET VSWITCH command or system configuration file.

When using RACF, the VMLAN class of profiles is used for access control. From an access control perspective, guest LANs and virtual switches are treated the same way. The VMLAN class contains two sets of profiles to protect LANs: base profiles that control the ability of a z/VM user to use a LAN, and for IEEE VLAN-aware virtual switches, VLAN ID-qualified profiles that are used to assign a user to one or more IEEE VLANs.

When used with a trunk connection to an Ethernet switch, the z/VM system administrator controls the assignment of a virtual server to a specific VLAN. CP also controls the capability of a virtual server to “sniff” the virtual network and to talk to other servers on the virtual network.

5.6 z/VM certification

The *Common Criteria* is an internationally recognized International Standards Organization (ISO) standard that is used by governments and other organizations to assess the security and assurance of technology products. Under the Common Criteria, products are evaluated according to strict standards for various features, such as security functionality and the handling of security vulnerabilities.

The LSPP security labeling system

Central to LSPP security is its system of security labeling. Each object in a z/VM LSPP-compliant system has a *security label*, or *SECLABEL*, that designates its relative confidentiality and its membership in a security category. An object's security label defines what sort of data it can contain and, by implication, what sort of data it cannot contain.

Mandatory access control (MAC) is a security policy that governs which subjects can access which objects, and in what way, based upon certain rules. These rules are the **-property* and the *simple security property*. RACF commands are used to manage MAC for Control Program commands, DIAGNOSE codes, and system functions. MAC restricts a subject's access to an object.

CAPP

The Controlled Access Protection Profile (CAPP) specifies a set of security functional and assurance requirements, including access controls that are capable of enforcing access limitations on individual users and data objects. CAPP-conforming products also provide an audit capability which records the security-relevant events which occur within the system.

CAPP was derived from the requirements of the C2 class of the U. S. Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), dated December 1985. This protection profile provides security functions and assurances which are equivalent to those provided by the TCSEC and replaces the requirements used for C2 trusted product evaluations.

In z/VM, the CAPP requirements are met through the following specific mechanisms:

- Discretionary Access Control (DAC)

  A method of restricting access to data objects based upon the identity of users or groups to which the users belong. DAC protects system objects from unauthorized access by any user. Normally, permission to access an object is granted by the owner of the object. Occasionally, it can be granted by someone else, such as a privileged administrator.

- Auditability of Security-Relevant Events

  The recording of facts that describe a security-relevant event taking place in a computing system. In general, a security-relevant event is one that occurs in a computing system that, for better or for worse, affects the safety and integrity of the system's processes and data. The facts recorded that describe such an event include the time and date of the event, the name of the event, the name of the system objects affected by the event, the name of the user who caused the event to occur, and additional information about the event.

  In general, the security-relevant events in z/VM are:

  – CP commands
  – DIAGNOSE functions

   - Communication among virtual machines
- Object Reuse

  A practice that prevents any newly-assigned storage object from making available to its new owner any data that belonged to its former owner. This includes any encrypted data. Object reuse also requires the elimination of any residual user authorization access to a previously existing object. This ensures that if another, new object occurs in the system later under the same name, the subjects having access to the old object will not have access to the new one.
- Identification and Authentication

  A method of enforcing individual accountability by providing a way to authenticate a user's identity uniquely and unambiguously. Thus, any security-relevant action users might take can be attributed to them.

z/VM V5.1 was evaluated for conformance to the Controlled Access Protection Profile (CAPP) and the Labeled Security Protection Profile (LSPP) of the Common Criteria, both at Evaluation Assurance Level (EAL) 3+.

z/VM V5.3 was evaluated with the RACF Security Server optional feature for conformance to the Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP) of the Common Criteria standard for IT security, ISO/IEC 15408, at Evaluation Assurance Level 4, augmented by flaw remediation procedures (EAL4+). This satisfies the statement of direction made in the Software Announcement dated February 6, 2007.

z/VM V5.4 has not been evaluated for conformance, but is designed to meet the same standards.

5.7 Referenced material

The following material has been used to gather information about z/VM:

- Redbook: SG24-7316, *Introduction to the New Mainframe: z/VM Basics*
- Redbook: SG24-7471, *Security on z/VM*
- URL: www.ibm.com/servers/eserver/zseries/library/techpapers/pdg/gm130145.pdf, *z/VM Security and Integrity*

6

Other operating systems

This chapter provides information about other operating systems than z/VM or z/OS. There are two other operating systems:

- z/VSE
- *zTPF* (Transaction Processing Facility)

6.1 z/VSE and security

The chapter discusses mainly the IBM Blueprint and the security aspects of z/VSE and how they are related to each other. In this conjunction we will refer to the IBM Redbooks Publication SG2-7691: *Security on IBM z/VSE*.

z/VSE is designed to provide robust, cost-effective solutions for customers with a wide range of processor capacity requirements. Especially customers with lower processor capacity requirements value the relatively small cost of operation and administration.

As with all IBM System z operating systems, z/VSE can run as a single operating system or share the mainframe with others by using the LPAR capabilities of System z. z/VSE can also run as a guest on the IBM z/VM system to use a customized environment that might contain emulated hardware features that are not necessarily installed on that particular model of server.

The Transmission Control Protocol/Internet Protocol (TCP/IP) in z/VSE allows interconnection to other platforms. With TCP/IP, z/VSE became accessible through the Internet. Special security functions are provided to make those connections secure.

TCP/IP enables z/VSE to support single sign-on with Lightweight Directory Access Protocol (LDAP) and secured integration in a multi platform environment using z/VSE connectors.

z/VSE supports the System z cryptographic solutions. They are used within the connection security and also within the tape encryption.

6.1.1 z/VSE and the IBM Security Blueprint

The IBM Security Blueprint uses a product-agnostic and solution-agnostic approach to categorize and define security capabilities and services that are required to answer business security requirements or areas of concern categorized by the IBM Security Framework. It also defines a common vocabulary to use in further discussions.

In the blueprint, IBM aims to identify architectural principles, that are valid across all domains, and fundamental services within and across the domains. It also highlights applicable best practices and IT standards.

This part will describe which z/VSE areas are related to the IBM Security Blueprint on the Security Services and Infrastructure layer, as highlighted on Figure 6-1. As a reference material we use the IBM Redbooks publication SG2-7691: *Security on IBM z/VSE.*

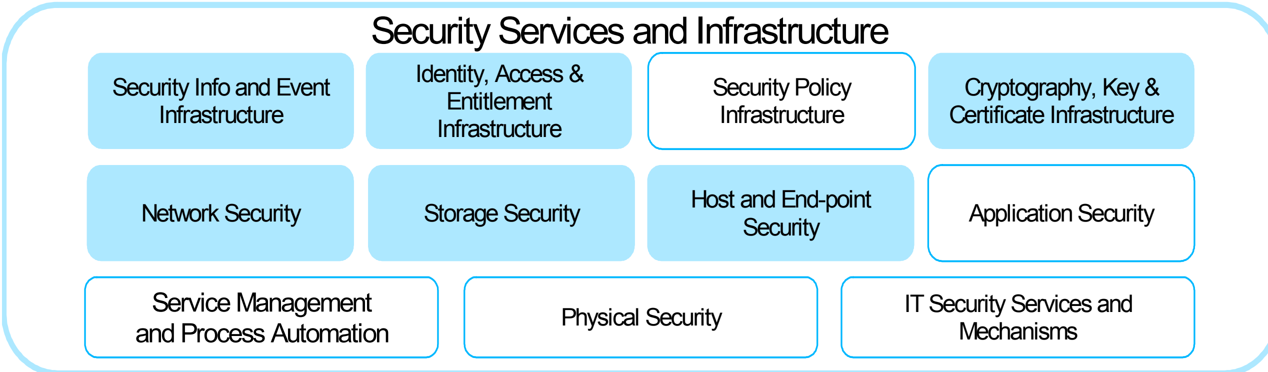


Figure 6-1 IBM Security Blueprint disciplines on z/VSE

Security Information and Event Management Infrastructure

On z/VSE, Security Information and Event Management disciplines are provided by components described under following headings:

- "*Intrusion detection*"

  The Intrusion detection tracks all denied sign-ons. If the maximum sign-on is reached, he User ID is disabled.

- "Basic Security Manager"

  BSM on z/VSE sends messages to the console for detect, unauthorized attempts to enter the system and or to access BSM-protected resources.

Identity, Access and Entitlement Infrastructure

On z/VSE, Identity, Access and Entitlement Infrastructure disciplines are provided by components described under following headings:

- "*Basic Security Manager*"

  BSM on z/VSE provides basic security support that includes user verification, resource authorization and logging capabilities.

- "*Single sign-on and LDAP*"

  LDAP on z/VSE can improve functionality and ease-of-use by enabling cost-effective administration when keeping information about services, users and other objects in a clear and consistent manner. z/VSE supports the single sign-on.

Security Policy Infrastructure

On z/VSE, Security Policy Infrastructure discipline has no role

Cryptography, Key and Certificate Infrastructure

On z/VSE, Cryptography, Key and Certificate Infrastructure disciplines are provided by components described under following heading:

- "*System z cryptographic solution*"

  Encryption Facility (EF) for z/VSE gives another encryption option and complements z/VSE support for the encrypting tape drives IBM TS1120 and TS1130.

Network Security

On z/VSE, Network Security discipline is provided by components described under following heading:

- "*Connector security*"

  On z/VSE the TCP/IP connections to z/VSE are based upon common standards, like Secure Sockets Layer (SSL) and Secure Electronic Transaction (SET).

- "*Secure FTP*"

  Secure FTP for z/VSE support user authentication, confidentiality and data integrity by using certificates, data encryption and secure hash functions.

Storage Security

On z/VSE, Storage Security discipline is provided by components described under following headings:

- "Basic Security Manager"

  BSM on z/VSE can be used to protect system resources like CICS resources, z/VSE files and z/VSE libraries which can be accessed out of CICS or from a batch job.

- "*System z cryptographic solution*"

  Encryption Facility (EF) for z/VSE is designed to allow secure exchange of the encrypted data with other locations within your company or even with other partners.

Host and Endpoint Security

On z/VSE, Host and Endpoint Security disciplines are provided by components described under following headings:

- "*Online security*"

  On z/VSE Customer Information Control System (CICS) manages the sharing of the resources and the integrity of data. CICS can use a security manager, such as BSM, to make the security decisions.

- "Batch security"

  On z/VSE Batch the User ID and password are specified on the JOB statement or on the JCL ID statement. If on online a batch job is submitted, the new job inherits the User ID and authorization from the online session.

- "*Compliance to policy*"

  z/VSE system is secured by using security features of the System z hardware, by maintaining compliance to security policy within operating practices, and by making use of the functions of the installed security manager

Application Security

On z/VSE, Application Security discipline has no role

Service Management and Process Automation

On z/VSE, Service Management and Process Automation disciplines has no role

Physical Security

On z/VSE, Physical Security discipline has no role

IT Security Services and Mechanisms

On z/VSE, IT Security Services and Mechanisms disciplines has no role

6.2 z/TPF and security

The chapter discusses mainly the IBM Blueprint and the security aspects of z/TPF and how they are related to each other. In this conjunction we will refer to book G299-0768: *z/TPF - The evolution of transaction processing to open standards* and the IBM Redbooks publication SG24-6776: *Introduction to the New Mainframe: Security*.

6.2.1 z/TPF

System z Transaction Processing Facility (z/TPF) is a special purpose operating system used by a few, very large installations. It was once known as the Airline Control Program (ACP) and was written for airline reservation systems. It is still used for this purpose and has been extended for other very large reservation systems and similar high volume transaction processing requirements.

TPF can use multiple mainframes and LPARs in a loosely-coupled environment to routinely handle thousands of transactions per second while experiencing uninterrupted availability measured in years. Very large terminal networks, including special-protocol networks used by portions of the reservation industry, are common. Early versions used applications written to rather limited, special interfaces and written in assembly language. Recent versions of TPF have added a high volume Web server, application programming in C, standard links to relational databases on z/OS systems, and cross-platform application development using z/OS.

6.2.2 The z/TPF family of products

The z/TPF Database Facility (z/TPFDF) is co-requisite to z/TPF itself. z/TPFDF provides the z/TPF programmer with a higher level interface to the z/TPF database, maintaining the performance attributes of z/TPF while offering both virtualization of the z/TPF data constructs and improved maintainability and accessibility.

The IBM TPF Toolkit for WebSphere Studio is an application development platform built on the open and standards-based Eclipse tooling framework. The IBM TPF Toolkit for WebSphere Studio includes a programmable editor, C/C++/Assembler build support, full-featured debugger, performance analyzer, high performance remote file transfer mechanism, and much more.

The TPF Operations Server is a console automation and enhancement application for the TPF system. This PC-based application provides a tool for the administration and maintenance of your TPF system through TPF operations consoles. The TPF Operations Server runs outside the TPF system complex and allows you to monitor your TPF system, automate operational tasks, and diagnose problems quickly and accurately, thereby improving the productivity of your operations staff and enhancing system availability. More information about the TPF Family of Products is available on the IBM TPF Web site at:

http://www.ibm.com/tpf

6.2.3 z/TPF and the IBM Security Blueprint

The IBM Security Blueprint uses a product-agnostic and solution-agnostic approach to categorize and define security capabilities and services that are required to answer business security requirements or areas of concern categorized by the IBM Security Framework. It also defines a common vocabulary to use in further discussions.

In the blueprint, IBM aims to identify architectural principles, that are valid across all domains, and fundamental services within and across the domains. It also highlights applicable best practices and IT standards.

This part will describe which z/TPF areas are related to the IBM Security Blueprint on the Security Services and Infrastructure layer, as highlighted on Figure 6-2. As a reference material we use z/TPF manual GTPS-7MS5: *z/TPF Security.*

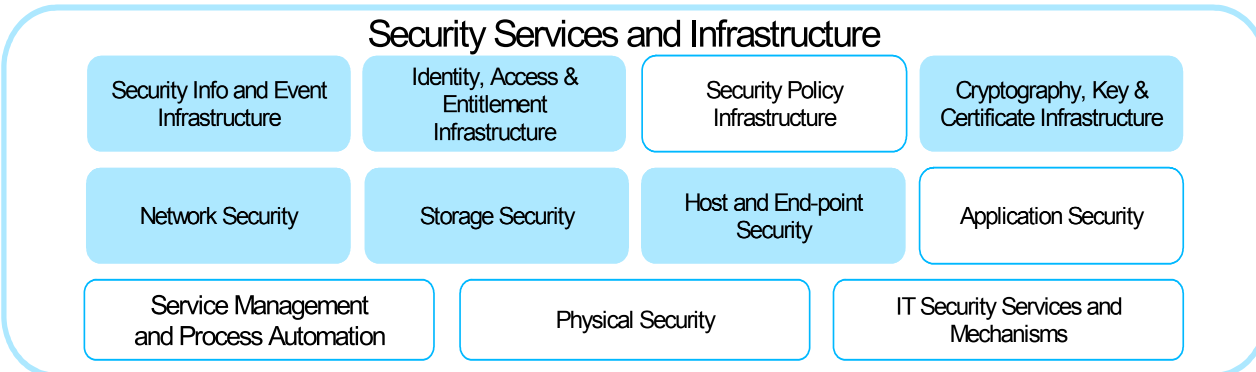


Figure 6-2 BM Security Blueprint disciplines on z/TPF

Security Information and Event Management Infrastructure

On z/TPF, Security Information and Event Management disciplines are provided by components described under following headings:

- "*Internet security*"

  With growing TCP/IP networks, Internet security has become an important issue. Here are two well-known types of security for the internet.

  – Secure Socket Layer (SSL) allows application to communicate in a secure manner over TCP/IP network.
  – Firewall support, which includes among others, the following functions:
    - z/TPF system provides protection against denial-of-service attacks directed forward TCP/IP stacks, along with the safeguards to prevent many potential denial-of-service attacks in the future.
    - Packet filtering

      Traffic limiting: When the traffic limit is reached, the z/TPS Internet service is prevented from reading more messages which, for TCP, prevent the remote end from sending more data.

      TCP connection limiting: One can limit the number of active remote client connections for a specifies z/TPF Internet service.

Identity, Access and Entitlement Infrastructure

On z/TPF, Identity, Access and Entitlement Infrastructure disciplines are provided by components described under following headings:

- "*TPF operator security*"

  User passwords and groups are used to control file access. The z/TPF system provides a set of default user names (userids) and group names and one can define additional user names and groups and assign passwords.

Security Policy Infrastructure

On z/TPF, Security Policy Infrastructure discipline has no role

Cryptography, Key and Certificate Infrastructure

On z/TPF, Cryptography, Key and Certificate Infrastructure disciplines are provided by components described under following heading:

- "*Symmetric cryptography with clear key APIs*"

  For symmetric cryptography, the z/TPF system uses the central processor assist for cryptographic function (CPACF) to perform bulk data cryptographic operations and encrypt keys in the z/TPF keystore which contains active and archived keys for symmetric key ciphers used by z/TPF applications.

- "*Protecting data at rest*"

  z/TPF system support tape encrypting with an out-of-box configuration for tape drive IBM TS1120.

Network Security

On z/TPF, Network Security discipline is provided by components described under following heading:

- "Secure Sockets Layer"

  SSL support enabled z/TPF application to use SSL. SSL support is based on OpenSSL version 0.9.7 open source package that was ported, ties an SSL session to a specific process. The OpenSSl open source package is available at http://www.openssl.org. This code was modified to work with z/TPF system and it is important to use the code modified and shipped by IBM.

- "*Protecting data in flight*"

  There are three basic methods of protecting data in flight:

  – Application using SSL: This is the standard industry SSL solution. SSL is a set of rules governing authentication and encrypted communication between client an servers.
  – Application using middleware using SSL: many middleware packages use SSL protocol to provide a secure way of transferring data to and from the z/TPF system.
  – Application performs data encryption itself: For some condition one might need to encrypt or decrypt data outside the scope of SSL. For example, one might need to encrypt sensitive data before sending it through the network by using own transport mechanism.

Storage Security

On z/TPF, Storage Security discipline is provided by components described under following headings:

- "*Protecting data at rest*"

  Stored data can be protected on a permanent medium by using the following methods:

  – Tape encryption which provide with greater ability to protect information if tape cartridges are lost or stolen by supporting the storage of the data in encrypted form.
  – Application performing the encryption itself using either Secure symmetric cryptography or Clear key APIs.

- "*Protecting data in use*"

  During the life of a transaction, there are times when data is stored unencrypted in memory. That data should not be exposed in system dumps, operator command displays and debugger displays. The primary method for protecting data in use on z/TPF is nondisplayable storage.

- "*Database security*"

Access to a file in the file system is POSIX-compliant and is controlled by the effective user ID, effective group ID and access permissions that determine if a process can access a file. The access permissions available for a file are read, write and execute (or any combination).

z/TPS system allows to have separate databases on the same server. Database on one subsystem are logically separated from database on another. Applications on one subsystem can only see the database on that subsystem.

Host and Endpoint Security

On z/TPF, Host and Endpoint Security disciplines are provided by components described under following headings:

- "*Program security*"

  One can control the programs that are added to the z/TPF system. To load a new program to the z/TPF system, it is not enough to use FIle Transfer Protocol (FTP), the program must be also loaded using a LOAD process which can be controlled.

- "*Memory security*"

  The z/TPF system uses virtual addressing which restricts ECBs (= Entry Control Block is same as process) from accessing other ECBs. So when a new ECB is spawned, it has access only to its own storage; it cannot see or modify another ECB's storage.

Application Security

On z/TPF, Application Security discipline has no role

Service Management and Process Automation

On z/TPF, Service Management and Process Automation disciplines has no role

Physical Security

On z/TPF, Physical Security discipline has no role

IT Security Services and Mechanisms

On z/TPF, IT Security Services and Mechanisms disciplines has no role

6.3 Referenced material

The following material has been used to gather information about z/VSE:

- Redbooks: SG24-7691, *Security on IBM z/VSE*
- G299-0768: *z/TPF - The evolution of transaction processing to open standards*.
- Redbooks SG24-6776: *Introduction to the New Mainframe: Security*
- z/TPF manual GTPS-7MS5: *z/TPF Security*

7

z/OS Security

This section builds on the description of hardware features which are described in Chapter 4 and shows how the z/OS Operating System is an operating system with integrity.

This chapter provides a description of the elements of z/OS which contribute to its rich set of security functions. It includes a description of those z/OS components which supply security services, but also describes how the kernel of the z/OS Operating System (previously known as MVS) has fundamental controls and how it places well-defined boundaries on executing programs.

In this chapter we discuss,

- The distinction between the z/OS Operating System and the rest of z/OS
- The use that the z/OS Operating System makes use of the System z hardware protection capabilities to provide integrity.
- z/OS and the use of SAF
- RACF
- Sysplex capabilities
- SMF
- System Logger
- The z/OS subsystem interface
- ......and a wealth of other features of z/OS

However, it starts with an attempt to show how z/OS relates to the IBM Security Blueprint.

7.1 z/OS and the IBM Security Blueprint

As we have seen in Chapter 3, z/OS is a very comprehensive package of software components comprising an operating system and multiple other pieces of software which enhance and expand its capabilities. Each of these components has security capabilities and characteristics which can be explored so that we can understand how z/OS on System z complements the IBm Security Framework and the IBM Security Blueprint.

This first part of this chapter shows which z/OS components we are examining, and how they contribute to the functions needed by the IBM Security Blueprint. Some terms are used here which may need reference to the rest of the chapter.

Please remember that this book is focussed on security and the mainframe. I have taken the time and space to explain some of the internal intricacies of *z/OS integrity* as this is fundamental to z/OS security. It shows that z/OS security has sound and firm foundations.

However I have not explained how *all* the components of z/OS work, nor is that the intention. The aim is to demonstrate the richness of the set of *security functions* in z/OS. Neither do I make any attempt to show how to use the features I describe. This chapter is too small for that. Instead I recommend the reader examine one of the following redbooks,

Designing for Solution-Based Security on z/OS, SG24-7344

ABCs of z/OS System Programming Volume 6, SG24-6986

Each of these books cover z/OS security at far a greater level of detail than you will find here; the first with a view to designing applications which are secure; the second with a view to explaining the security infrastructure.

But let us commence with how the eleven elements of the IBM Security Blueprint relate to z/OS security features. The following is organized by each of those eleven elements.

7.1.1 Security Information and Event Management Infrastructure

z/OS provides several mechanisms for tracking events over long periods of time, and with great temporal accuracy.

The most significant of these from a security standpoint is the component called System Management Facility or SMF. This can provide a single sysplex-wide view of multiple security events.

z/OS has an integrated message repository known as SYSLOG. This to can be organized to produce a sysplex-wide view of message production.

In addition to the above two sources of security events is a third mechanism for the tracking of software and hardware error events and known as LOGREC. Due to the high temporal accuracy of this log, it can be used together with the above logging mechanisms.

7.1.2 Identity, Access, and Entitlement Infrastructure

RACF provides a single view of identity and access control for the z/OS sysplex. RACF can be accessed via LDAP if required, providing an alternate view of this information.

7.1.3 Security Policy Infrastructure

RACF provides both the repository for security information, and a means of defining detailed policy. It can allow for policy changes to be brought into play in a consistent manner across a sysplex.

7.1.4 Cryptography, Key, and Certificate Infrastructure

z/OS provides strong support in the area of cryptographic support. Together with the CryptoExpress2 feature of System z, ICSF provides a wealth of services in terms of encryption, MACing, hashing, financial PIN management and signing. It provides both secure key and clear key support for symmetric keys, and secure key support for asymmetric keys. ICSF also supplies interfaces to the clear key capabilities of the CPACF processor.

z/OS supplies a full certification authority and certificate management solution in the form of z/OS PKI services.

RACF supplies the RACDCERT command which can be used for the generation and management of certificates.

z/OS provides a wealth of different secure transport mechanisms over TCP/IP, such as SSL, AT-TLS, and IPSec services.

7.1.5 Network Security

Networking capability on z/OS is provided by the z/OS Communications Server. This supplies a wealth of security capabilities including IP Filtering and Intrusion detection services as well at IPSEC services, and Application Transparent TLS.

7.1.6 Storage Security

z/OS provides a rich set of functions for the management of storage, including a wide range of access methods to disk data and to tape data, each of which makes extensive use of the access control facilities in RACF.

Data can be organized independently of disk storage volumes, by making use of Catalogs which classify data by name. The data is access controlled based on the name, using RACF profiles.

In addition data can be classified, stored and secured in hierarchical file system using UNIX conventions and UNIX access controls.

From the encryption point of view there are no significant storage mechanisms, although z/OS can access hardware encrypted tapes and disks by using extra key management software.

7.1.7 Host and Endpoint Security

As we explain later in this chapter, z/OS is an operating system with integrity. It is capable of using hardware base protection mechanisms to protect itself against modification or subversion.

7.1.8 Application Security

z/OS provides mechanisms for applications to make use of RACF based security queries. Queries can also be made over LDAP connections.

z/OS provides assured identities for work processing in each address space. This is used by all applications to ensure the correct access levels to resources.

7.1.9 Service Management and process automation

z/OS provides several mechanisms for the automated processing of work based on events. This includes the provision of the sub-system interface which can be used to trigger events in authorised applications.

7.1.10 IT Security Services and Mechanisms

z/OS provides several mechanisms for collecting information about security and configuration information. This can be provided by RACF commands, by RACF ISPF panels, by RACF API's and by LDAP query mechanisms.

Other information can be collected by using MVS commands to display the software and hardware configuration information.

7.2 The heart of z/OS

In order to discuss the security capabilities of z/OS we need to be sure that security constructs, identities and so forth can be relied upon. Operating systems can have a quality known as integrity which relates to the ability of the operating system to protect itself.

In this section we attempt to demonstrate that the z/OS Operating System does have integrity. We have seen that the integrity of any system can be defined as the its ability to protect itself from having its control subverted or compromised.

> **Note:** To say an operating system has integrity, is to assert that controls on its modification cannot be bypassed. If those controls are left open; i.e. a large number of people or processes have access to them, then this does not mean the operating system has no integrity; (although one might say that this instance has insufficient security). This merely means that the security controls on integrity are insufficiently tight.
>
> So integrity requires security controls. Conversely security requires integrity to ensure that those security controls cannot be compromised.

The aim of this chapter is to demonstrate that z/OS is a securable operating system. It has the tools and capabilities to be secured.

7.2.1 MVS? BCP? Kernel? What's in a name?

The term "z/OS" is used to describe the large set of software components and products which are sold together by IBM. However, at the heart of z/OS is the successor to the operating system previously known as MVS. In some of the z/OS manuals this is still referred to as MVS. Others refer to the BCP (Base Control Program) although this is also used to refer to the central component of the z/OS Operating System; what in other operating systems might be termed the "kernel".

We shall refer to the operating system component of z/OS as the "z/OS Operating System". This is to distinguish it from the entire set of software known as z/OS.

It is the z/OS Operating System which provides the bedrock of reliability, integrity and predictability of operation which are essential in supporting secure controls for applications.

Any set of secure controls has to be based on features which have these characteristics of reliability and predictability.

Chapter 3has described a wealth of features of System z hardware which can be relied on in maintaining security controls. The z/OS Operating System makes extensive use of these hardware features to provide further capabilities in software which provide a firm basis for architecting secure controls.

The discussion that follows does not reference all of the mechanisms that are available, but it serves to demonstrate how the hardware mechanisms of System z are used by the z/OS Operating System software mechanisms to provide a secure, controllable, base on which security constructs can be built and used.

7.2.2 Protection mechanisms

Storage access control by key

We have seen that the System z hardware has the capability of controlling access to the memory based on the storage key in the PSW. This is possible only because each page (normally 4096 bytes) of memory has a 4 bit storage key associated with along with some other bits. One of these other bits is known as the Fetch-Protection bit.

> **Note:** Remember that when we are discussing "Storage Keys" we are discussing keys associated with memory. The zArchitecture manuals refer to this as "Storage".

Storage access works as follows. (This description is slightly simplified.)

If the storage key in the PSW is zero, then the access is granted. If the storage key in the PSW matches the storage key in the page of storage (shown as ACC in Figure 7-1 on page 147) then the access is granted. If the storage key in the PSW is non-zero, does not match the key in the page of storage, the operation in question does not require to change the storage, AND the Fetch-Protection bit is not on, then the instruction will be allowed.

If the non-zero storage key in the PSW does not match the key in the page and the Fetch-Protection bit is on, then access will be denied regardless of the type of action.

The logic of this is shown in Figure 7-1 on page 147.

So right at the heart of the hardware are vital protection mechanisms. We will see later that z/OS uses conventions for storage keys.

Privileged Instructions

We have also seen that not all hardware instructions are always allowed to execute. The PSW contains a bit known as the "state" bit (see Figure 7-1 on page 147). This bit is used to determine which set of instructions can be executed. When this bit is on the machine is said to be "problem program" state, and is restricted in the instructions it may execute. However, when the bit is off, the machine is said to be in "Supervisor State" and may execute all instructions. Many of the instructions which require supervisor state are designed for the use of the operating system and control such functions as,

- Setting the storage key of a page of storage.
- Data transfer between memory and some external physical device (i.e. I/O instructions).
- Loading a new PSW
- Manipulation of system clocks
- Inter processor communications

General Instructions

The general instructions (i.e. those which are not privileged) are used to perform operations on data within a specific address space and execute against data which is private to the executing application or program.

General instructions can be used in both "Supervisor State" and in "Problem Program" state.

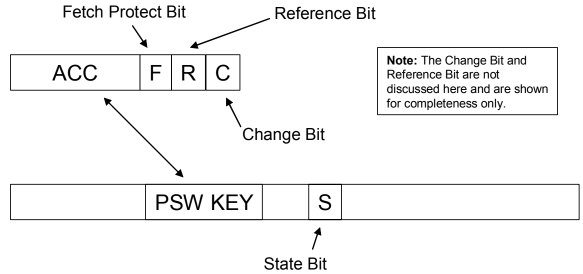


Figure 7-1

Address Translation

The system z hardware also contains an address translation capability. This enables programs to reference storage via “Virtual Addresses”. These addresses are translated to “real addresses” dynamically during instruction execution. (This is a process known as Dynamic Address Translation.)

The characteristics of this are such that multiple identical virtual addresses can be mapped to multiple real addresses, depending on the value set in Control Register 1.

This is a very powerful concept, and allows z/OS to construct a logical view of storage which it calls “Address Spaces”. See Figure 7-4 on page 149. Remember that general instructions always refer to virtual addresses.

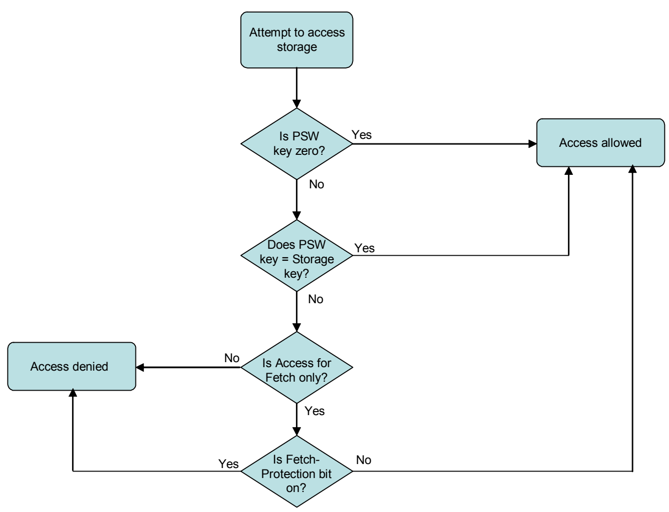


Figure 7-2

Memory Protection

As was stated above this has been a simplified description of the memory protection systems in System z and z/OS. In addition to the memory protection mechanisms described there are others. The full set of memory protection mechanisms used within z/OS include,

- Key-controlled protection (described above)
- Page protection (referenced above)
- Address space list-controlled protection
- Low address protection

Address Spaces

As we have seen the System z hardware uses 64-bit addresses. Each Address Space uses 64-bit addressing and so it starts at byte 0 and is of length 16 exabytes. (16 Exabytes is the number of bytes that can be addressed using a 64-bit address.)

However, the addressability of instructions is limited to the current address space while operating in “problem program” state. It is this restriction which provides the basic level of isolation of one set of programs and data from another set of programs and data. Figure 7-4 on page 149 shows multiple address spaces each in a different colour. Each of these address spaces can have storage with the same 64-bit storage address but is still unable to access data in other address spaces.

The z/OS Operating System refers to “address spaces”. The hardware manuals (such as the zArchitecture Principles of Operation SA22-7832) refer to address *numbers*. However as we know the hardware and software were designed to work together we understand these concepts are largely interchangeable.

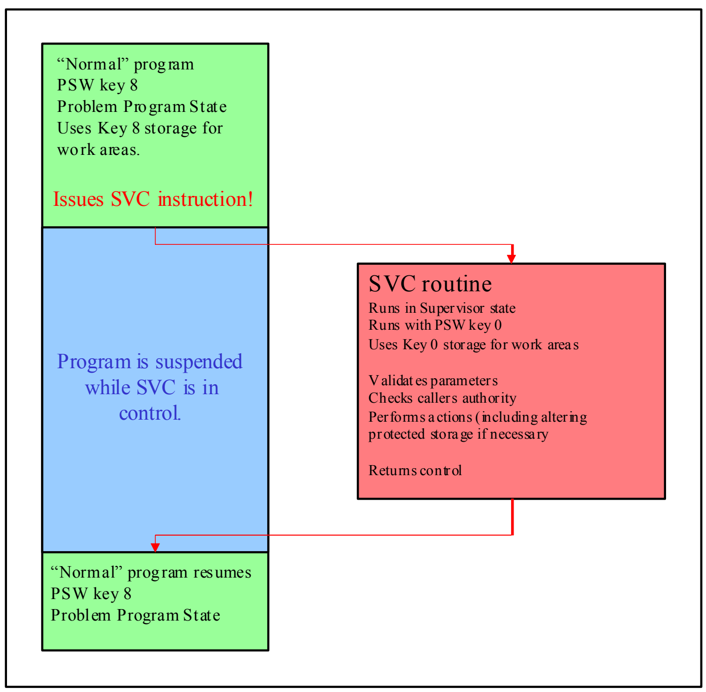


Figure 7-3

Another hardware capability mentioned in Chapter 3 is that of interrupts. One class of interrupts is the Supervisor Call (or SVC) interrupt. SVC interrupts are designed to be a mechanism for a program to call the z/OS Operating System in order to perform some operation. When an interrupt occurs the current PSW is stored in a fixed location and a new

PSW is loaded. It should be apparent that the new PSW can have the “state” bit turned off, thus switching the machine into Supervisor State.

Thus the SVC instruction is mechanism for an executing program to switch between “problem program” state and “supervisor state”. The PSW storage key can also be changed at this time. It should also be clear that once the PSW has moved into supervisor state it becomes possible to modify the Control Registers, so the “current” address space can be altered by changing the value in Control Register 1.

As the SVC is such a powerful mechanism it is important that all programs given control from SVC interrupts are very closely controlled. Most of the programs given control from SVC interrupts are part of z/OS itself, and these routines perform operations for the problem program such as opening a dataset or closing a dataset. However, z/OS allows the systems programmer to define and install other SVC routines. If this is done it is essential that these conform to very tight guidelines. See Figure 7-3 on page 148 for diagram of SVC flow of control.

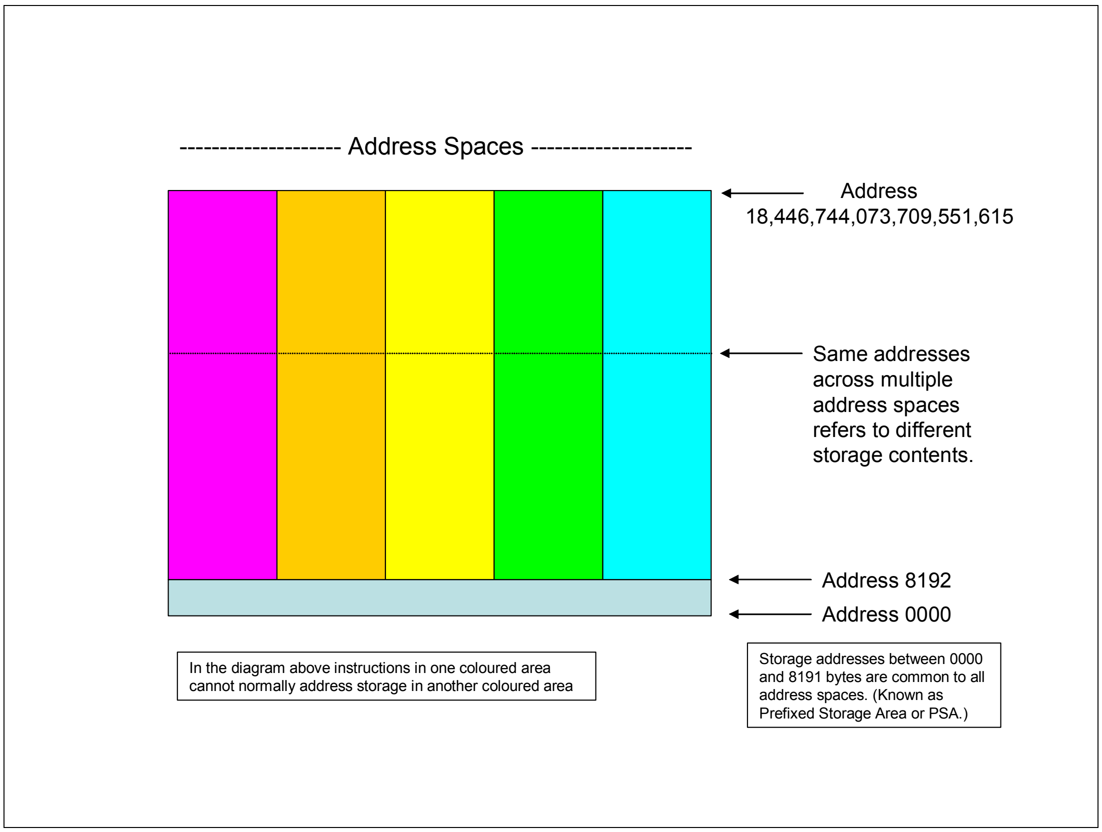


Figure 7-4

Problem Program State and Key 8

The z/OS Operating System has conventions on the use of storage keys. The conventions are shown in Table 7-1 on page 150.

Table 7-1

Key	Usage
0	If the PSW is set to 0 this is recognised by the System z hardware as a special case and the program as able to see and alter nearly all storage. The exceptions are the first 512 bytes of each of the first two 4096 blocks of storage (i.e. the PSA). These area are protected by another mechanism called "low-address protection".
1	Used by allocation and scheduler. This is primarily JES2, JES3 and the scheduler components of z/OS, but also includes the APPC scheduler address spaces. Also used by TSO/E, PSF, Network Print Facility, IP Printway and Infoprint products.
2	Websphere Application Server (WAS) but previously used for now defunct product VSPC. Also used for RACF address space.
3	Availability Manager (AVM) (AVM is a z/OS component which is not discussed in this book, but relates to
4	Reserved for z/OS (Currently unused).
5	Data management - DFSMS component of z/OS, including OAM and SMSVSAM address spaces. Also used by CICSVR component of CICS Transaction Server.
6	Communications server component of z/OS, including TCAM, VTAM, TCAS, TCP/IP, TN3270, Telnet and so on.
7	IMS, DB2, IRLM and MQ program products.
8	This is the "normal" z/OS key. z/OS starts all programs in this key unless specially instructed to use a different key.
9	Special key recognised by System z hardware. Storage in this key can be altered by programs in other keys even if the PSW key is not set to 9. This behaviour is also dependent on the settings of the fetch-protect-override control and the storage-protect-override control bits in Control register 0. This is used by the CICS Transaction Server program product.
10 to 15	Reserved for programs which run with V=R. This means that storage addresses for virtual storage match real addresses. This is now a very rare technique and requires extra parameters to be set in the system configuration (REAL= in IEASYSxx).

There is one other rule regarding keys which is important in z/OS. If the key in the PSW is any value between 0 and 7 inclusive, then z/OS treats the program as being "authorised". z/OS refers to a program executing with a key between 0 and 7 as executing in "*system key*".

A fuller definition of "authorisation" is provided later.

While a program is executing in problem program state z/OS will normally restrict it to operating in key 8. Thus the program cannot alter any storage which is not in key 8. Because it is executing in problem program state it cannot use any of the "privileged" instructions which might be used to change its own state (e.g. the Load PSW instruction LPSW, is a privileged instruction which can change the contents of the PSW. This instruction cannot be used by programs operating in problem program state.)

So effectively the normal user program is "locked in" to what it can address and what it can do. If required, it can request the operating system to perform actions on its behalf using SVC instructions.

7.2.3 Authorised Program Facility

MVS has a powerful concept know as the Authorised Program Facility. A program can be designated as “APF Authorised” if it is loaded from an APF authorised library. If this is the case, then the program has authority to execute a class of SVCs which are themselves “APF Authorised”. If a program which is not APF authorised attempts to execute these SVCs then the SVC will not perform the requested function.

One of these APF Authorised SVCs is called MODESET. This SVC will allow a program to alter its state from problem program state to supervisor state. It will also allow a program to change its PSW storage key.

Thus it should be apparent that there must be strict controls over the definitions of programs that are APF Authorised. In practice this is achieved with a combination of z/OS controls which use SAF and RACF.

When a program from an APF-authorised library is loaded into storage under certain circumstances it will be marked “APF-authorised”. When this happens a bit is set n a control block called the Job Step Control Block or JSCB. This bit is the JSCBAUTH bit. From this point on the program can execute with APF-authorisation and so may execute the MODESET SVC.

However, the JSCB is a control block created and maintained by the z/OS Operating System. It resides in key 0 protected storage. While any program operating within the address space can observe this bit (the JSCB is not fetch-protected) it cannot be changed unless the PSW key is 0.

Note the following,

- If a program is in Supervisor state it can use instructions to change its PSW storage key so that it can make itself APF-Authorised.
- If a program is running with a PSW which is less than 8 then the z/OS Operating System will allow it to use MODESET to move to Supervisor state.
- If a program is APF-authorised the z/OS Operating System will allow it to use MODESET to set any PSW key or change to Supervisor state.

Thus if any of these three conditions pertains, then there is no bar to getting the other privileges.

“Normal” Programs

Hence all normal programs run in Key 8 and problem program mode and are NOT APF-authorised. In consequence normal problem programs cannot get access to any of the privileged instructions which can alter the structure of address spaces, nor can they access data outside of the local address space without prior agreement with the operating system. The operating system is always in control.

If a program is,

- not APF-authorised
- not in system key
- not in supervisor state

then it should not be possible for the program to gain control in any of those states. I refer to such programs as *normal* programs.

“APF-authorised” and “authorised”

We have said that there is a class of programs which the z/OS Operating System treats as APF-authorised. However, we have also seen that the z/OS Operating System treats programs operating in other states as authorised.

In general the z/OS Operating System will treat as “authorised” any program which is operating in one or more of the following states,

- In Supervisor state
- With a PSW protection key between 0 and 7 inclusive
- As APF-authorised.

Some z/OS Operating System services require a specific subset of the above. Users of those services need to ensure they follow the documented requirements.

Figure 7-5 on page 153 shows how programs in one state can move to one of the other states. Note also that a program running in any non-zero system key can also issue MODESET to change key or to switch to supervisor state.

> **Note:** There is also another situation in which the z/OS Operating System will regard a program as authorised. This is when the PSW Key Mask (PKM) in control register 3 has a non-zero value for any key in the range 0-7. This PKM value is used in a complex feature of the z/OS Operating System called Cross Memory Services. Normal processes never have these values set.

How does a program library become APF-authorised?

We glossed over the definition of a APF authorised library a little above. So let us examine what defines such a program in a little more detail.

Within the z/OS Operating System is a set of libraries used for configuration purposes called PARMLIB. (In earlier versions this was limited to a single library called SYS1.PARMLIB. Now this is a collection of libraries which can be treated as “PARMLIB”.)

Within this library the set of libraries known as “APF-authorised” libraries is defined. Programs which are obtained from these libraries will be treated as “APF-authorised” as long as certain fixed conditions are met.

There are z/OS Operating System commands which can be issued by the system operator which can add to this list, or remove libraries from this list.

There is a programming interface supplied by the z/OS Operating System which enables the dynamic updating of the list of APF authorised libraries. This service is known as the CSVAPF macro.

It is also possible to define programs stored in the UNIX System Services (USS) component of z/OS as APF-authorised. In this case a special SAF check (see 7.3, “System Authorisation Facility (SAF)” on page 155 for details of SAF) is made to ensure authority to set the APF bit for the executable file.

There are also restrictions on the ability to mount new USS file systems. If this were not the case then a file system containing a program with the APF authorised attribute set could be prepared on another z/OS Operating System and then imported and mounted onto the running system. This would provide a mechanism to introduce APF authorised programs into the z/OS Operating System.

In addition very tight restrictions are placed upon authorised programs to ensure that they do not inadvertently provide the opportunity to compromise system integrity.

To understand these restrictions the reader should refer to the z/OS manual, *z/OS 1.11 MVS Authorized Assembler Services Guide,* SA22-7608-14.

It should be apparent that controls need to be placed upon,

- Who can update the PARMLIB libraries
- Who can issue the z/OS Operating System commands to alter the APF-authorised library set.
- Who is allowed to use the programming interface called CSVAPF.
- Who can define APF-authorised programs within USS.
- Who can import file systems into USS

We will see later in this chapter how this can be accomplished with SAF and RACF.

How does z/OS designate an APF-authorised program in storage?

When a program from an APF-authorised library is in storage the z/OS Operating System keeps track of this attribute using another “bit” in key 0 storage. Because it is in key 0 storage it is not possible for this to be altered by a program which is in key 8 (as is the case for normal user programs).

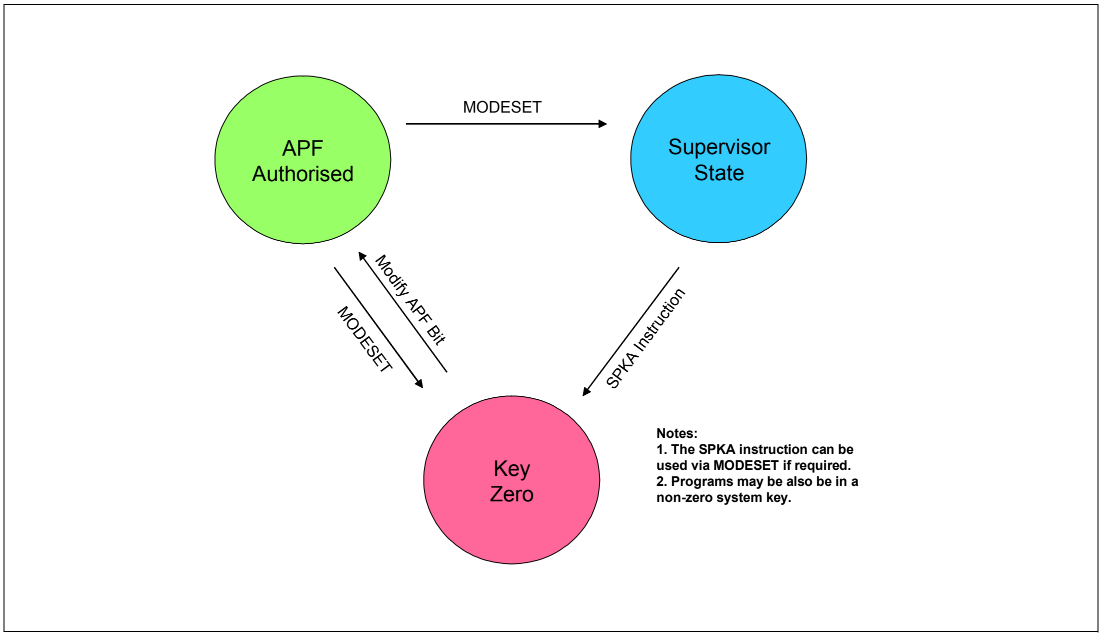


Figure 7-5

7.2.4 Summary

1. Machine instructions are divided into two classes,

   Privileged Instructions
   Problem Program Instructions

2. All System z storage (i.e. memory) is key protected using a storage key which controls the ability to observe and modify storage.
3. While executing under z/OS, user programs (i.e. programs without any special privileges) operate in PSW key 8 and Problem Program state.

4. User programs are limited to accessing only storage within a single address space.
5. SVC instructions can be used to request the z/OS Operating System to perform a function for an executing program.
6. Supervisor state programs can change their PSW storage key using privileged instructions.
7. System key programs can use the MODESET SVC to change the PSW storage key or their state (problem program or supervisor).
8. APF authorised programs can use the MODESET SVC to change their state (problem program or supervisor) or their PSW storage key.
9. The ability to determine which programs can start execution in an APF-authorised state is controlled using SAF and RACF.

The above statements summarise our explanation of the ability of the z/OS Operating System to protect itself, and hence provide a firm basis for security. While we recognise this description has been abbreviated, and does not cover all System z hardware mechanisms nor all z/OS software mechanisms, we believe it is sufficient to demonstrate how the z/OS Operating System is structured to provide integrity

7.2.5 Statement of Integrity

Much of the above discussion has been simplified. It has not discussed what are known as semi-privileged instructions, concepts such as common storage, the DAT-OFF nucleus, the z/OS Operating System dispatcher, nor covered a wealth of other mechanisms within the z/OS Operating System. Nevertheless we have shown that the z/OS Operating System can provide a secure platform for supplying complex and efficient security mechanisms.

Thus security mechanisms can be designed safe in the knowledge that the z/OS Operating System takes care of the integrity of storage, and only the operating system and its associates (i.e. APF-authorised programs) can step outside of these protection mechanisms.

IBM recognises the strength of the platform and also realises it can be used in a very wide variety of complex environments each potentially bringing its own set of problems. IBM has great confidence in the protection controls supplied by System z hardware and the z/OS Operating System software and consequently has produced a “Statement of Integrity”. This was first produced in 1973 for the forerunner of z/OS, which at the time was simply known as MVS.

The current statement of integrity can be found here,

http://www-03.ibm.com/servers/eserver/zseries/zos/racf/zos_integrity_statement.html

The body of the text is as follows,

```
IBM's commitment includes design and development practices intended to prevent
unauthorized application programs, subsystems, and users from bypassing z/OS
security – that is, to prevent them from gaining access, circumventing,
disabling, altering, or obtaining control of key z/OS system processes and
resources unless allowed by the installation. Specifically, z/OS "System
Integrity" is defined as the inability of any program not authorized by a
mechanism under the installation's control to circumvent or disable store or
fetch protection, access a resource protected by the z/OS Security Server
(RACF), or obtain control in an authorized state; that is, in supervisor state,
with a protection key less than eight (8), or Authorized Program Facility (APF)
authorized. In the event that an IBM System Integrity problem is reported, IBM
will always take action to resolve it.
```

Figure 7-6 statement of integrity

The main aim of the description above is to demonstrate the degree to which the operating system and the hardware provide a solid reliable base for security controls.

IBM has received feedback on integrity problems for many years, and continues to work with its customers to maintain the high standards of the integrity statement.

For example it is important that the identity of a user is established and then is not capable of being changed by that user or any other user of the z/OS Operating System. Thus the reponsibility for establishing user identities must be performed by the operating system or one of its components using code which runs in a privileged state.

7.3 System Authorisation Facility (SAF)

Clearly there are execution points within any code which manages security where decisions need to be made. The z/OS Operating System refers to these as Control Points. Examples of this might be,

- Determining the validity of identity credentials
- Determining the authority of an identity to access a known resource
- Determining the nature of auditing actions

The code at these decision points make use of a z/OS Operating System component known as the System Authorisation Facility. This component is used by resource managers in making decisions over access.

The System Authorisation Facility can function on its own. It does not require another product as a prerequisite. However overall security of z/OS is greatly enhanced when SAF works with another component to resolve these security questions. So SAF rarely works on its own.

Control points pass control to the SAF Router using the RACROUTE service (the name of this service shows its heritage in the early releases of RACF). SAF usually passes control to the z/OS component called RACF (Resource Access Control Facility). Alternatively SAF can be configured to pass control to some other security manager. There is third party software available which can be used in place of RACF.

RACF and each of these third party products are known as External Security Managers.

7.3.1 Validating identity credentials - Authentication

Any program which performs the role of validating identity credentials should not be capable of being subverted in any way. Consequently, for the protection of such programs they can hold data in a storage key other than key 8. This will require that they be authorised either by running in supervisor state, by being specified to run in a non-user key (i.e. not key 8) or running with APF authorisation. In practice most of these applications run with APF authorisation and then move into a system key or supervisor state as necessary.

There are several z/OS components which perform this type of operation, and several major software applications which also perform this type of operation. Examples are,

- TSO logon (a part of z/OS)
- Batch job verification (perform by JES, which is a part of z/OS)
- APPC initiators (APPC is a component of z/OS)
- Tivoli Netview
- CICS
- IMS
- Websphere Application Server (WAS)

It should be apparent that these software components will need to be trusted components in a similar manner as the z/OS Operating System itself.

The software performing this process of identity verification uses a system service called RACROUTE REQUEST=VERIFY. It will pass to this RACROUTE service items such as the following,

- Name of the userid whose identity is being established
- Associated terminal identification or Port of Entry.
- Identity verification credentials (e.g. password, password phrase, passticket)

The application also has access to other information such as the time of day, the date and other environmental information.

7.3.2 Resource Managers

Resource Managers might be independent sections of code or they may be embedded into other modules and routines. It is usual for Resource Managers to reside in authorised code. (However, in certain specific situations it is possible for the a resource manager to perform satisfactorily without the code being authorised.) The role of a resource manager is to control access to a resource. So for example, in the case of an attempt to access a data set this is performed in the OPEN SVC routine in z/OS.

Each time that a data set is accessed for the purposes of reading or writing the OPEN SVC is used first. Only after this OPEN SVC has completed without error can a program perform read and write operations to the data set. So this OPEN SVC is invoked by the program to prepare the data set for access, and also (via its resource manager) to check that the user running the program has the necessary access to the data set.

When the resource manager performs this check it uses the RACROUTE REQUEST=AUTH or RACROUTE REQUEST=FASTAUTH service. If an External Security Manager is present, this will result in a check being made which is based on the following types of questions,

- Who is attempting the access?

- What is the class of the resource?
- What is the name of the resource?
- What type of access is being requested (e.g. READ or WRITE)?

When RACROUTE passes control back to the resource manager, the resource manager can examine the Return Code and Reason Code values and can then make a decision on the basis of those codes.

It should be emphasised here that it is the resource manager that makes decisions on access following information extracted from the External Security Manager.

7.3.3 Auditing Actions

If a user has “entered the system”, that is the user has been authenticated to the system, then it is likely that local control will require this action be audited or logged.

If a user has accessed a resource, or has attempted to access a resource then it may be necessary to produce logging information to show the event has occurred.

In addition there are other situations where other actions must be logged (such as the issuing of commands which make changes to security information). In each of these cases the z/OS Operating System uses the RACROUTE REQUEST=AUDIT service to perform the logging. (Note the MVS heritage of this component is retained in its name.)

The RACROUTE service performs the logging using the z/OS Operating System’s high-performance logging process called SMF (which stands for System Management Facilities). SMF is used for recording many events that occur within the z/OS Operating System and has formal record structures defined for specific purposes. (So this is not a message log. Security messages are usually less formally structured, are designed to be read by humans and are logged separately.) SMF is discussed further in 7.5.7, “System Management Facility (SMF)”

Note: Messages are normally written to the z/OS Operating System log known as SYSLOG. They can optionally be suppressed using a capability called MPF (Message Processing Facility). In contrast SMF records are selected to be written by Record Type or Record Subtype each of which is a numeric value. In any z/OS Operating System instance which has serious security requirements the SMF security records will all be selected to be written and the Security messages will not be suppressed.

7.3.4 SAF, RACF and integrity

We have seen that SAF is given control for Authentication, Access Control and Auditing via use of the RACROUTE service This service is frequently configured to pass control to the z/OS Security Server component called RACF.

You may remember the z/OS Integrity Statement which was mentioned in 7.2.5, “Statement of Integrity” on page 154. You will notice that this statement made specific reference to RACF. Thus if some alternate External Security Manager is used the statement of integrity is not necessarily applicable.

In most of this chapter on z/OS we will assume the External Security Manager is RACF unless explicitly stated otherwise.

7.3.5 Summary

- SAF is part of the base z/OS Operating System.
- SAF usually works with an External Security Manager (ESM) such as RACF.
- SAF is invoked at many points within the z/OS Operating System where security decisions have to be made.
- SAF is invoked to perform Authentication, Resource Access control, and Auditing.
- SAF can be invoked by other software

7.4 z/OS Security Server - RACF

Resource Access Control Facility or RACF provides the tools to help each installation manage access to its resources.

RACF has been available to mainframe users since 1976, when it was made available for users of IBM's MVS operating system. Since that time RACF has increased in capability and function so that today it encompasses the following capabilities,

- Two sets of APIs. One available via the System Authorization Facility, the second via a set of callable services.
- Identification and authentication of z/OS users via various credentials, such as password, PassTicket or password-phrase. Also supports management of user identities which have no credentials.
- Identity mapping for z/OS users authenticated via X.509 V3 digital certificate or Kerberos ticket.
- User extensions in support of Kerberos.
- Resource access control for z/OS applications and components on the basis of the user's identity or on the basis of the user's membership of a group, or on the basis of security label domination.
- Management of X.509 digital certificates.
- Ability to manage some RACF services remotely via the z/OS LDAP server.
- J2EE role security model support.
- Auditing of all security events detected by RACF or reported to RACF.
- Multi-system support via a variety of methods.

With each new release RACF has provided compatibility to existing applications while growing to support new workloads, and new security constructs required for them.

7.4.1 RACF database

At the heart of RACF is a database which records information about users and resources. The entities which are stored in the database fall into the following categories.

1. **Users.** User details are recorded in profiles which include the "userid". This userid is actually the name of the profile. Typical information that is stored with the userid will be the user's name, privileges the userid has, the membership of groups for this userid, the credentials used to verify this user (one-way encrypted password, and optionally one-way encrypted passphrase), and other statistical information, such as creation date, last used date, ownership, etc.

2. **Groups.** Each userid must be a member of at least one group. Groups can have many userids connected to them. The group is a construct used to simplify the management of user access to resources. The group profile contains information about ownership, statistical information and a list of the users connected to the group.

   Optionally the group may be defined so that the list of users is not present for most users.

   By making use of the group structure we can define a set of groups such that membership of those groups constitutes a **business role**. This is a useful concept as we can place people in roles, and give the roles access to the necessary resources. This is useful insofar as it divorces the membership of a role from the resource access that role has.
3. **Resource Profiles.** These profiles represent resources. However, the name of a resource may not be exactly the same as the profile used to protect it. There is a mapping process which is used to associate a profile with a resource name. This process makes use of various types of *generic* profiles, or may make use of grouping profiles.

   Each resource profile contains an access list. This is simply a list of which users and groups can access a resource. Against each entry for a user or a group is an access level (see 7.4.6, “Access levels” below).

   There may also be a second access list, known as a conditional access list. This will contain users and/or groups as before, with access levels, but also with certain conditions attached. The conditions supported include references to consoles, system identifiers, terminals, programs, and others. Some of these conditions are valid only for certain classes of resource.
4. **Certificates.** x.509 certificates may be held in the RACF database for use in authentication mechanisms using Public Key Infrastrcuture. The keys for these certificates may be held in the RACF database also, or they may be held using ICSF services and stored in the PKDS.

> **Note:** ICSF is the Integrated Cryptographic Service Facility and is the z/OS component which manages encryption capabilities. It includes key stores, one of which is the PKDS.

The RACF database is designed so that it has both a primary and backup database. Updates are normally duplexed to ensure availability.

The RACF database may be shared between multiple instances of the z/OS Operating Systems. This sharing can be done between systems within a sysplex if required. In this case some higher performance options are available.

Updates from one RACF database may optionally be transferred to another z/OS Operating System instance and applied automatically there. The communications link uses SNA protocol. This is referred to as Remote RACF Sharing Facility or RRSF.

7.4.2 RACF Commands

In order to maintain definitions on the RACF database, RACF uses a set of commands which can be used from a TSO terminal. These commands provide the user interface to RACF for defining and maintaining profiles of the four types shown above. In addition there are some commands which must be issued from a z/OS Operating System console.

The commands fall into categories associated with those same 4 entities. However, due the special way in which RACF handles resource profiles for datasets, profiles for datasets have their own category of commands. In addition there are some other commands which are used for the configuration of the system and of RRSF.

Table 7-2 RACF commands

Profile Type	Commands
Users	ADDUSER, ALTUSER, DELUSER, LISTUSER, PASSWORD, RACMAP
Groups	ADDGROUP, ALTGROUP, DELGROUP, LISTGRP
Datasets	ADDSD, ALTDSD, DELDSD, LISTDSD,
Resource profiles	RALTER, RDEFINE, RDELETE, RLIST
Multiple profile updates	CONNECT, REMOVE, PERMIT
Certificate management	RACDCERT
Configuration commands	DISPLAY, RACLINK, RESTART, RVARY, SET, SETROPTS, SIGNOFF, STOP, TARGET
Other commands	RACPRIV, SEARCH

In order to simplify the commands z/OS supplies a panel system which can be used to issue the commands. Further simplification can be achieved using the Tivoli zSecure product.

A comprehensive guide to the commands can be found in the manual, *z/OS Security Server RACF Command Language Reference*, SA22-7687.

7.4.3 Resource Profile classes

We mentioned above that RACF treats dataset profiles slightly differently to other resources profiles. These other resource profiles are termed “General Resource Profiles”. These are divided into classes.

RACF recognises a wealth of classes describing other resources such as Tape volumes, transactions in CICS and IMS, cryptographic keys, batch jobnames, spool files, operator commands, programs, and so on.

The concept of a resource class is to enable the grouping of like resources. The meaning of a resource class name is really the responsibility of the resource manager.

Many resource classes provide access to do something and hence an access level is not meaningful. An example of this is the ACCTNUM class which control the ability to specify an account number for TSO users. For this class the only levels of access that are meaningful are None, Read and Alter. Read grants access to the resource and Alter will enable manipulation of the access list of the resource. None simply refuses access to the resource.

New resource classes are introduced with each level of z/OS.

7.4.4 RACF Segments

Since its inception the RACF database has been a repository of security related information for z/OS and its predecessors. There is frequently a need to store new security related information with each userid, or possibly with one of the other profiles types. RACF has a concept known as a “segment”. By using a new segment for a profile, a new set of structured information can be added to a profile, and subsequently used in specific applications. Examples of these segments are,

- TSO segment used to contain details related to TSO sessions used by the userid
- CICS segment used to contain details of the CICS sessions used by the userid

- OMVS segment used to contain details of the Unix System Services values associated with a userid.
- STDATA segment is used to contain details of the userid and other attributes to be associated with a started task general resource profile (class STARTED).
- CFDEF segment used to define the attributes of fields in the CFIELD class for RACF custom fields.

The concept of segments is wide and flexible, and makes RACF a useful repository of information related to security and user identity.

7.4.5 Authentication

In order to perform its role RACF needs to identify who is requesting access (i.e. the user) and needs to understand what resource is access requested for, and also in what manner the user wishes to use the resource.

We have seen that the identification of the user can be the responsibility of a z/OS component, or can be the responsibility of some other piece of trusted software. Whichever component performs the authentication the same set of services are used.

So typically when a user is authenticated a user identification string (known as a userid) is quoted, along with a set of credentials which are confirm the identity of the user. The credentials originally consisted of a password which was up to 8 bytes in length. More recent advances have enabled a password phrase to be used which can be far longer. Also a passticket can be used in some circumstances. A Passticket is a one-time password which can be generated by software. It will be valid only within a short time-frame, but can be used programatically.

> **Note:** The authentication service creates a protected control block which then identifies the user. This is called the Access Control Environment Element or ACEE. This control block cannot be altered by normal application programs because of the storage key of the memory where it resides.

When it is necessary to perform access checks the ACEE can be used to represent the user's identity. It can also be used when producing audit records for actions or attempted actions.

7.4.6 Access levels

As we have seen RACF holds the identity of users and details of profiles which map to resources. This is largely so it can provide answers to questions such as,

Is LENNIE allowed to READ resource SYS1.PARMLIB in class DATASET.

In order to provide this capability RACF must have definitions of the user (LENNIE), the class of the resource (DATASET), the name of the resource (SYS1.PARMLIB) and the level of access (READ). We have dealt with users, resources and resource classes. Let us now look at access levels.

RACF recognises several levels of access and these are implicitly defined by the resource managers. By this I mean that when an access check is performed the resource manager determines at what level the check should be made. The levels are treated hierarchically. Thus each "higher" level allows access at the lower levels. The levels (low to high) are,

- None
- Execute

- Read
- Update
- Control
- Alter

Note: These levels have the acronym NERUCA which some people find a useful acronym.

Not all these levels apply to all types of resources. Some resource managers will not use all levels. Some of the levels are designed for specific classes of resource. In particular you will see that the levels of access appear to be related to the way in which datasets are used. Given that the first class of resource ever supported by RACF was the dataset class, this is hardly surprising.

None access is exactly what it says. No access is to be allowed to the resource.

Execute access means that an object is allowed to be executed. This attribute currently applies only to resources in the program class and the dataset class.

Read access generally means that access is allowed to observe the contents of some entity but not change it.

Update access means that access can be granted to change a resource. This is particularly relevant to datasets of course.

Control access is designed specifically for a specific type of dataset known as a VSAM dataset. For VSAM datasets control access will enable updating the dataset using a lower level access mechanism called control-interval processing. However, Control access level is used for resource classes other than just dataset.

Alter access to a resource normally grants the ability to rename and possibly even to destroy the resource. Again this refers to datasets predominantly. However, it can also grant authority to alter access lists for some resources.

The access levels have a meaning which is really defined by how the resource manager uses it. Access levels of Read and Update have pretty clear meanings for datasets, but are far less meaningful for other classes of resource.

Ownership. In addition to the above, RACF recognises the concept of resource ownership. However, this is assigned by ownership of a profile which protects the resource. Hence one might more accurately describe this is profile ownership. The owner of a profile may make any changes to the profile.

The owner may also be assigned as group. In this case there are decentralized controls that may be used to perform the ownership role for the profile.

7.4.7 A RACF resource request

Figure 7-7 show the main RACF components that are used in a resource access control request.

In the diagram, item 0 shows that a user has already been authenticated by an application. When the application needs access to some resource the following steps are taken by the z/OS Operating System, and RACF.

1. Shows a request for access to a resource. Control flows to the resource manager.
2. The resource manager calls SAF by invoking the RACROUTE service.

3. SAF is configured to use RACF, so it passes the request to RACF.
4. RACF will then make checks on the basis of the name of the resource. The resource will be mapped to a profile, and the attributes of the profile interrogated. Checks will be made which refer to the user's identity and one or more groups of which he is a member. (Other checks may be made as well; see 7.4.9, "Multi-Level Security" on page 164)
5. On the basis of details about the user, system settings, and details about the resource, logging may take place. RACF calls SMF to perform the logging.
6. SMF is responsible for placing the log records onto the SMF log datasets.
7. Once RACF has completed its actions it passes return code and reason code information back to SAF.
8. SAF returns control to the Resource Manager, with its own return and reason codes, and with RACF's return and reason codes.
9. The Resource Manager then makes a decision as to whether the access is granted. In most cases the RACF return and reason codes will direct the Resource Manager as to its decision. However, in some cases (such as when RACF does not recognise the resources in question) the resource manager will make its own decision.

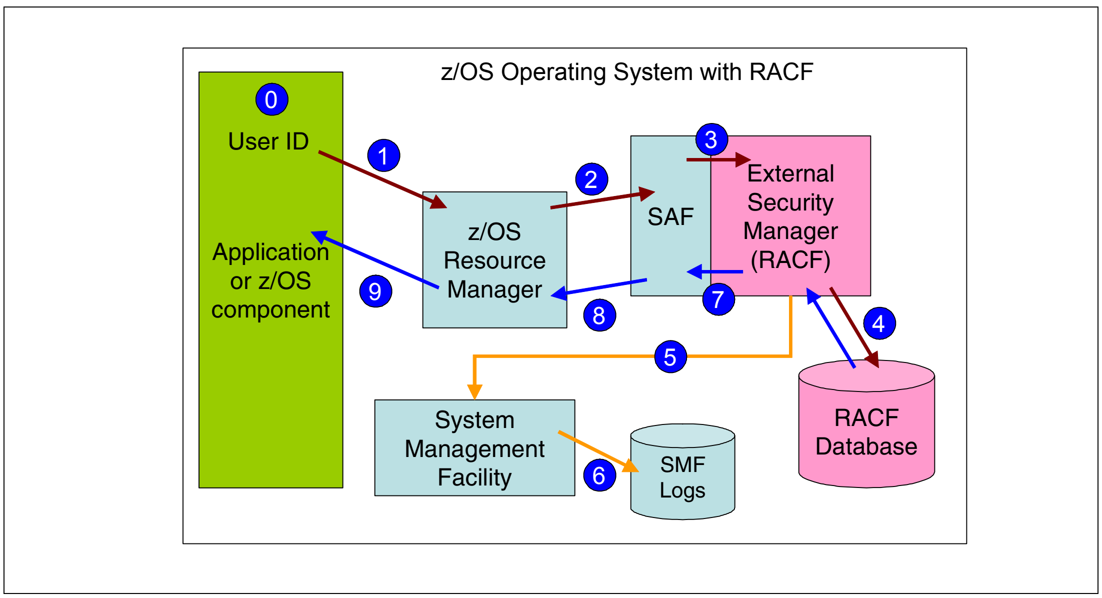


Figure 7-7

At this point the application will receive control back from the resource manager and will have resource manager codes to determine what has happened.

7.4.8 Digital Certificate Management

Public Key Infrastructure digital certificates are increasingly being used to verify identities for an ever larger set of applications. One of the more frequent applications that needs these certificates is the Secure Sockets Layer (SSL).

The management of certificates can be done using the RACF command RACDCERT. The x.509 certificates themselves can be stored within the RACF database, and can be associated with userids so that they are available when required.

X.509 certificates rely on a trust mechanism, whereby each certificate is signed by another which is trusted by multiple parties. The signature makes use of asymmetric encryption using Public and Private keys.

The RACF RACDCERT command can either be used to establish a trusted signing authority (known as a Certification Authority) or can work with an existing Certification Authority such as Verisign to have certificates signed and then imported into the RACF database.

RACDCERT can import and export keys in multiple industry standard formats such as PKCS#7 and PKCS#12. If required the keys can be kept in the ICSF key store known as the PKDS. This provides a more secure storage mechanism as the keys are held encrypted under another master key, which itself is held in tamper-proof hardware.

Each certificate may contain RSA or DSA keys, both private and public as well as many other fields used for identification and management. In addition links can be created in the RACF database so that it contains information for mapping a remote client certificate to a local z/OS RACF UserID.

Once an application has received a partner's certificate and has authenticated it (remember that RACF is not involved in the authentication process for a digital certificate) the application can pass the certificate to RACF and request a mapped-to RACF userID.

The mapping can be achieved in two ways:

1. A filtering process is used to associate a userid with fields within the certificate, such as the subject's name and the issuer's name. Based on these fields a userid can be assigned. This means that many certificates can be matched to a single userid. This method can also be used when the certificate is not installed in the RACF database, but in some other application file.

   In order to use this method the RACF classes DIGTNMAP and DIGTCRIT are used.

2. By installing a copy of the expected partners' certificates in the RACF data base, and then creating an association with the RACF user profile. RACF checks if the passed-to certificate is one of those certificates residing in the data base and returns the corresponding userID. This is always a one-to-one mapping. This is achieved using resources in the DIGTRING class

Accountability is maintained in both cases accountability is maintained as certificate information is maintained along with the ACEE in protected storage.

Further information on these techniques can be found in, *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

7.4.9 Multi-Level Security

The description given in 7.4.7, "A RACF resource request" on page 162 which describes resource access authorisation can be added to by using a concept called "security labels". Security labels is really an alternate authorisation process which is based on a hierarchy of security levels (up to 99) and a set of security categories.

The authorisation checks which are made using security labels are in addition to the existing access list controls. A security label is a combination of a single security level and one or more security categories. See Figure 7-8 on page 165.

When access lists are used for controlling resources access this is referred to as Discretionary Access Controls (DAC). However, when security labels are used RACF can operate what is known as Mandatory Access Controls (MAC). This type of access control is required by some very high-security institutions.

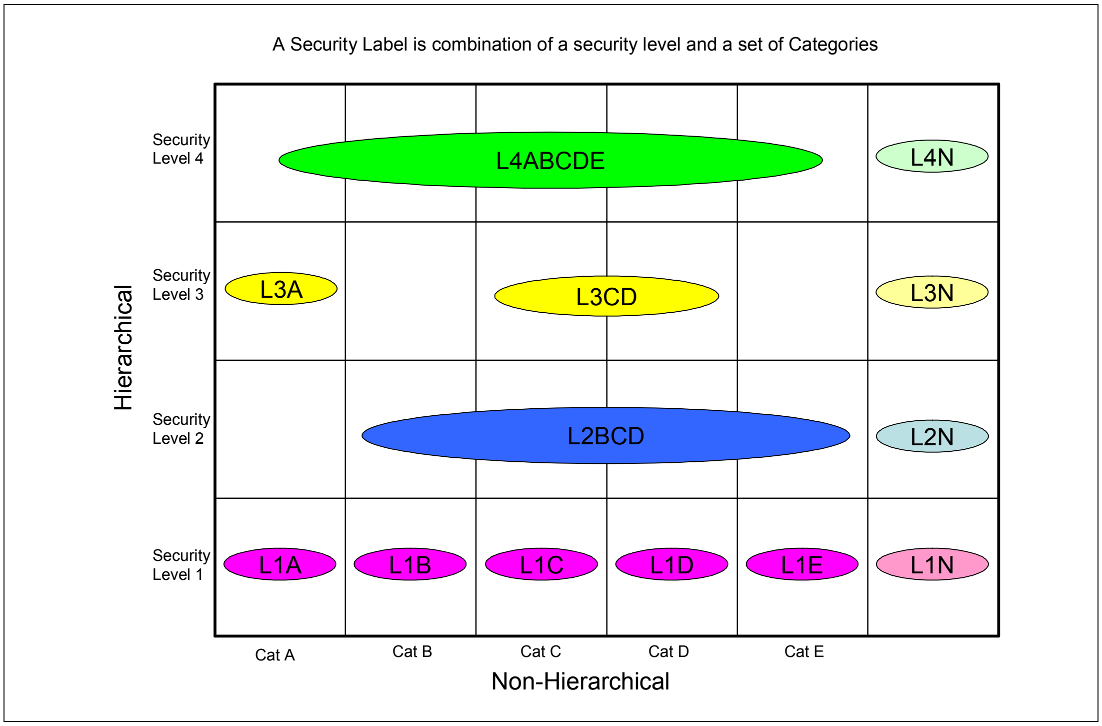


Figure 7-8

In order to make use of MAC each user and each resource must be associated with a security label. A user is normally allowed access to a resource only if the users security label “dominates” the resource’s security label.

The relationship between two security labels can be one of the following,

1. Dominance

   A security label is said to dominate anther label if the security level is equal to or higher than the second label, and,

   The set of categories in the first label is equal to or greater than the set of categories in the second label.

   So in Figure 7-8 security label L4ABCDE dominates security label L3CD.

2. Equivalence

   A security label is said to be equivalent to another security label if the security levels are the same, and the set of categories in each label is the same.

   So in Figure 7-8 the cases of equivalence would be if each security label were compared to itself.

3. Disjoint

   When there is no equivalence and neither security label dominates the other, then the labels are said to be disjoint.

   So in Figure 7-8 on page 165 security label L2BCD is said to disjoint with respect to security label L3CD.

It is also possible for RACF to be configured to insist that equivalence be established in order for an access to be granted.

The use of MLS is quite rare, but it is possible to use components of this structure with DB2 for high performance row-level security,

7.4.10 LDAP

The IBM Tivoli LDAP Server can be used with a “back-end” database called SDBM. This schema allows some RACF operations to be performed using LDAP requests over TCP/IP links. In addition there is a schema which will access the change log so that alterations can be driven from one system and commands produced to update another system.

This will allow the following set of capabilities,

1. Ability to access USER profiles and GROUP profiles and perform CONNECT and REMOVE operations between them.
2. Ability to access General Resource profiles (i.e. all resources except datasets) and perform management and access list changes.

Alternatively, RACF USER and GROUP profiles can be synchronized in different RACF data bases using an LDAP infrastructure, where each data base is hosted by a z/OS instance which has the LDAP server with the SDBM backend operating.

An LDAP client has to manage the data transfer between (or among) the z/OS instances. This is a role which can be fulfilled by the BM Tivoli Directory Integrator (ITDI).

However, note should be taken that this is *not* a full replication service.

7.4.11 RRSF

Remote RACF Sharing Facility is a mechanism designed to propagate updates from one z/OS RACF database to another.

Links are established over communication lines using SNA communications architecture although this could make use of Enterprise Extender technology to use TCP/IP links and Transport Level Security (TLS).

RRSF can be configured, so that RACF passwords, and other security information, is replicated to other RACF systems automatically and securely according to a policy. The types of information replicated can range from as little as the passwords for a select set of users, to full replication of all updates. This will include updates made by authentication services (such as userid last-used dates) as well as those made by RACF commands.

This could be used as part of a strategy for enterprise wide identity management, or could simply be used to provide a remote backup capability for the RACF database.

7.4.12 RACF administration roles

In order to maintain security each installation will have staff performing a variety of roles. These roles RACF probably fall into the following categories.

Systems Programmer

The systems programmer is normally responsible (among other duties) for the configuration of RACF in the z/OS Operating System. There are several configuration tables which must be maintained in order for RACF to function.

This systems programmer may also have responsibility for the installation of other parts of z/OS, including the maintenance of those program libraries which are APF-authorised.

It should be possible for RACF access to be configured such that no special privileges are required for the systems programmer to perform the role. However, clearly there must be the capability to update systems configuration datasets, and other system libraries.

Security Administrator

The security administrator is the role responsible for the maintenance of profiles within the RACF database. Clearly this is a highly responsible role, and many installations will wish some separation of duties. For example, user definition might be managed separately from resource access changes.

It is also possible certificate management would be treated as a specialized role.

RACF has a user authority called SPECIAL. Users who have this authority can issue all RACF commands and manipulate all RACF profiles. Hence this authority is frequently given to users who manage systems with centralized administration.

RACF also supports a decentralized administration model. This model makes use of the SPECIAL attribute at the group-connection level.

Auditor

The Auditor role will normally require the RACF AUDITOR attribute. This grants the ability to change settings which affect which audit records are written to the SMF dataset. It also grants authority to view all profiles.

The Auditor's responsibility may vary from company to company, but this is probably a technical auditor role rather than a theoretical one. <Not sure I am happy with that>

7.4.13 Summary

- We have seen that RACF is built on the secure integrity-related characteristics of the z/OS Operating System and the System z hardware capabilities.
- We have seen that RACF supplies a rich set of functions which can be used to maintain user identities across multiple z/OS systems both within a sysplex and also across sysplexes.
- We have seen that RACF is capable of handling the multiple types of security and resource entities which can be defined for the z/OS Operating System.
- We have seen that RACF is capable of managing business roles.
- We have seen that RACF can manage digital identities.
- RACF can supply support for holding security and identity related information for other products or z/OS components (such as TSO, OMVS, CICS).

Given these RACF capabilities we can now describe services which can make use of them.

7.5 z/OS Operating System components

Let us take a step back now and discuss some of the components of the z/OS Operating System.

There are a wealth of services supplied by z/OS which we discuss in this section. Some are discussed because they have significant security implications in the way in which they are used. Other are mentioned here because they contribute some security control to the z/OS Operating System or contribute to the z/OS Operating System integrity and security when used in certain configurations.

7.5.1 Sysplex

Sysplex is not really a component of the z/OS Operating System, but is a set of technologies present in hardware, operating system(s) and applications which can provide very high-performance multi-system communications and collaboration. Sysplex is IBM's clustering solution for zOS instances.

Sysplex comes in two variations; basic sysplex and parallel sysplex. The basic form makes use of inter-system communication links to perform some functions. For example it is possible to provide a single systems view of command execution and batch operation.

Parallel Sysplex provides greater collaboration opportunities. It involves each z/OS image having connections to a "coupling facility" which can contain large amounts of tabulated data or lock structures.

Sysplex requires a common time reference be established among all members of the sysplex. The coupling links from the z/OS instance hosts to the coupling facility are very fast.

We mention these characteristics of sysplex here because they have a bearing on some of the items we are to discuss. However, sysplex does not truly introduce security implications or capabilities of itself.

7.5.2 Subsystem Interface

The z/OS Operating System has a feature which provides a great deal of extensibility for features of products which need to interface with it. This is known as the Subsystem Interface. This is formal mechanism which can be used to pass messages, requests, commands and other events from one piece of software to another, or to many other software components.

A subsystem is a provider that performs one function or many functions, but does nothing unless it is requested. There can a primary subsystem and secondary subsystems. The primary subsystem will be JES2 or JES3. These are discussed later in 7.5.4, "Job Entry Subsystem (JES)" on page 169.

The subsystem concept is important because it provides several points at which extra function may be added to the z/OS Operating System by extra software.

A good example of this is that each time a message is issued to a console this is handled by a service known as a WTO (Write To Operator). Each WTO issued involves a subsystem interface call. This enables other software to receive notification that a message has been issued, which means that this can be used to trigger actions on the basis of the message. This is taken advantage of by IBM Software such as Tivoli Netview and Tivoli Systems Automation.

7.5.3 System Logger

Many components of z/OS require a high performance logging system. Examples of this are the systems message log known as SYSLOG (see , “Logging”), the Systems Management Facility (See 7.5.7, “System Management Facility (SMF)”) and the machine error recording mechanism known as LOGREC.

These can all take advantage of the System Logger to manage data in streams. The streams are subsequently off-loaded to datasets.

The System Logger is a z/OS Operating System component that provides a logging facility for z/OS components and applications. Among the advantages bestowed by the System Logger are that responsibility for the following is managed by the logger,

- Saving the data
- Retrieving the data
- archiving the data
- Expiring the data

The logger can also supply a single merged view of data which originates from multiple members of a sysplex.

At any given time the log data managed by the System Logger may reside in processor storage, in a Coupling Facility structure, on DASD, or potentially on tape. Regardless of where System Logger is currently storing a given log record, from the point of view of the exploiter (i.e. the z/OS component or the application), all the log records are kept in a single file that is a limited size.

Data location, and movement and management of data, is transparent to the exploiter. So the task of tracking where a specific piece of log data is at any given time is handled by System Logger.

By providing these capabilities using a standard interface, many applications can obtain the benefits that System Logger provides without having to develop and maintain these features themselves.

7.5.4 Job Entry Subsystem (JES)

Role of JES

Each z/OS Operating System is expected to have a JES. In formal terms JES is known as the Primary Subsystem. We have just discussed the formal status of subsystems, and the mechanisms used for subsystems to communicate with the rest of the z/OS Operating System. The JES is the “primary” subsystem and among other items is responsible for the following activities,

- Scheduling of batch jobs to batch initiators (an initiator is simply a service address space which runs batch jobs).
- Handling of spooled print output to local and remote print devices.
- Handling of output to other instances of the z/OS Operating System via a process known as NJE (Network Job Entry).
- Management of the above

Which JES?

The richness of function of the z/OS Operating System is illustrated by the fact that there are two available Job Entry Subsystems available, called confusingly JES2 and JES3. (There was a very first JES, called simply JES rather than JES1, which was used on an older mainframe operating system called OS/VS1.)

In the context of security there is no specific difference between JES2 and JES3. The characteristics are not significantly different in terms of the issues discussed below. Hence this section simply refers to JES. If there is a need to refer to one rather than the other this will be made clear.

Security Issues

One might wonder what role JES would have to play with regards to security, but there are multiple security issues here. Consider the following types of issues,

a. Each batch initiator must assume the identity of the batch job it is running, and therefore is responsible for the authentication of the userid under which the job runs.
b. If one job “submits” another job then the identify of the “submitted” job must be controlled in some way.
c. Each batch job has a name. Security controls can be put in place to control the use of those names.
d. Jobs may arrive from other z/OS instances (or from z/VM instances) via NJE, and security decisions have to be made as to whether to trust the incoming data, or determine whether security credentials can be used as supplied.
e. Data on the spooling system can be accessed by various means apart from printing it, so it is necessary to have security control points for spool access as well.

Note: In z/OS we talk about “submitting” a batch job. In other operating systems we might not have batch jobs. Instead we would have “background processes” or “services”. One of the strengths of the mainframe (which in this context I mean the IBM System z hardware with z/OS software) is that large suites of background file processing can take place in a highly automated fashion under the control of a scripting language called JCL (Job Control Language). To submit a batch job is to pass a JCL script to the z/OS Operating System for it to scheduled to be processed. The output from the batch job is frequently placed on a spooling system, for later processing by printers. In this sense it could be seen as a sort of “delayed spawning” of a process.

For each of the above situations JES has to make security decisions, and it does so by making use of the SAF interface. Ultimately the request or call is passed to RACF which passes back its results in the normal way.

Logging

There is a z/OS component which is called SYSLOG. In the early days of the operating system this was physical log (i.e. sent to a print device) which captured messages issued during the operation of the operating system. Latterly it has become a destination for messages. Those messages may be displayed on consoles or they may be used as trigger for actions. Messages are now said to be “presented”. SYSLOG is the component which gathers all messages together and this can use JES to aggregate them. (It can instead make use of the System logger to perform a similar function.)

Batch Initiators

JES works with address spaces called Initiators. These address spaces are passed work to do, from queues of work which are maintained by JES within the spool. Each piece of work is represented by a batch job, which is a script written in JCL. The first type of entry in each batch job is a JOB record (or JOB card) which identifies the name of the job, and the user identity under which it is to run.

However, in order to authenticate a user some security credentials are required. In the case of batch jobs this is done using a password which can be placed in clear text on the job card.

As this is not a particularly secure mechanism to use, some other methods are available to enable an identity to be associated with a batch job.

Identity propagation

If one address space “submits” a batch job, then JES will allow the identity for the batch job to be taken from the identity of the submitting address space. This submitting address space could be a previous batch job, it could be a TSO user, or it could be a started task, such as the Tivoli Workload Scheduler (which is an additional piece of software used for the management of suites of batch jobs which have cross dependencies.)

Identity propagation will take place simply by removing the identity and authentication credentials from the batch job (i.e. the USER and PASSWORD parameters from the JOB card).

If it is required that a userid be denied the ability to submit jobs then the userid can be defined to RACF to prevent it.

Surrogate Job Submission

If the batch job is required to run under a different identity from the submitting job, then this could be accomplished by specifying the USER and PASSWORD values on the JOB card.

To address the weakness of this security RACF can be configured to allow one userid to have surrogate authority to another userid. This will allow an address space to submit a batch job with an alternate identity.

This is achieved using resources in the RACF general resource class SURROGAT. Once the surrogate authority is in place, it is used by placing the new userid identity on the JOB card, but omitting the security credentials (i.e. placing a USER= parameter on the JOB card with no PASSWORD= parameter).

Jobnames

We saw earlier that each batch job has a Job Name. It is often necessary to have some control over these names, so that it is not possible for unauthorized persons to interfere with the streams of batch jobs that many installations run. Hence a RACF general resource class is available called JESJOBS. Using this class it is possible for the security administrator to define controls which will restrict the use of batch job names.

These controls can also be used to restrict the ability to submit batch names with a combination of jobnames and userids.

It is even possible to restrict the ability to submit any batch jobs for a specific userid, even if the password for that userid is known.

SPOOL Access

Files which are held on the JES spool can be accessed using the SDSF component of z/OS or by using TSO/E commands. This allows the viewing or other manipulation of those files. The files might be input files (which could potentially contain passwords), or could be output files.

The JES spool is a set of data sets containing spool files which are micro-managed by JES. Each of these spool files smaller files each of which can have separate security rules.

Consequently the first point is that all access to the JES datasets containing those spool files should be prevented except to the userid under which JES runs.

Then protection can be applied to individual spool files using the JESSPOOL class.

Network Job Entry (NJE)

Network Job Entry handles two functions,

a. the delivering of jobs from one operating system instance to another, and
b. the delivery of output from one operating system instance to another

> **Note:** I have carefully used the term "operating system" in this context as other operating systems than z/OS can take part in this transfer. Both z/VSE and z/VM also have components which can work with NJE on z/OS.

Let us consider a job arriving from one z/OS instance to another z/OS instance. NJE protocols refer to each operating system instance as an "NJE node".

If we are confident that the software and security environment at the sending NJE node has correctly configured the control blocks within NJE then we can accept input from an NJE node. However it is also possible to have a configuration in which the sending node is trusted to the extent that password validation is done remotely. In addition a node can be treated as untrusted. In this instance jobs arriving from that node are discarded.

Controls can also be placed on output so that it is processed or not processed.

All of these controls are implemented using the RACF general resource classes WRITE and NODES, and can be used in conjunction with the JESJOBS and SURROGAT classes above.

Summary

- The JES components of z/OS can be secured.
- While jobs can be configured to supply authentication via passwords, this can be avoided in a large number of cases.
- Each batch initiators will assume the identity of the batch job it is processing.
- Controls can be placed on the use of Job names.
- NJE jobs and output can be secured.
- Access to JES spool files can be secured.

7.5.5 UNIX System Services (USS)

Nature of USS

UNIX System Services has been a part of the z/OS Operating System for many years. It was introduced into the z/OS Operating System as an optional component of MVS, but it is now an integral component.

The existence of USS is what makes the z/OS Operating System a fully compliant UNIX system. It has file structures, commands and security which behave exactly the same as a UNIX system on any other platform.

UIDs and Userids

Each RACF userid can optionally have a UID associated with it. This is achieved by the use of a RACF Segment for USS known as the OMVS segment. Each UID is a number between 0 and 2147483647. The mappings can be one to one or can use other techniques. UID 0 has a special meaning. Along with the UID other information is stored which is used during USS sessions, for example the home directory with the file system, and the unix shell program to use for the user.

GIDs and Groups

In a similar manner USS has GIDs which can map to RACF Groups. The GIDs have values from 0 to 2147483647. However GID 0 has no special meaning.

Accessing UNIX files

When accessing UNIX files the UID is used in preference to the userid and the GID is used in preference to the group. This is because the UNIX definitions all work with UIDs rather than userids. However, as long as there is a one-to-one mapping between the set of userids and the uids then the translation can be successfully achieved. This is not always possible for userids which are mapped to uid 0.

However, for “normal” users, i.e. those that do not need access to uid 0, it is possible to put controls in place which ensure that uids and gids are not shared.

It is possible to have a uid and gid assigned on an automatic basis when they are first needed for a userid.

Note: It is also possible to assign a default OMVS segment which is used whenever a UID is required, but this no longer recommended.

UID 0

Userids which are associated with UID 0 have authority to issue any USS command and access any USS resource. UID 0 is referred to a superuser. A user with UID 0 has a very wide range of capabilities.

The normal authorities of a UID of zero can be split. The individual authorities for different functions may be granted independently using RACF profiles in the UNIXPRIV class.

For example, this enables the authority to manage files to be independent of the authority to manage processes. This is entirely sensible in a z/OS environment where the operator and the security administrator are entirely separate roles. Full details of the authorities which may be defined can be found in the manual, *z/OS Unix System Services Planning*, GA22-7800-14.

File systems

The UNIX file system in a z/OS Operating System contains a security packet for each file. This security packet can be manipulated with UNIX commands such as CHMOD to set access levels for

- User
- Group
- All others

While the security packet is associated with the file, the checking and reporting is still achieved via SAF and RACF (or some other External Security Manager).

Access Levels

USS recognises the following access levels,

- Read
- Execute
- Write
- Append

Unlike RACF, these access levels are not hierarchical. Thus, for example, it is possible to have Write access to a file but not Read access.

Access lists (ACLs)

The security packet for each file can be complex to manage as it is based on three 3 bit fields with authorities mapped to them. The three fields represent access via a uid, via a gid, or for all others. This has proved tricky to manage.

An alternative mechanism can be used which is more similar to RACF access lists insofar as each file can have an access list with users and groups represented within it, and an access level associated with each entry. This can be achieved using the USS commands getfacl and setfacl.

This way of managing file access requirements provides greater flexibility. Note that the access lists are still held in the file system with the file, and not in the RACF database.

Summary

- z/OS Operating System Unix System Services enable Unix based programs to run on z/OS.
- It provides all the necessary interfaces for unix systems, but also provides many enhancements which enable a more secure foundation for applications.
- RACF works with USS to provide access mechanisms for files.
- RACF supplies many enhancements to Unix in the management of UID 0 or Superuser, so that a restricted set of authorities can be granted.

7.5.6 Controlling Program Execution

There are some features within the z/OS Operating System which enable the controlling of programs and ensuring that it is not possible to introduce rogue versions programs in place of the required versions.

RACF Program control

Program control is a security control which is implemented inside the z/OS Operating System itself. It is used to provide very strong controls both over the execution of programs and also over the datasets that those programs can access. It also provides control over access to RACF general resources in the SERVAUTH class. This class is used to protect resources related to the use of TCP/IP.

Program control is a complex function which formally supplies several capabilities,

1. Simple controls to restrict the ability to execute specified programs by granting users either READ or NONE access through the PROGRAM class, and (when necessary) READ access to the DATASET profile that protects the load library that contains the program.
2. More complex controls that can prevent users from copying sensitive programs or viewing the contents of such programs by granting the users either EXECUTE or NONE access through the PROGRAM class, or (in some cases) EXECUTE to the DATASET profile that

protects the library that contains the program. Programs controlled in this way are referred to as execute-controlled programs.

3. Improved resistance to attacks by malicious users or programs implementing malicious functions (such as Trojan horses) in a z/OS UNIX environment when you define the BPX.DAEMON profile in the FACILITY class and require that the program execution environments for UNIX daemons and servers remain clean.
4. Program access to data sets (PADS) to allow users to have more access to data sets than they would otherwise have while running specified programs that provide restricted access to the data.
5. Program access to SERVAUTH resources to allow access to IP addresses only when executing certain programs.

The controls provided by the program control component are embedded in a part of the z/OS Operating System known as Contents Management. In order to understand how they function it is necessary to have some understanding of how individual programs are invoked and the mechanisms that are available for one program to invoke or pass control to another and also have some understanding of the control blocks involved.

In order to facilitate some of the controls, restrictions must be placed on the programs that can be loaded into some environments. Specifically, programs which are not “program controlled” can make an environment “dirty”. So any program which is not defined to RACF can make an execution environment “dirty”. If an environment is “dirty” then no controlled program can be executed within it. There is no service available to make a “dirty” environment clean. One can only destroy the “dirty” environment and create a new clean one.

An illustration of some of the capabilities of this control are shown in Figure 7-9 on page 176.

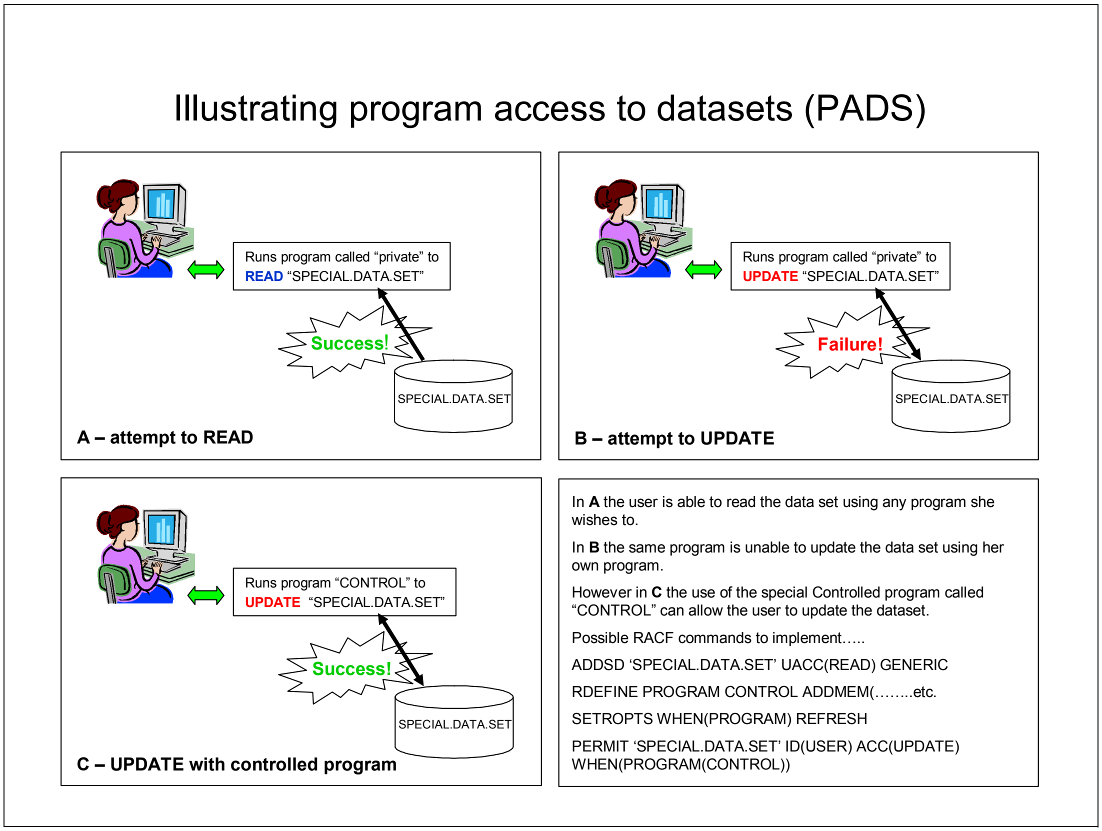


Figure 7-9

In order to implement program controls the RACF General Resource class PROGRAM is used. However, there are very important distinctions between this class and other RACF General Resource classes and administrators may need to be aware of this. Further details of the use of PROGRAM class can be found in Chapter 9 of *z/OS V1R11 Security Server RACF Security Administrator's Guide,* SA22-7683.

Signed Programs

The z/OS Operating System also has the capability to produce signed executable modules. This is designed to ensure that programs are not modified in any way prior to execution. This capability requires the use of certificates and public/private asymmetric key pairs.

The signature used for signing programs is a digital signature which is produced by a cryptographic operation between the code of the program and the private key of an asymmetric key pair. The public key of that pair is stored in the signature, and can be used to verify that the signature itself is correct. In addition the validity of the public key can be verified against the signing certificate which is held in the RACF database.

At the z/OS instance where the code is built, there is a need to provide several RACF General Resource profiles in the FACILITY class. However the code can be executed at a separate z/OS instance with a separate RACF database, as long as the signing certificate is made available.

Code signing and execution is based on RACF program protection. Therefore it is required that the signed code also be protected using RACF program controls (see , “RACF Program control” on page 174). A RACF segment called SIGVER (which is only for the PROGRAM class) is used to specify the requirements for the program with regards to signature verification. Options include,

- Allowing the load only when the signature is good and the certificate is confirmed as trusted.
- Allowing the program to load when the signature is good, but the chain of trust cannot be verified.
- Always allowing the load
- Producing audit records when the load fails
- Producing audit records only when the load fails for a bad signature
- Producing audit records for all attempted loads

Because these options can be specified for each individual profile in the PROGRAM general resource class, it is frequently possible to apply this to a whole library or to individual programs within a library.

Signed programs can only be stored in a PDSE. They cannot be stored in a PDS or in the Unix files system. Signed programs can be APF authorised or not APF authorised as required.

Summary

This is a highly complex subject, but the important issues for this publication are,

- It is possible to ensure that a program can be executed by only a restricted group of userids.
- It is possible to ensure that a dataset can be accessed at a specified access level only by using a particular program, and by a restricted group of userids.
- It is possible to ensure that TCP/IP resources can only be accessed using a particular program and by a restricted group of userids.
- The library containing the controlled program can be defined in such a manner that even if READ access is granted, and the program is then copied to a new library, the privileges will not be transferred to the new copy of the program. (This prevents the privileged program from be copied and then altered, whilst retaining its privileges, thus compromising the controls).
- Programs can be signed if required, so that further assurance can be supplied that no alterations have been made to the execution code.
- Signed programs can be supplied together with a certificate by third parties, and executed on a z/OS instance.

7.5.7 System Management Facility (SMF)

The z/OS Operating System has a major component called the System Management Facility or SMF.

SMF provides a high-performance logging process for many and various z/OS Operating System components. It is also available for use by other applications which run on z/OS (such as CICS, DB2 and so on).

Access to SMF logging capability

In general SMF is only available to specifically authorised components. It can be used for logging data only by,

1. Use of the z/OS Operating System services known as SMFWTR and SMFEWTR, which require the invoker to be “authorised” (as discussed in , ““APF-authorised” and “authorised”” on page 152), or

2. Via the USS services known as BPX1SMF and BPX4SMF. In order to use these services it is necessary to have access to a RACF general resource in the FACILITY class called BPX.SMF.

Thus use of SMF can be completely controlled by the installation.

Where is SMF used?

In a manner similar to the insertion of SAF Resource Managers, there are control points within the z/OS Operating System where SMF records are written. As has been explained above it is not possible to bypass the controls over the writing of these records. Therefore it is not possible to “spoof” SMF records. Neither is it possible to avoid SMF records from being written if installation standards and control require them to be written.

How are SMF records Organized?

SMF data is organized according to “record type”. Each of these types represents a class of event. For example Table 7-3 shows some SMF records and their usage.

It may not be immediately obvious from the description inTable 7-3 on page 178, but the data within record type 80 is produced by RACF according to the auditing options which can be system wide, or specified against each resource, resource class, or user.

Note: When Program Products such as CICS and DB2 use SMF they write records using record types which have been pre-assigned to those products.

Table 7-3

Record Type	Title of Record	Detail
00	IPL Record	Written after every SMF initialization, and includes details of central storage sizes and details of some SMF options
09	Vary DEVICE online	Written when command is issued to vary a DEVICE online to the z/OS Operating System. It identifies the device by device class, unit type and device number.
35	LOGOFF	Written when a TSO logoff command is processed.
80	Security Product Processing	Record type 80 is produced during Resource Access Control Facility (RACF) processing and Public Key Infrastructure (PKI) Services processing.
82	ICSF Record	Record type 82 is used to record information about the events and operations of Integrated Cryptographic Service Facility (ICSF). Record type 82 is written to the SMF data set at the completion of certain cryptographic functions.

Some types of record are further sub-divided into sub-types. In the table above record types 80 and 82 have further subtypes for different types of operation. For example Table 7-4 shows the subtypes associated with record type 82.

Table 7-4

Record Subtype	Detail
1	is written whenever ICSF is started.

Record Subtype	Detail
3	is written whenever there is a change in the number of available processors with the cryptographic feature
4	is written whenever ICSF handles error conditions for cryptographic feature failure (CC3, Reason Code 1) or cryptographic tampering (CC3 Reason Code 3).
5	is written whenever a change to special security mode is detected.
6	is written whenever a key part is entered via the key entry unit (KEU).
7	is written whenever a key part is entered via the key entry unit (KEU).
8	is written whenever the in-storage copy of the CKDS is refreshed.
9	is written whenever the CKDS is updated by a dynamic CKDS update service.
10	is written when a clear key part is entered for one of the PKA master keys.
11	is written when a clear key part is entered for the DES master key.
12	is written for each request and reply from calls to the CSFSPKSC service by TKE.
13	is written whenever the PKDS is updated by a dynamic PKDS update service.
14	is written when a clear key part is entered for any of the PCI Cryptographic Coprocessor master keys.
15	is written whenever a PCI Cryptographic Coprocessor retained key is created or deleted.
16	is written for each request and reply from calls to the CSFPCI service by TKE.
17	is written periodically to provide some indication of PCI Cryptographic Coprocessor usage.
18	is written when a PCI Cryptographic Coprocessor, PCI Cryptographic Accelerator, PCI X Cryptographic Coprocessor, Crypto Express2 Coprocessor, or Crypto Express2 Accelerator comes online or offline.
19	is written when a PCI X Cryptographic Coprocessor operation begins or ends.
20	is written by ICSF to record processing times for PCIXCCs and CEX2Cs.
21	is written when ICSF issues IXCJOIN to join the ICSF sysplex group or issues IXCLEAVE to leave the sysplex group.
22	is written when the Trusted Block Create Callable services are invoked.
23	is written when the token data set (TKDS) is updated

SMF data may be selected or suppressed on the basis of record type and record subtype. It will depend on the policies applicable to the management of the System z installation as to which records are to be written to the SMF datasets.

Most of the detail of the SMF records produced by the z/OS Operating System is described in the manual, z/OS V1R11 MVS System Management Facilities (SMF) SA22-7630.

However, details of record type 80 for RACF are found in the manual, *z/OS V1R11 Security Server RACF Macros and Interfaces*, SA22-7682.

Note: Details of SMF records produced by other products, such as CICS, DB2 and so on, will be provided in the documentation for that product.

How is SMF data managed by z/OS?

Once SMF data has been recorded by the auditing or recording component, the z/OS Operating System can be directed to harden the data in one of two ways. Either it is written to one of several datasets, whose names are identified in the SMF configuration options, or it may be managed using the System Logger and stored in multiple logstreams. The logstreams can be sysplex managed if desired.

During execution the SMF configuration will ensure that SMF records are not lost. This is achieved by having multiple datasets or logstreams available for writing to, and having SMF automatically switch from one to another if the active one becomes full.

Whether datasets or logstreams are used the resulting data can be large and some management processes will be needed to ensure it remains accessible for appropriate time.

It is the responsibility of the management of the System z installation to ensure that the SMF data is kept securely and for the time specified in the appropriate policy.

It may be that different policies will apply to different types of SMF records.

How is the SMF configuration specified?

SMF is configured using a member of the system PARMLIB library called SMFPRMxx. The configuration can also be modified via operator commands.

> **Note:** In a highly secure MLS environment it is possible to specify that should there be no way to record an event (due to there being no space in a dataset or logstream, or due to some other failure), the system will halt rather than continue. This is specified by the NOBUFFS(HALT) option in PARMLIB. If this situation occurs a wait state X'D0D' will be loaded.

Summary

- SMF data is used to record information about events that occur during the running of the z/OS Operating System.
- The controls over SMF records cannot be subverted or compromised by unauthorized programs without special access.
- SMF data is classified according to type and sub-type.
- SMF record type 80 contains auditing information about resource access.
- SMF data should be managed according to carefully designed policies.

7.5.8 Global Resource Serialization (GRS)

Global Resource Serialization does not on the surface have anything to do with security. This is a component of z/OS whose purpose is to supply serialization mechanisms to operating system components, and applications. the serialization can be between competing processes (tasks) within an address space, between competing address spaces within a single z/OS instance, or between competing address spaces within an entire sysplex.

The important message is that the integrity of processes and data is maintained across the entire set of z/OS instances that comprise a sysplex. Because z/OS has components to supply this serialization at all levels it can be relied on to ensure that objects such as

- data sets
- databases
- program libraries

can function in a multi-system clustered environment sharing resources without risk of corruption.

Summary

One might say that GRS provides the mechanisms by which the z/OS Operating System prevents accidental corruption of otherwise secure resources.

7.5.9 System Consoles and Commands

The z/OS Operating System sometimes needs interaction with a System Operator. Consider,

- Some applications and operating systems components may encounter unexpected conditions which need decisions on the correct course of action before continuing.
- Some types of peripheral devices need actions during normal operations, such as loading a tape onto a tape device.
- These devices will need maintenance from time to time and so may need to be removed from the system so that work can be carried out.
- Unlike some other computing platforms the z/OS Operating System is designed to be run continuously for very long times without reloading the operating system. Many companies run for many months or even years without the z/OS Operating System stopping. In these circumstances changes sometimes have to be made to the configuration of software.

Each of these situations is normally handled by using System Commands. The z/OS Operating System was originally designed to have a system operator who performs these tasks, and most enterprises still use personnel whose role is to operate the z/OS system. This is normally achieved by using System Consoles.

System Consoles

System consoles are specified in the configuration of a z/OS instance. They normally use what is known as the 3270 display console format. These consoles were originally only available to be accessed via devices attached close to the physical mainframe. However, in recent times it has become possible to have these consoles attached over wider area communication links.

System Consoles can be signed onto using RACF. The authority then associated with the console is that of the user performing the sign on.

It is also possible to have the console automatically signed on. This is especially useful in a physically secure computer room.

Commands

There is a set of System Commands which are used to perform actions. This set of commands is entirely separate from the TSO commands which are used by other users. However, System Commands may be entered by other users via various means, for example,

  - Using a system console
  - Using the SDSF component of z/OS
  - Using Netview
  - Using the CONSOLE command in TSO

It should be clear that system commands can make significant changes to the configuration of z/OS and consequently need some controls applied to them. Hence the system commands can be protected in a RACF General resource class called OPERCMDS.

The list of resources that can be protected with the OPERCMDS classes is documented in the manual, *z/OS V1R11 MVS Planning: Operations,* SA22-7601-09

There are many resources in this resource class which may be considered less risky to leave unsecured. These resources include those commands which can merely display, rather than make changes. However, the information produced by these commands might be used as part of some kind of attack.

At the other end of the scale there are commands which can be used to alter the set of libraries which are considered APF authorised. If such a command is not controlled, then the integrity of the operating system itself is exposed, and this will affect the security of the z/OS Operating System and hence of the applications running on it.

Summary

- The z/OS Operating System is frequently operated centrally by dedicated System Operators.
- The zOS Operating System has System Consoles which can issue System Commands.
- System Commands are distinct from TSO commands.
- System command can be used through interfaces other than just System Consoles.
- System Commands can be protected using RACF profiles.

7.5.10 Hardware Configuration Directory (HCD)

In order for the z/OS Operating System to read and write data to peripheral devices a configuration is required. Some of the information required overlaps with the information required by the IOCP.

The z/OS Operating System component known as Hardware Configuration Definition (HCD) provides the capability to define and manage this configuration.

It also provides the capability to provide a new definition which may be capable of being implemented dynamically; that is it can be activated without requiring the z/OS Operating System to be restarted.

HCD provides the capability to update,

- The IOCP (this is the System z configuration which applies to all LPARs on the System z processor)
- The MVSCP (this is the z/OS Operating System configuration)

Clearly there may be more than one instance of the z/OS Operating System running on an individual System z processor. Hence the HCD component can manage multiple instances.

The management of HCD and the functions it uses to make changes to the configuration has security implications. It enables new devices to be recognised by the z/OS Operating System, and possibly these devices can contain entities which masquerade as required security objects (such as RACF databases, APF libraries, and so on). Consequently it is important that HCD is secured.

The security issues are related to who is allowed to make changes to the configuration and what changes are made. The control of access to the files and data sets used is accomplished in the same manner as access to all files and data sets. The dynamic activation function is controlled via securing the System Command ACTIVATE.

Summary

- HCD can be used to make configuration changes
- Changes made by HCD could introduce fake security entities
- HCD can be secured by means of RACF resources in the DATASET class and the OPERCMDS class

7.5.11 DFSMS

Data Facility Storage Management Subsystem (DFSMS) is the z/OS Operating System component which handles data sets. Authorities for data sets is based on the data set name.

DFSMS consists of several components, some of which are optional. The formal components are shown in Table 7-5.

Table 7-5

Component	Type	Description
DFSMSdfp	Base	**Data Facility Product.** This includes DASD storage management, tape mount management, distributed data management, advanced copy services, and the object access method (which is used for tapes managed within libraries).
DFSMSdss	Optional	**Data Set Services** This supplies the Data Set Services feature which performs high-performance dumping and restoring of data, as well as copying or moving of data.
DFSMShsm	Optional	**Hierarchical Storage Manager** This performs automated migration (e.g. from DASD to tape) and automated recall. This enables the better use of disk storage. Also provides an automated backup process for changed data. Data can be restored on demand.
DFSMSrmm	Optional	**Removable Media Manager.** Provides a tape volume and dataset management facility. Tape volumes are managed within libraries. Provides shelf management, volume management and tape dataset management.
DFSMStvs	Optional	**Transactional VSAM.** This component allows the sharing of VSAM datasets across CICS and batch processes, allowing for the concurrent use of these datasets in continual operations.

DFSMSdfp

This base function is used during nearly all dataset access by the operation system and applications. Some of the major items are the following.

Creation of data sets

Data sets can be created on tapes or disks (i.e. DASD). On DASD this process consists of allocating space and creating the dataset. On tape it consists of an allocation process, but the tape labels are created at the time the dataset is opened for OUTPUT. RACF is invoked (via SAF) to determine if the user has authority to create a data set of the given name. This level of authority is provided by one of the following methods,

- having ALTER authority to the RACF dataset profile that will protect the data set once it is created

- having a connect authority of CREATE or higher to the group whose name matches the high level qualifier of the dataset
- having the dataset have a high level qualifier which matches the userid.

Renaming of data sets

This operation is supported on DASD only. This involves changing the name of an existing data set. RACF is invoked (via SAF) to check that the user making the request has the necessary authority to the existing data set and also authority to the new name, just as if he were creating a dataset of that name. In this case ALTER access will be needed to the old data set name, and authority to create (just as above) will be needed to the new data set name.

Removal (destruction) of data sets

This involves the removal of the dataset from the DASD. Removal of data sets from tapes is a logical operation only. RACF is invoked (via SAF) to ensure the user has the correct level of access to the data set.

Reading of data sets

This involves the issuing of the SVC for OPEN, which makes a data set ready for reading the data within the data set. RACF is invoked (via SAF) to ensure the user has the correct level of access to the data set. This will be READ access of course.

Updating of data sets

This too involves the issuing of the OPEN SVC, which makes a data set ready for reading and writing. RACF is invoked (via SAF) to ensure the user has the correct level of access to the data set. This will be UPDATE access in most cases, but may be CONTROL access for some special cases of VSAM data sets.

Closing of data sets

The closing of a data set involves ensuring the data set is properly configured on the tape of disk. For tapes this might involve the writing of a trailer label on the tape. For DASD datasets this will involve ensuring various fields in the data set labels are updated accordingly. There is no interface with SAF or RACF during this process.

Extension of data sets

This involves the adding of DASD space to a DASD dataset. There is no interface with SAF or RACF during this process.

Releasing space from data sets

This involves the removal of DASD space from a DASD dataset. There is no interface with SAF or RACF during this process.

Catalog Management

Data sets are normally “catalogued” on z/OS. The Catalog is special VSAM data set which contains pointers to data sets. This enables data sets to be placed on any of many DASD or tape volumes but still be capable of being located. In addition a data set can be moved from one volume to another and still be easily found.

Authorities for updating catalogs have different meanings from those for other datasets. For example, in order to catalog a dataset, UPDATE access is needed to the catalog. However, having UPDATE access will not allow the user to update any entry, as there are extra checks against the data set name being cataloged.

In addition, the catalog cannot be opened as if it were any other type of data set unless the program performing the open is APF authorised.

Advanced copy services

These services involve the replication of datasets, and the ways in which the advanced functions of Disk Storage system can be used. This includes facilities such as

- Metro Mirror (also known as synchronous Peer-to-Peer Remote Copy or synchronous PPRC)
- Global Mirror (also known as asynchronous PPRC)
- Global Copy (also known as PPRC-Extended Distance or PPRC-XD)
- Global Mirror for zSeries (also known as Extended Remote Copy or XRC)
- Flashcopy
- Concurrent Copy

The reader should refer to the following manual for details of these services, *z/OS DFSMS Advanced Copy Services* SC35-0428.

DFSMSdss

DFSMSdss provides for the following types of activities,

– De-fragmentation of free space on DASD volumes
– Positioning of data sets for performance purposes
– High performance backup and recovery.

All of these functions have security resource managers embedded within them, so that RACF is called (via SAF) whenever a resource is accessed.

There are also controls within this component so that particular functions can be allowed or disallowed. These controls are implemented by giving access to resources which are protected via RACF General resource profiles in the FACILITY class.

DFSMShsm

DFSMSdss provides for the following types of activities,

– Migration of data which has been not used in a period of time
– Automated recall of data from migration, driven by reference.
– Automated backing up data which has changed.
– On demand restore from backups.

All of these functions have security resource managers embedded within them, so that RACF is called (via SAF) whenever a resource is accessed.

DFSMShsm provides individual users control over datasets they own, while still allowing storage administrators the capability to provide these functions on their behalf.

DFSMSrmm - Tape Management

DFSMS includes an optional component called the Removable Media Manager (RMM). This component is used for managing tapes and tape datasets, as well as optical volumes. RMM enables the management of the removable media by shelves, by volumes or at the dataset level.

Tapes contain dataset that are frequently set to expire. However, a process is need to determine when those datasets expire and to then determine whether the volume on which they reside can now be re-used.

RMM makes use of a special type of catalog called a VOLCAT. The VOLCAT contains tape volumes and the shelves where they are located.

RMM can manage the full life cycle of tapes from their introduction into a tape library, to the point where they are discarded.

Tape data can be encrypted on the volume, and this can be performed by the tape device. However, control of this is *not* an RMM function.

In all of the above operations, access to the tape and the datasets on it are controlled via resource manager calls to RACF (via SAF).

Summary

- DFSMS calls RACF (via SAF) for all functions where datasets are used.
- All access to data sets is controlled by the data set name.
- Catalogs are used to keep track of the position of data sets.
- Catalogs have special access mechanisms so that their integrity cannot be compromised.

7.6 Other z/OS Components

7.6.1 z/OS Healthchecker

The formal name of the z/OS Healthchecker is the IBM Healthchecker for z/OS. The role of this software component is to identify potential problems before they cause outages. z/OS Healthchecker checks various configuration values against industry recommendations and suggests they be altered where unwise values or unwise combinations of values are detected.

Heathchecker looks at actual values in use, rather than values specified in configuration libraries. This means that any operator changes, dynamic changes or any other changes, however implemented, are taken into account.

The code which implements a given check is supplied as part of the z/OS component in question. The Healthchecker is simply a framework for implementing those checks on a regular basis, and then providing the necessary alerts. The alerts are presented in various places, and can be monitored in various ways, suitable for automation if necessary.

The healthchecker allows each instance of a z/OS Operating System to be configured so that a check is not made, or so that a different value is used. Overrides will be necessary in some installations, as it would be misleading to continually present an exception in a case where installation policy has decided that the configuration they have chosen is the one they require, regardless of other recommendations.

The main security element of the healthchecker is that RACF supplies some checks to the healthchecker.

RACF Checks

The following RACF checks are made.

RACF_GRS_RNL

This check is designed to ensure that the GRS setup for RACF is correct. It is possible that GRS can be configured incorrectly for RACF sharing in a sysplex. If this is the case then corruptions can occur in the RACF database.

RACF_ICHAUTAB_NONLPA

This check ensure that the configuration of the RACF authorised caller table is correctly configured. IBM recommends that this table is not used, and there should be no cause for its use. However, some customers still wish to use this table. The check ensures that the table is used in a secure manner.

RACF_SENSITIVE_RESOURCES

This is a very important check. This check will identify any z/OS Operating System resources which should be carefully protected and determines whether this could be a potential weakness in the security configuration.

This check will examine RACF profiles protecting PARMLIB datasets, Linklist datasets, APF authorised libraries and other datasets of similar sensitivity, and produce a report if the datasets are unprotected or have access levels higher than READ for the universal access, or for the userid '*' (which would match any userid).

If required a userid can be specified so that it is possible to determine if the userid has a higher level of access to the resources. This can be used to make basic checks on the safety of the security configuration.

RACF_IBMUSER_REVOKED

The userid IBMUSER is provided in new RACF database with a known password. It is used to ensure that a system can be configured from scratch. Once having been used it should be revoked. This check will determine if IBMUSER is revoked.

> **Note:** IBMUSER should not be deleted, as RACF initialization will re-create it if it is missing, and will then do so with a documented known password.

Other RACF Checks

Using definitions in the RACF general resource class RACFHC it is possible to define other sensitive resources and then determine the access levels to those resources. This is entirely customizable to enable each customer to define what resources they regard as sensitive.

Summary

- The healthchecker checks configuration values
- The RACF can checks provide a basic level of assurance that sensitive resources are adequately protected.
- Alerts can be produced when resources are detected which have wide levels of access.

7.6.2 Communications Server

The communications Server component of z/OS provides capability for communications using two major protocols; Systems Network Architecture (SNA) and TCP/IP. In older systems, the forerunners of z/OS used a great deal of SNA communications, but latterly the use of TCP/IP has grown very significantly. In order for z/OS to inter-operate with the web and other computing platforms TCP/IP has become an essential part of the communications protocols. Indeed a major component of the z/OS Communication Server today is the Enterprise Extender. This component allows SNA traffic to flow over a TCP/IP backbone. Thus the System z heritage in SNA communications continues to function in a world where TCP/IP has become highly pervasive.

Communications Server is a very large component of z/OS and provides a wealth of features which are described in detail in many existing Redbooks, such as, *IBM z/OS Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking* - SG24-7699

This book makes no attempt and providing the level of detail supplied in SG24-7699, but concentrates in describing the capabilities and characteristics of major security features.

The z/OS Communications server can work in a Multi-Level Security environment (see 7.4.9, "Multi-Level Security" on page 164). Many of the network configuration constructs have parameters to relate security labels to network resources.

A note about resource managers

Much of the preceding sections has discussed security in terms of identification, authentication and then resource control. This model presumes that we can provide secure identification mechanisms under the control of the z/OS Operating System.

However, with data and security constructs which originate on some other platform, we need to make use of different concepts to confirm identity. Consequently in some cases we move to using Public Key Infrastructure (PKI) to provide secure channels for communication, and to provide mechanisms for trusted identities. These are based on cryptographic concepts, which we expect the reader to be familiar with.

Policy Agent

The z/OS Communications Server provides a feature known as the Policy Agent. This is a flexible construct which is used to define security policy with regards to the management of TCP/IP traffic flowing both into and out of the z/OS environment.

Using the policy agent we can define how IP packets of data is managed and filtered. This applies to both TCP and UDP protocols. We can also use Policy agent to specify that certain traffic be encrypted using industry standard protocols such as TLS, or provide requirements for the construction of Virtual Private Networks (VPNs).

Policy Agent can provide a policy for the following components of the z/OS Communications Server,

- Qos. Quality of Service
- IDS. Intrusion Detection Services
- AT-TLS. Application Transparent - Transport Layer Security
- IP filtering
- IPSec static VPNs
- IPSec dynamic VPNs
- PBR. Policy Based Routing

So the Policy agent is not a security feature in itself; however it is used to provide security definitions and requirements which are subsequently implemented by software components within the TCP/IP stack.

The Policy Agent can also be used as a Central Policy Server, so as to provide Policy Based networking controls for an enterprise.

SAF checks

Before we move on to discuss the security features supplied by the policy agent lets take a little time to demonstrate those security features which are managed by SAF and RACF.

Many of the checks of this nature used by the z/OS Communications Server use the RACF general resource class SERVAUTH. Resources in this class enable protection of the use of the TCP/IP stack. So for example a check is made against the resource name, EZB.STACKACCESS.*sysname.tcpipname*

where "*sysname*" is the name of the z/OS Operating System instance where TCP/IP is running, and "*tcpipname*" is the name of the job or started task in which TCP/IP is executing.

Note: Remember that like other systems, each z/OS instance can have multiple TCP/IP stacks running. Such instance of z/OS would be described as multi-homed.

Within the TCP/IP configuration is a set of definitions describing the TCP/IP networks which are to be accessed. These are described within the TCP/IP configuration using a statement called the NETACCESS statement. The construction of the NETACCESS definitions allows for the specification of a name to be associated with each subnet. Using this name a SERVAUTH resource can be constructed which is then used to check access to the subnet. We then have a resource such as,

EZB.NETACCESS.*sysname.tcpipname.security_zonename*

where "*sysname*" is the name of the z/OS Operating System instance where TCP/IP is running, "*tcpipname*" is the name of the job or started task in which TCP/IP is executing, and "*security_zonename*" is the name associated with the subnet.

TCP/IP ports can also be secured via further definitions in the TCP/IP configuration (PORTRANGE statements) and further resources within the SERVAUTH class. The port definitions can be associated with both TCP ports and UDP ports.

There are further controls over the use of broadcast functions.

Further still there are fine-grained controls over the use of advanced IPV6 capabilities.

We have seen that z/OS Operating System commands can be secured using SAF and RACF. Many TCP/IP commands can be secured in this way, as well as commands used to make changes to SNA networking capabilities.

TCP/IP TSO commands such as NETSTAT and its facilities can be secured using further SERVAUTH resource names, such as,

EZB.NETSTAT.*sysname.tcpipname.option*

where "*sysname*" is the name of the z/OS Operating System instance where TCP/IP is running, "*tcpipname*" is the name of the job or started task in which TCP/IP is executing, and "*option*" is the NETSTAT option in question.

IP Filtering

When TCP/IP packets arrive at the z/OS Communications Server it is important to be able to filter them, so as to not allow several forms of attack. The z/OS Communications Server provides filtering at the IP layer for both IPv4 and IPv6.

IP filters are rules defined to either permit or discard packets. Filtering can be based on source or destination IP addresses, protocol, source or destination port, direction of flow, or time.

IP filtering can control traffic which is being routed, or can control access to the host endpoint. Even if an external firewall is providing filtering of traffic, the z/OS Communications Server can provide further filtering, or a second line of defence.

IP filtering is actually referred to as IPSec filtering within the z/OS Communications Server.

IPSec

IPSec is a suite of protocols and standards defined by the Internet Engineering Task Force (IETF) to provide an open architecture for security at the IP layer of TCP/IP. It is frequently used as the protocol to create a Virtual Private Network or VPN.

A VPN is a Virtual Private Network. This is a communications path established between two (or more) points in which all traffic which flow is encrypted.

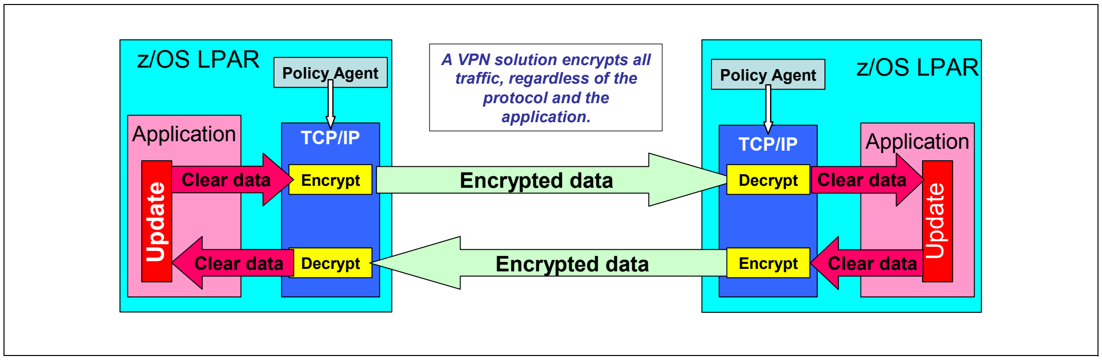


Figure 7-10

As shown in Figure 7-10 IPSec works at the IP networking layer the security it provides can be provided without any necessity to modify applications. It can also be used to protect both TCP and UDP traffic. This enables it to be used to protect SNA traffic flowing over Enterprise Extender links.

The IPSec support in z/OS Communications Server include both static VPNs and dynamic VPNs.

Static VPNs use a shared secret symmetric encryption key and encrypt data using that key. However dynamic VPNs use Public Key Infrastructure to establish a level of trust between the servers in question and then use that to derive a new encryption key dynamically.

Dynamic VPNs are more commonly used than static VPNs. These networks provide,

- Data origin authentication
- Data integrity
- Replay protection

IPSec includes two protocols called Authenticated Header (AH) and Encrypted Payload Security (ESP). The z/OS Communications Server supports both of these protocols.

Authenticated Header provides a mechanism to authenticate an entire IP packet. A hash of the entire packet is produced and encrypted using a known key. Each header also contains a sequence number. The header itself is not encrypted so that routing can continue.

Encrypted Payload Security ensures that the data portion of the IP packet is encrypted, so that it is of no use to anyone snooping on the communications session.

IPSec also includes the protocol known as Internet Key Exchange. This is the handshake protocol used to establish IPSec sessions and to change encryption keys on a regular basis. z/OS Communications Server performs this function using an IKE daemon.

AT-TLS

We have not yet discussed Secure Sockets Layer programming capabilities because that is a function of the Cryptographic Interfaces of z/OS which is discussed in the next section.

However, z/OS Communications Server has an implementation of Transport Layer Security (TLS) which is functionally very equivalent to, and can be made compatible with, Secure Sockets Layer 3.0 (SSL 3). This component is called Application Transparent Transport Layer Security or AT-TLS.

Secure Sockets Layer is an applications protocol. It requires each application which needs to secure its communications to be coded to make the appropriate SSL calls in the sequence of operations comprising the communications programming. This means that applying security to an already running application can be time consuming and costly. AT-TLS attempts to resolve this issue by providing a mechanism to make existing TCP/IP applications make use of TLS.

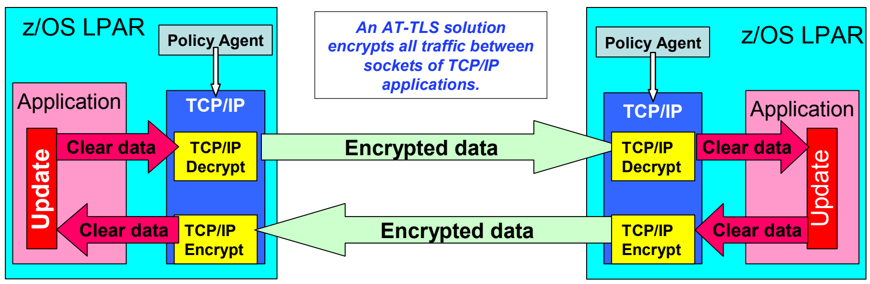


Figure 7-11

The Policy Agent is used to provide specifications for AT-TLS so that the process of encryption is indistinguishable from the encryption supplied by TLS. Thus while Figure 7-11 on page 191 shows two z/OS instance communicating with one another, either platform could be some other server which is running some other operating system.

IDS

Intrusion Detection Services (IDS) is a mechanism that inspects inbound and outbound network activity and identifies patterns of activity which may indicate a network or system attack is taking place. IDS is capable of inspecting packets which may be maliciously formed in an attempt to fool simple firewall filtering rules.

The z/OS Communications Server component already silently protects against many types of attack. IDS provides further configurable capability to protect against specific types of attack and to gather documentation when it occurs.

If such an attack is detected, IDS can make policy based decisions to respond to that activity. Possible responses to attack might be,

- Event gathering
- Statistics gathering
- Packet tracing
- Discarding of packets

IDS makes use of the Policy Agent to configure and store its rules.

IDS has three forms of detection,

- Scan detection
- Attack detection

– Traffic regulation

Scan detection

Scan detection is designed to determine if a malicious person is scanning ports. This type of detection relies on the scan originating from the same IP address. Hence a rule can be established which is triggered when multiple ports are accessed from a single IP address. The parameters regarding how fast the scan takes place, the acceptable IP addresses from which scanning can take place, the protocols and ports involved, and the actions to take are all specified in the policy.

Attack detection

Attack detection is designed to detect deliberate attacks. Such attacks can be *active* or *passive*. In this context an *active* attack is one which attempts to alter system behaviour, whereas a *passive* attack is designed to learn about the system.

Attack policies within IDS are geared towards active attacks and can threats as shown in Table 7-6.

Table 7-6

Category	Attack description	Actions
Malformed packets	There are numerous attacks designed to crash a system's protocol stack by providing incorrect partial header information. The source IP address is rarely reliable for this type of attack.	► TCP/IP stack: Always discards malformed packets. ► IDS policy: Can provide notification.
Inbound fragment restrictions	Many attacks are the result of fragment overlays in the IP or transport header. This support allows you to protect your system against future attacks by detecting fragmentation in the first 256 bytes of a packet.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
IP protocol restrictions	There are 256 valid IP protocols. Only a few are in common usage today. This support allows you to protect your system against future attacks by prohibiting those protocols that you are not actively supporting.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
IP option restriction	There are 256 valid IP options, with only a small number currently in use. This support allows you to prevent misuse of options that you are not intentionally using. Checking for restricted IP options is performed on all inbound packets, even those forwarded to another system.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.

Category	Attack description	Actions
UDP perpetual echo	Some UDP applications unconditionally respond to every datagram received. In some cases, such as Echo, CharGen, or TimeOfDay, this is a useful network management or network diagnosis tool. In other cases, it might be polite application behavior to send error messages in response to incorrectly formed requests. If a datagram is inserted into the network with one of these applications as the destination and another of these applications spoofed as the source, the two applications will respond to each other continually. Each inserted datagram will result in another perpetual echo conversation between them. This support allows you to identify the application ports that exhibit this behavior.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause packet to be discarded.
ICMP redirect restrictions	ICMP redirect packets can be used to modify your routing tables.	► TCP/IP stack: Will discard ICMP redirects if IGNOREREDIRECT is coded in the tcpip.profile. ► IDS policy: Can provide notification and disable redirects (this can optionally be coded as a parameter in the tcpip.profile).
Outbound raw restrictions	Most network attacks require the ability to craft packets that would not normally be built by a proper protocol stack implementation. This support allows you to detect and prevent many of these crafting attempts so that your system is not used as the source of attacks. As part of this checking, you can restrict the IP protocols allowed in an outbound RAW packet. It is recommended that you restrict the TCP protocol on the outbound raw rule.	► TCP/IP stack: No default action. ► IDS policy: Can provide notification and cause the packet to be discarded.
TCP SYNflood	One common denial of service attack is to flood a server with connection requests from invalid or nonexistent source IP addresses. The intent is to use up the available slots for connection requests and thereby deny legitimate access from completing.	► TCP/IP stack: Provides internal protection against SYN attack. ► IDS policy: Can provide notification.

TRMD

The Traffic Regulation Management Daemon (TRMD) is a daemon supplied as part of the z/OS Communications Server. It is used to capture logging information from the Intrusion Detection Services and IPSec components of the z/OS Communications Server.

This component includes a utility program called TRMDSTAT which can produce reports from the logged records.

- The following types of report can be produced,
- Overall summary of logged connection events
- IDS summary of logged events
- Logged connection events
- Logged intrusion as defined in the ATTACK policy
- Logged intrusions as defined in the TCP policy

– Logged intrusions as defined in the UDP policy

SNA

Systems Network Architecture is an IBM networking protocol which has been in use over many years, but which has lately been usurped by the far more prevalent use of TCP/IP which flows over public networks.

SNA was designed at a time when most enterprises had private networks, and so many of the issues which TCP/IP has to deal with were not present.

In more recent times SNA is still used a great deal within enterprises, but when used over longer distances, it is frequently encapsulated using technologies such as Enterprise Extender.

SNA security can be divided into two distinct areas; Subarea security, and APPN.

Subarea

The networks that contain genuine SNA traffic are usually not public - or at least are considered to be secure networks, again reducing the security requirements of SNA traffic. In the event that security measures are considered appropriate for SNA traffic, the following features can be used:

- LU authentication

  When using an encrypted session, LU authentication can be performed to certify that the key used by each endpoint is the same. However, if authentication is not requested, the mismatch of the session keys prevents any data from being unencrypted at either end. This encryption is symmetric encryption and requires the user to organize the sharing of the static encryption key.

- Message authentication

  An additional code can be sent with all SNA data messages. This code can be used to verify that the message has not been altered in transit. Data encryption Data between LUs can be encrypted to ensure confidentiality between sessions.

APPN security

It is reasonable to state that the majority of APPN traffic is now encapsulated when it is on the network using UDP/IP (i.e. using Enterprise Extender). In other words, SNA has evolved from being a network architecture. Instead, it is being transformed into a set of protocols that define an architecture for interapplication communications. From an IP standpoint, APPN is an application architecture, not a networking architecture. When APPN traffic is carried over UDP/IP, standard IP-based security methods can be used, such as VPN tunnels.

For APPN traffic that is not traveling over an IP network, or if IP-based security measures are not considered appropriate or adequate, APPN has the following features available:

- Authentication

  The identity of a session partner can be confirmed by VTAM session level services or at the application program level (user identification).

- Encryption

  An APPN session can be defined to require that data be encrypted between LUs.

Summary

- The z/OS Communications server is a highly configurable component, which has a very rich set of security features.

- Access to networking resources from within z/OS can be controlled at a very granular level
- A comprehensive set of security features is available to filter incoming network data, and to identify, document and repel attacks.

7.6.3 Cryptographic Services

z/OS includes several components which supply cryptographic services, some of which are interdependent. However, there are also some cryptographic services which are supplied by other componentry. For example, RACF has some cryptographic capabilities in its management of Digital Certificates.

ICSF

The Integrated Cryptographic Support Facility is the component which can supply highly secure cryptographic operations on z/OS. It does this by leveraging the System z hardware component called the CryptoExpress2. As has been seen in <chapter 4 somewhere> the CryptoExpress2 feature is capable of performing highly secure cryptographic operations

ICSF fulfills three main functions,

Firstly it provides a capability to manage the status of the CryptoExpress2 devices. This includes their physical status (online or offline) with respect to the CryptoExpress2 domain assigned to the current LPAR, as well as the status of the master keys within that domain.

Secondly, ICSF supplies keystores. There are three keystores, known as the CKDS, PKDS and TKDS. The CKDS is a keystore of symmetric keys in which the entries are encrypted using the symmetric master key. The PKDS contains asymmetric keys which are encrypted using the asymmetric master key. The TKDS contains PKCS#11 tokens.

> **Note:** Remember that all Master keys are held in the confines of the tamper-proof CryptoExpress2 device.

The third function of ICSF is that it supplies APIs for calling the cryptographic services.

A diagram showing the main components of ICSF is shown in Figure 7-12

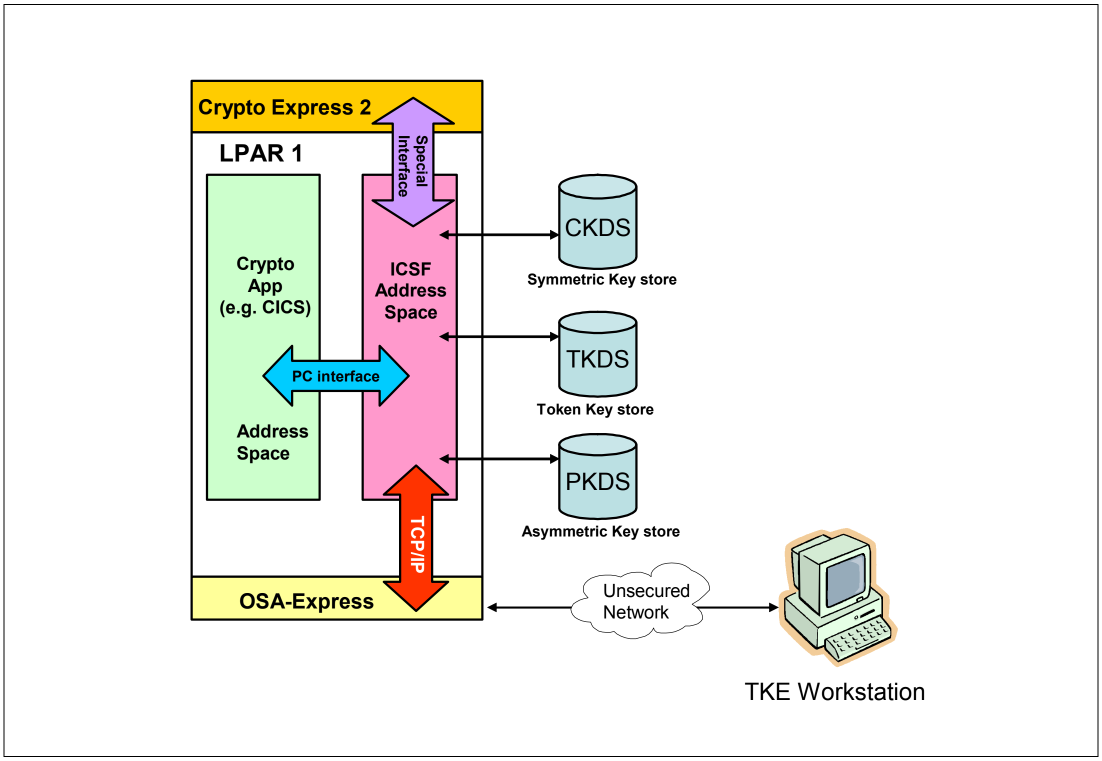


Figure 7-12

Also shown in Figure 7-12 is the TKE Trusted Key Entry workstation. This hardware component can be used for highly secure key entry of master keys and operational keys.

Management of CryptoExpress2

Each CryptoExpress2 device has 16 domains which would normally each be associated with a distinct LPAR. Thus more than one crypto device domain can be assigned to a given z/OS system, but only one domain is allowed from each CryptoExpress2 device, and if two domains of different numbers are assigned then ICSF has to be instructed which domain to use. If the same domain number is assigned from two distinct CryptoExpress2 devices, then the ICSF instance will be able to access both device concurrently.

Only one instance of ICSF can run on each z/OS instance.

The master keys for each domain of a CryptoExpress2 can be assigned using the ICSF panels or by using a TKE workstation. For high security multi-domain environments the TKE is recommended.

The keystores

The three keystores are all VSAM datasets, and all contain cryptographic components. All of the keystores are VSAM KSDS data sets. They are keyed on a combination of a 64-byte label and an 8 bytes key type. These labels can be used in cryptographic APIs, so that actual key values (whether encrypted or not) are never required to be used in programming. This level of indirection is very useful, and contributes to the ease of changing master keys when required.

Each of the keystore datasets is accessed via a separate z/OS dataspace which is create and then loaded at ICSF initialization time with the contents of the keystore. As keys are updated via the APIs the dataspace is updated to reflect those updates, as well as the keystore itself. There are also utilities to refresh the dataspace from the contents of the keystore dataset.

In a z/OS Sysplex environment each of the keystores can be independently configured to make use of XCF services to apply updates from an ICSF instance running on a single member of the sysplex to all the other instances of ICSF running on the other members of the sysplex. This provides consistency of access to the keys.

The *CKDS* contains symmetric keys, which are used either for some form of DES (single DES, 2 key triple DES, or 3 key triple DES) encryption, or some form of AES encryption. The keys can either be held in clear or can be held encrypted under a master key. Master keys are held within the bounds of the tamper-proof hardware of the CryptoExpress2 device. The Key store concept is shown in Figure 7-13.

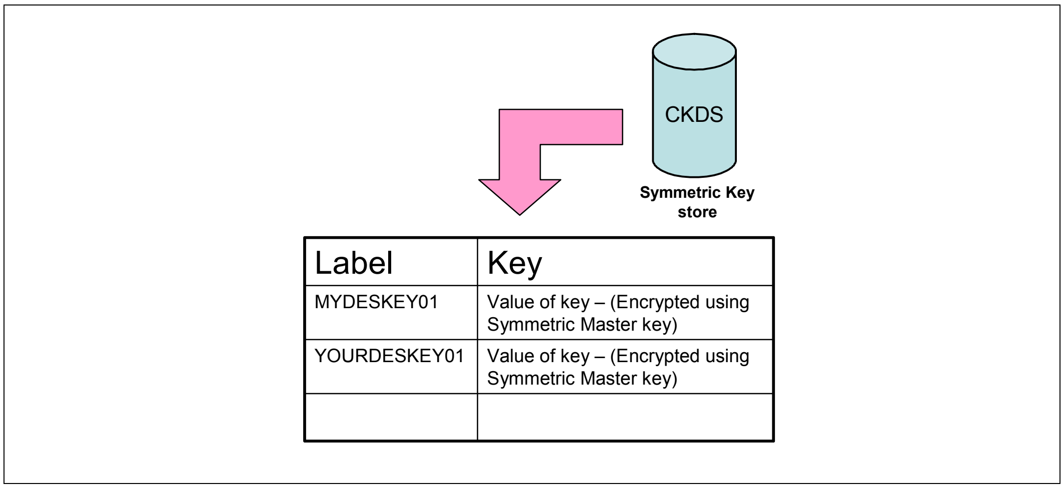


Figure 7-13

If the keys are encrypted then the processes using them are referred to as secure key processing. The clear value of the secure key is never exposed outside of the tamper-proof hardware. Secure DES keys are encrypted under a 16-byte triple DES master key, which can be unique for this domain. Secure AES keys are encrypted under a 32 byte AES master key, which can be unique for this domain.

ICSF also supplies a batch update program for the CKDS. This program is called CSFKGUP and is capable of generating and storing many key types. However, it does not support all possible key types. Also note that updates performed by CSFKGUP will not update the dataspace associated with the CKDS. A separate update utility must be run.

The *PKDS* holds asymmetric keys. Each record can either be a single public key or a public key and a private key. If the private key is stored then this is held encrypted under the asymmetric master key, which is a 24 bytes DES key. Asymmetric keys can be RSA keys under 4096 bits in length.

The *TKDS* holds PKCS#11 tokens and token objects. It may also contain private keys associated with those tokens and objects. These token may be used to map to tokens such as those used as an analogue to smart tokens used on other single-user operating systems.

The ICSF APIs

The APIs supplied by ICSF can used from many different languages including any languages supported by z/OS Language Environment®, System z Assembler, and REXX.

This includes services to,

– Create and delete symmetric and asymmetric keys of multiple types

- – Import and export keys under other keys
- – Generate and verify Message Authentication Codes (MACs)
- – Provide hashing services using a many industry standard mechanisms
- – Provide Digital signatures
- – Provide for exporting and importing symmetric keys under asymmetric keys
- – Encrypt and decrypt data using DES, double length triple DES, triple length triple DES and various lengths of keys for AES.
- – Generate and manipulate PINs for various financial transactions.
- – Provide a mechanism for remote key loading of ATM keys
- – Support SSL services
- – Provide custom extensions

The services supplied by ICSF fall into three broad categories in terms of how they are implemented.

- The majority are managed by calls to the secure cryptographic functions of the CryptoExpress2.
- A second group is handled by the use of machine instructions on the CPACF chip.
- The remainder are handled via software.

There are in excess of 100 APIs available in total. The functions and capabilities get larger with each release of ICSF.

Many of the services supplied by ICSF and the CryptoExpress2 processor conform to the Common Cryptographic Architecture developed by IBM in the 1990s. Part of this architecture provides a concept known as “control vectors”. This provides a mechanism to ensure that keys are used for defined functions only. For example a key used for encrypting PIN blocks in financial processing should not be used for decrypting data. As we have said, keys are held in the key store datasets encrypted under master keys. However, before the master key is sued for this encryption it is modified (using a logical process called exclusive OR processing). Thus in order for the services in the CryptoExpress2 to make use of this key, the decryption of the key within the CryptoExpress2 must take into account the modifications to the master key. This key separation of duties is major benefit of CCA. The logic of the control vectors is shown in Figure 7-14 on page 198.

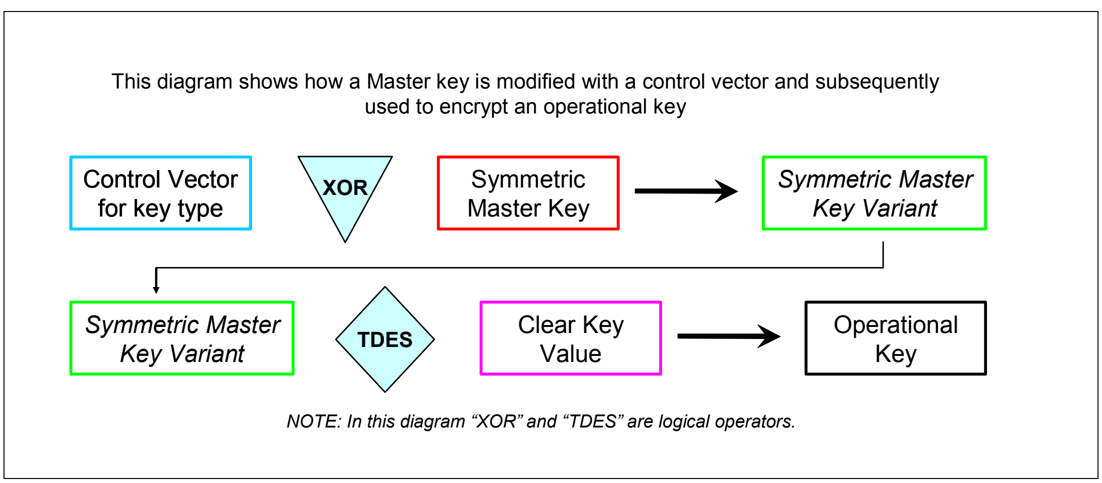


Figure 7-14

SAF interfaces and controls

ICSF makes extensive use of SAF calls to authorize use of its services and the keys in the keystores.

Most of the services (i.e. the APIs) call SAF prior to executing the function. Thus the caller of the API will require access to the function in the RACF general resource class CSFSERV. This is not called for every service. In particular it is not called for those services which are supplied via CPACF. These operations supplied by those services can be performed without using ICSF, so SAF calls do not make sense. Access checks are performed at the READ level.

ICSF will also ensure that any attempt to use a key also results in a call to RACF (via SAF). These checks are made in the RACF general resource class CSFKEYS. So any attempt to use a key will require at least READ access to a profile in the CSFKEYS class.

There are options to enhance these controls via a set of profiles in the RACF general resource class XFACILIT. These extra controls are called keystore policy controls and can be used to enable several enhancements to the basic security including,

- Ability to ensure that access to a key also checks access to any key in the keystore which has the same clear key value as the key being used.
- Ability to prevent duplicate token within each keystore. This will prevent keys with the same clear key value from being store into a single keystore under different labels. This applies to the CKDS and PKDS.
- Ability to require higher levels of access to a key in order to make changes to that key.
- Ability to provide a “default” level of protection to any keys which are not defined in the CKDS or PKDS, but which are used by ICSF.

A further level of protection is available using resources in the RACF general resource class XCSFKEY. These resources can be used to allow or prohibit the exporting of keys using the CSNDSYX API. The key labels in question will then be defined in the XCSFKEY class and UPDATE access would be needed to allow the exporting via CSNDSYX.

OCSF

OCSF is the z/OS implementation of the Common Data Security Architecture (CDSA) that was developed by the Open Group

The CDSA architecture is made up of four major layers. Each builds on the services of the layer below it:

- The application domain, which is the highest level, calls upon a standard set of system security services like SSL, S/MIME, IPSEC, and so on.
- The system services layer calls upon the OCSF framework to invoke the specific security services.
- The OCSF framework provides a standard set of APIs and relates them to the installed service provider modules via a registry construct.
- Finally, the service provider modules either provide the security services required or interface with other system elements to provide the services required.

System SSL

System Secure Sockets Layer is a set of libraries which are available to applications written in C and C++ which enable the application to use TCP/IP services with an encryption capability. SSL 3.0 and its IETF compatible version TLS (Transport Layer Security) can be used by applications to encrypt TCP packets flowing between applications on z/OS and some other platform (which could of course be another z/OS instance).

Figure 7-15 on page 200 shows a logical view of SSL (or TLS) as used by two instances of z/OS.

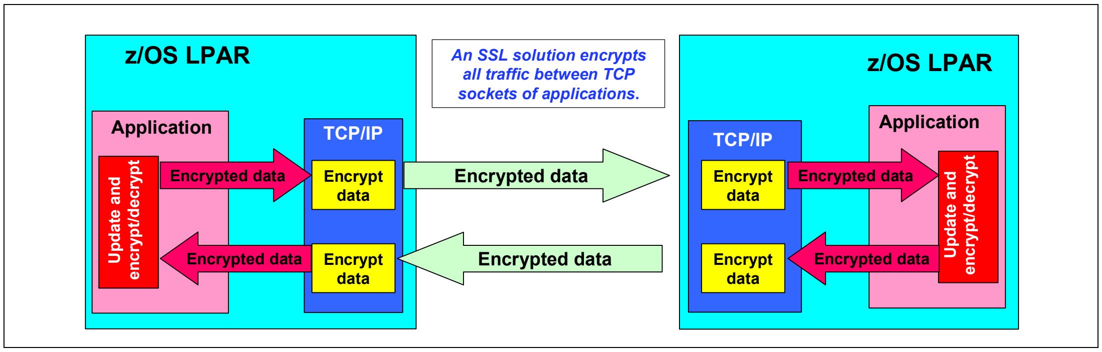


Figure 7-15

z/OS PKI Services

z/OS PKI Services is a base component of z/OS that implements PKIX Certificate Authority functions. A Certificate Authority (CA) operates at the core of a so-called Public Key Infrastructure (PKI).

All details about the implementation, setup, and use of the z/OS PKI Services can be found in *z/OS Cryptographic Services PKI Services Guide and Reference*, SA22-7693.

The Public Key Infrastructure (PKI)

A PKI is the set of software and storage products, networking facilities, and administration services that support the use of digital certificates. The role of the Certification Authority is to provide certificate users with services to obtain, renew, or revoke certificates and publish information on revoked certificates.

Many enterprises make use of public commercial organizations to supply signed certificates for use within the enterprise. As the number of internal servers increases, and as the level of client authentication rises, the costs can increase dramatically. However, if these certificates are used entirely within their own enterprise, then there is no absolute need for the certificates to be signed by such commercial organizations. Instead a local Certification Authority can be established using z/OS PKI services.

Implementation of z/OS PKI Services

z/OS PKI Services is a base component of z/OS intended to provide CA services hosted by z/OS to users on z/OS or other platforms. It is up to an installation to decide whether to set it up and make it available to the installation's users. It is necessary to additionally start two instances of the z/OS HTTP server (also a base component of z/OS). Figure 7-16 on page 201 shows the components of z/OS PKI Services.

PKI services may be used to manage existing public/private keys for certificates or it can generate the keys itself. If PKI services is used to generate these keys then it will make use of ICSF to store the keys in the TKDS.

Supported Standards

z/OS PKI Services supports the following standards for public key cryptography:

- Secure Sockets Layer (SSL) version 2 and version 3, with client authentication
- PKCS #10 browser and server certificate format, with a base64-encoded response
- IPSEC certificate format
- S/MIME certificate format
- Browser certificates for:
  - Microsoft Internet Explorer version 5.x
  - Netscape Navigator and Netscape Communicator version 4.x
- Server certificates

- LDAP standard for communications with the directory
- X.509v3 certificates
- Certificate revocation lists (CRLv2)
- Key lengths up to 4096 bits for the RSA CA signing private keys and up to 1024 bits for DSA keys
- RSA algorithms for encryption and signing
- DSA algorithms for signing
- MD5 and SHA1 hash algorithms
- RFC 2560: Online Certificate Status Protocol - OCSP
- RFC 4291: IP Version 6 Addressing Architecture
- Cisco Systems' Simple Certificate Enrollment Protocol (SCEP) (Internet draft: draft-nourse-scep-11.txt)

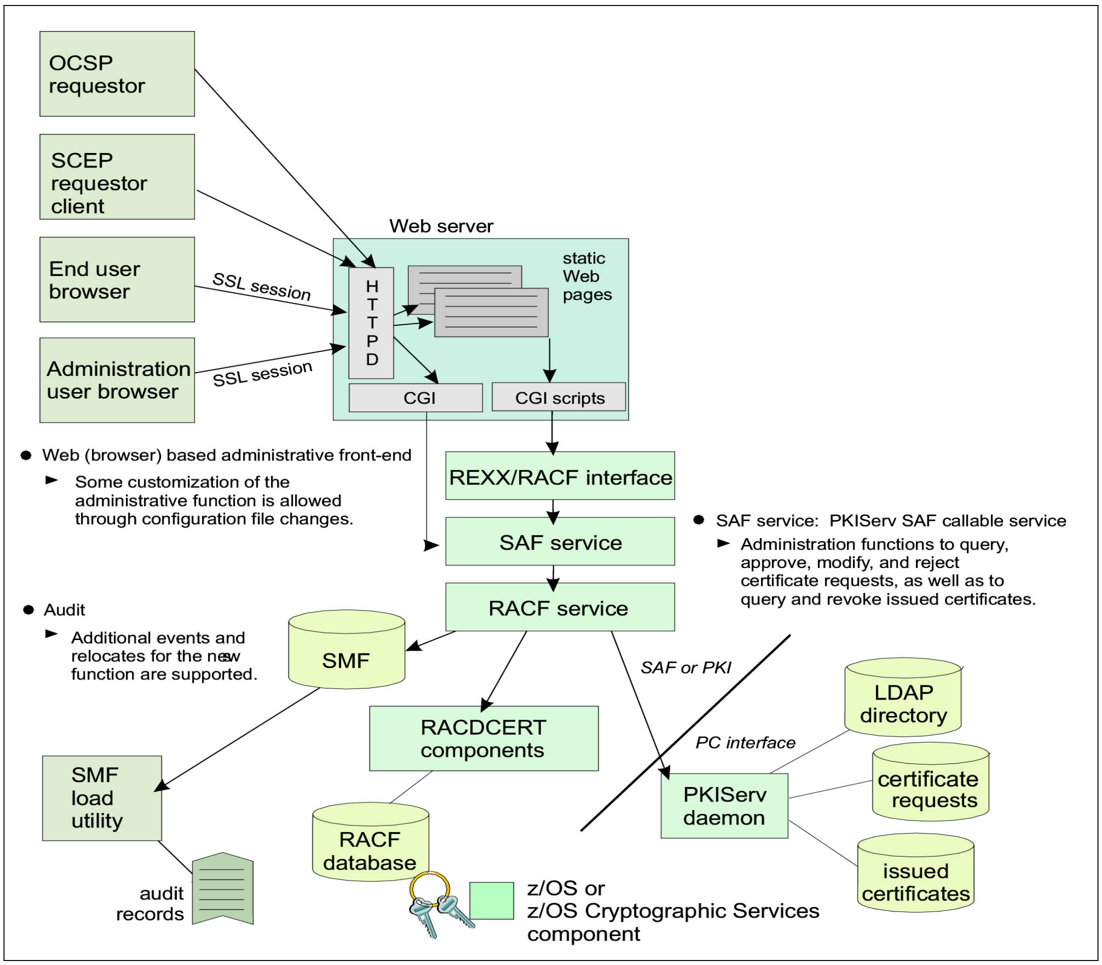


Figure 7-16

7.6.4 z/OS Integrated Security Services

This section discusses three z/OS components which form part of the Integrated Security Services.

Enterprise Identity Mapping (EIM)

Enterprise Identity mapping provides a capability of mapping multiple user registries of user identity with a view to providing a single view of the user.

It comes in two pieces:

- A set of administrative utilities and APIs to build and maintain identity mapping information in an LDAP directory, which is then called the EIM “Domain Controller”.
- An EIM client API for C/C++ and Java™ applications that allows these applications to connect to the EIM Domain Controller and get the mapping information via the LDAP protocol.

The intent is that these applications can then map the foreign user identity they have been provided with to an identity local to the platform they are executing on. This local identity can then be used by the access control mechanism of this platform.

EIM is further described, with examples, in *z/OS 1.6 Security Services Update*, SG24-6448.

Network Authentication Service (NAS)

Network Authentication Service was initially the z/OS implementation of the Kerberos authentication protocol and Key Distribution Center, along with the support of the Generalized Security Services API (GSS-API), or the Kerberos API, for C/C++ Kerberos-enabled applications. Access to a subset of the GSS-API functions has been made available to non-Language Environment applications with the R_GenSec RACF callable services.

Kerberos is a widely used authentication protocol in the distributed world, today at Version 5 and supported by many platforms such as UNIX, AIX®, Windows, and so forth. z/OS applications that are Kerberos-enabled can interoperate for authentication, and optionally for encryption of data, with partner applications on those platforms. As of the writing of this book the following z/OS applications are Kerberos-enabled:

- DB2 V7 and above (for authentication)
- FTP client and server (for authentication, optionally data integrity and privacy)
- Telnet server (for authentication, optionally for data integrity and privacy)
- LDAP client and server (for authentication)
- rshd server (for authentication, optionally for data integrity and privacy)
- NFS server (for authentication, data integrity and privacy)

More information on the z/OS Kerberos implementation is available in *Putting the Latest z/OS Security Features to Work*, SG24-6540 and *z/OS 1.6 Security Services Update*, SG24-6448.

LDAP - IBM Tivoli Directory Server for z/OS

The z/OS LDAP server is intended to support interactions with LDAP clients that go beyond the normal use of an LDAP directory. It comes with “backends” that support these specific interactions along with the connections to specific data repositories. The backends available are listed below. Note that the LDAP server can run with multiple active backends.

- The SDBM backend

  The SDBM backend allows authorized LDAP clients to access, in search or update, RACF data kept in the USER and GROUP profiles.

- The LDBM backend

  This backend provides the “classical” LDAP service, that is, the capability of storing and retrieving information in an LDAP directory. The directory data is backed up in z/OS UNIX files.

  If the LDAP object stores user passwords, the LDBM backend can proceed with an encryption of the passwords; or if the user happens to also be a RACF user, it can refer to the password as stored in the user’s RACF USER profile (z/OS LDAP “Native Authentication”).

- The GDBM backend

  The GDBM backend directory is a journal of changes that occurred in other directories managed by the LDAP server in the z/OS image. The intent is to make the occurrence of a change directly visible to an LDAP client, assuming that this LDAP client will then manage to inspect and get the relevant changed data.

- The TDBM backend

  This backend provides the “classical” LDAP service, that is, the capability of storing and retrieving information in an LDAP directory. The directory data is backed up in DB2 tables. If the LDAP object stores user passwords, the TDBM backend can proceed with an encryption of the passwords; or if the user happens to also be a RACF user, it can refer to the password as stored in the user’s RACF USER profile (z/OS LDAP “Native Authentication”).

  When running with the IBM TDS server, the user has the choice to store the directory data in z/OS UNIX files (with the LDBM backend) or in DB2 tables (with the TDBM backend). Both backends support the same functions. The choice is mainly driven by data volume size and scalability considerations.

7.7 Certification

z/OS has been evaluated at several times during its journey from OS/VS2 to z/OS. A very well-known certification was performed on MVS 3.1.0e (<verification needed>) when the concepts of multi-level security were first introduced. This certification was performed after changes had been made with a view to MVS 3.1.0e conforming to the B1 standard as defined in the famous (or infamous) Orange Book. This book was produced by the USA department of defense and specified levels for what was described as the Trusted Computer System Evaluation Criteria.

In more recent times z/OS has been certified against the Common Criteria (see Chapter Two <specific reference needed>). The most recent of these was performed for z/OS Version 1 Release 9. The certification level that was achieved can be seen in the report which can be found here,

http://www.commoncriteriaportal.org/files/epfiles/20080527_0459a.pdf

You will see that the level achieved was EAL4+ALC-FLR.3.

Target Of Evaluation (TOE)

For z/OS Version 1 Release 9 the Security Target can be found here,

http://www.commoncriteriaportal.org/files/epfiles/20080527_0459b.pdf

This document defines what the Target Of Evaluation includes. From Security Target document it should be apparent that not all components of z/OS are included. This may be important dependent on the use that is made of z/OS components. Remember that this does not necessarily mean that the excluded components have a lower view of security or integrity; it merely means that they were not included in the assessment. In addition there are strong restrictions placed on the configuration of z/OS. Certain functions within the environment must be disabled. See section 2.3.1 in the Security Target document for full details of configuration.

However, the TOE used in this evaluation does include,

  – TSO/E

- Batch Processing (JES2 only)
- Unix System Services
- TCP/IP communications

  (Note that TCP/IP communications that occur outside of the Sysplex are limited to a single security label.)
- Kerberos
- LDAP
- AT-TLS

Note that the Security Target document states that the addition of any other software which does not run in an "authorised state" or with access to specific USS resources will not undermine or invalidate this certification.

It further states that other software which does require one of those capabilities may be added as long as that software is evaluated to the point where it can be shown that it does not undermine the security policies described in the document.

The document does state that some components are explicitly excluded. These include,

- Bulk Data Transfer
- Connection Manager
- DCE component of the Integrated Security Services element.
- DCE services
- DFS Server Message Block (SMB) and DFS DCE-DFS components of the Distributed File Service element
- Enterprise Identity Mapping component of the Integrated Security Services element.
- Infoprint Server
- JES3
- APPC/MVS
- Process Manager component of USS

The Target of Evaluation makes no reference to the Trusted Key Entry (TKE) workstation.

For this level of certification z/OS was assessed against two security profiles; CAPP and LSPP.

Applicability

The assurance measures provided for the TOE can be found in section 6.10 of the Security Target document. Attention is drawn to the last of these (AVA_VLA.2) which refers to IBM's own vulnerability analysis and penetration testing capability and the experience IBM has with performing this type of testing.

The extent to which the security evaluations are relevant to an individual organization or enterprise will relate to the extent to which the software and hardware configuration matches the evaluated version.

For example,

- z/OS Version 1 Release 9 can be run on a z900 processor, but the TOE environment does not include that processor.

- The enterprise uses JES3 in place of JES2.
- The enterprise may make use of software which requires some new modules be run in an authorised state, but that software does not have certification.
- The enterprise in question may choose to make use of software components which are outside the TOE, or of configuration options which then place the z/OS instance outside the TOE.

So the applicability of the certification is a judgement that each enterprise must make independently. Nevertheless the fact that z/OS has been evaluated and achieved certification at the level it has, should increase the level of confidence and trust in z/OS as a whole.

8

Hosting the building blocks of the IBM Security Framework in z/OS

We give in this chapter a very high level description of some other IBM products that potentially contribute to the implementation of the IBM Security Framework and exploit the strengths of System z. These products either complement the z/OS built-in security services or they provide additional enterprise wide security-oriented functions.

The products that we are describing here relates to the following Blueprint Security Services and Infrastructures:

- Security Information and Event Management Infrastructure
- Identity, Access, and Entitlement Infrastructure
- Security Policy Infrastructure
- Cryptography, Key, and Certificate Infrastructure
- Storage Security
- Host and Endpoint Security

8.1 Complementing z/OS RACF

To use the RACF administration commands, or relevant TSO/E ISPF panels, requires specific administration skills that many installations today find not to be synergistic with their other needs regarding the administration of their non-z/OS platforms, which, for most of them, can be administered via somehow friendly graphical interfaces.
There is also now a need for regulatory compliance checking that goes beyond a simple verification of the system-level security, and actually requires to get both a focused and global view of the security setups and events that the RACF built-in reporting facilities do not provide.

The IBM approach to meet these additional requirements is to complement the RACF native user interface in z/OS with IBM Tivoli products that provide:

- Add-on functions for automating the administration and auditing of RACF data.
- Monitoring, auditing and compliance checking tools that consolidate RACF information with other systems information, in order to provide a view and rating of the security-related events and behaviors at the enterprise level.

8.1.1 IBM Tivoli zSecure administration products

With the acquisition of the Consul company in January 2007, IBM Tivoli now owns the zSecure suite of products that perform both administration and auditing of mainframe external security managers, such as RACF.

The zSecure administrative products consist of zSecure Admin, zSecure Visual, and zSecure CICS Toolkit, which are intended to assist for RACF administration.
The products all share common components, including the Consul's proprietary CARLA high level programming language.

Note: a Tivoli zSecure Manager is also available for RACF in z/VM. Applicable features of IBM Tivoli zSecure Admin and IBM Tivoli zSecure Audit components for IBM z/OS, that we are describing below, form the foundation for this functionality.

IBM Tivoli zSecure Admin

This product consists of an ISPF-based user interface for the administration of RACF attributes. It runs entirely within z/OS without requiring any collaborating component on a distributed platform. zSecure Admin is intended to enable more efficient and effective RACF administration, using significantly less resources, and can be used to:

- Automate routine tasks to simplify administration
- Identify and analyze problems to minimize threats
- Merge databases quickly and efficiently
- Display data from the active (live) RACF database
- Integrates smoothly with IBM Tivoli zSecure Audit
- Store non-RACF data to reduce organizational costs

Figure 8-1 shows an ISPF panel generated by zSecure Admin. zSecure Admin provides an interface that makes the RACF database look like a scrollable page of data (the panel in Figure 8-1 is actually the output of a RACF LISTUSER command). Using the analogy of an ISPF editor to present RACF profiles, the user can type over fields and make the changes in “what the user sees is what the user gets” mode. When the user makes a change to the panel and presses enter the proper RACF command is automatically generated.

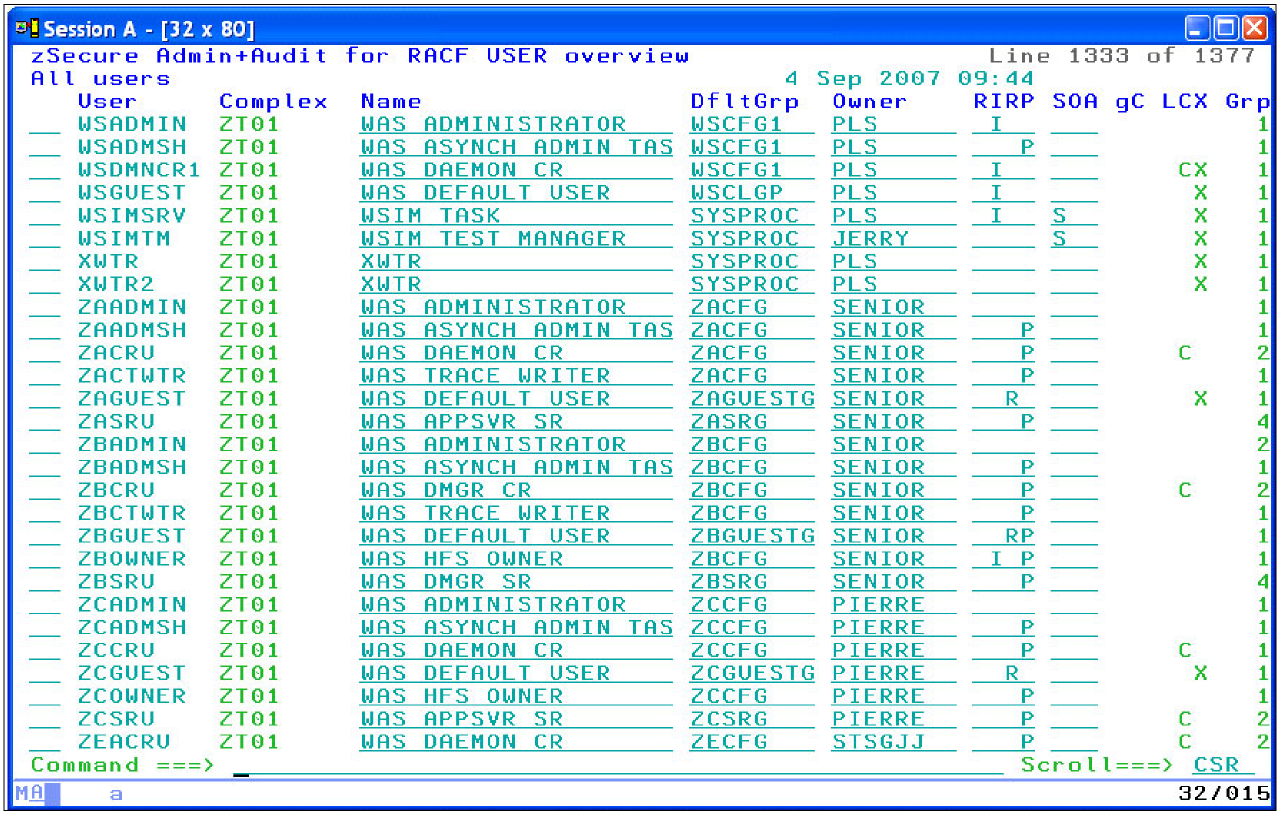


Figure 8-1 IBM Tivoli zSecure Admin

IBM Tivoli zSecure Visual

This product consists of a Windows-based user interface running on a Windows machine. It communicates, via TCP/IP, with a z/OS started task that performs limited RACF administration on behalf of the user. The intent of zSecure Visual is to reduce the need for sometimes scarce RACF-trained expertise through a Microsoft Windows–based GUI that is used to drive RACF administration and can be used to:

- Decentralize RACF administration to optimize resources
- “Scope Down” Administrative Capabilities
- Avoid need for TSO/ISPF rollouts
- Administer a live RACF database
- Easily clone user templates

Figure 8-2 shows an example of graphical menus used by zSecure Visual, which displays a list of USER profiles in the RACF database with the REVOKED attributes. Actually these profiles were obtained after triggering a search for USER profiles with the drop down menu at the right hand bottom corner of the screen.

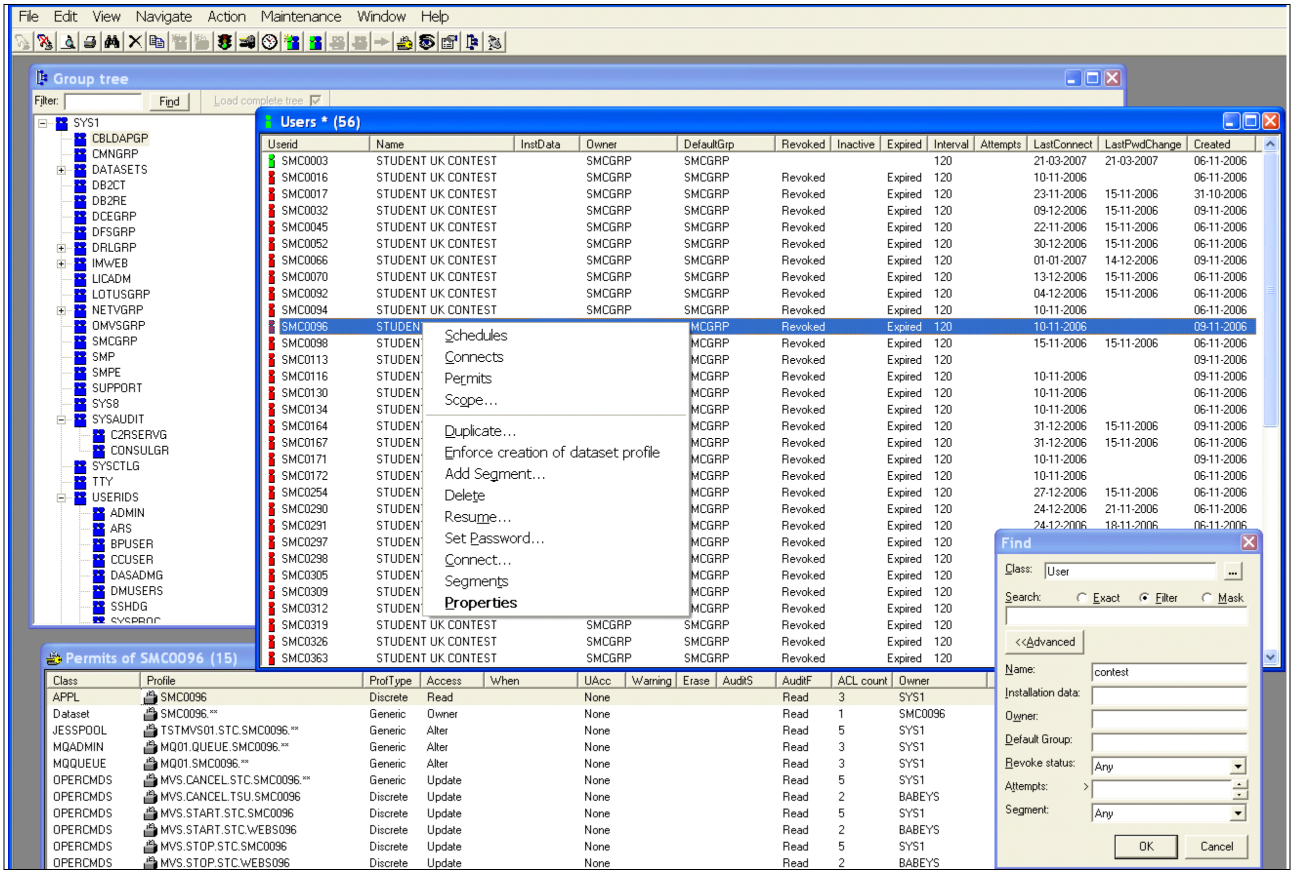


Figure 8-2 IBM Tivoli zSecure Visual

8.1.2 IBM Tivoli zSecure CICS Toolkit

This product has two facets. It is a pre-built administrative interface that runs as a CICS transaction in a CICS region. It also provides a CICS API to allow applications to perform their own security functions. For example, if an application needed a re-verification of user credentials when certain program constraints were met (such as funds transfer over a certain amount) the CICS Toolkit API could be used to drive this re-verification.

Similarly, focusing on the same examples, the CICS Toolkit API could also be used to drive RACF audit capability. For instance, an SMF record can be generated to log the fact that this particular user executed a funds transfer above a certain amount.

Figure 8-3 shows a pre-built CICS RACF administration menu.

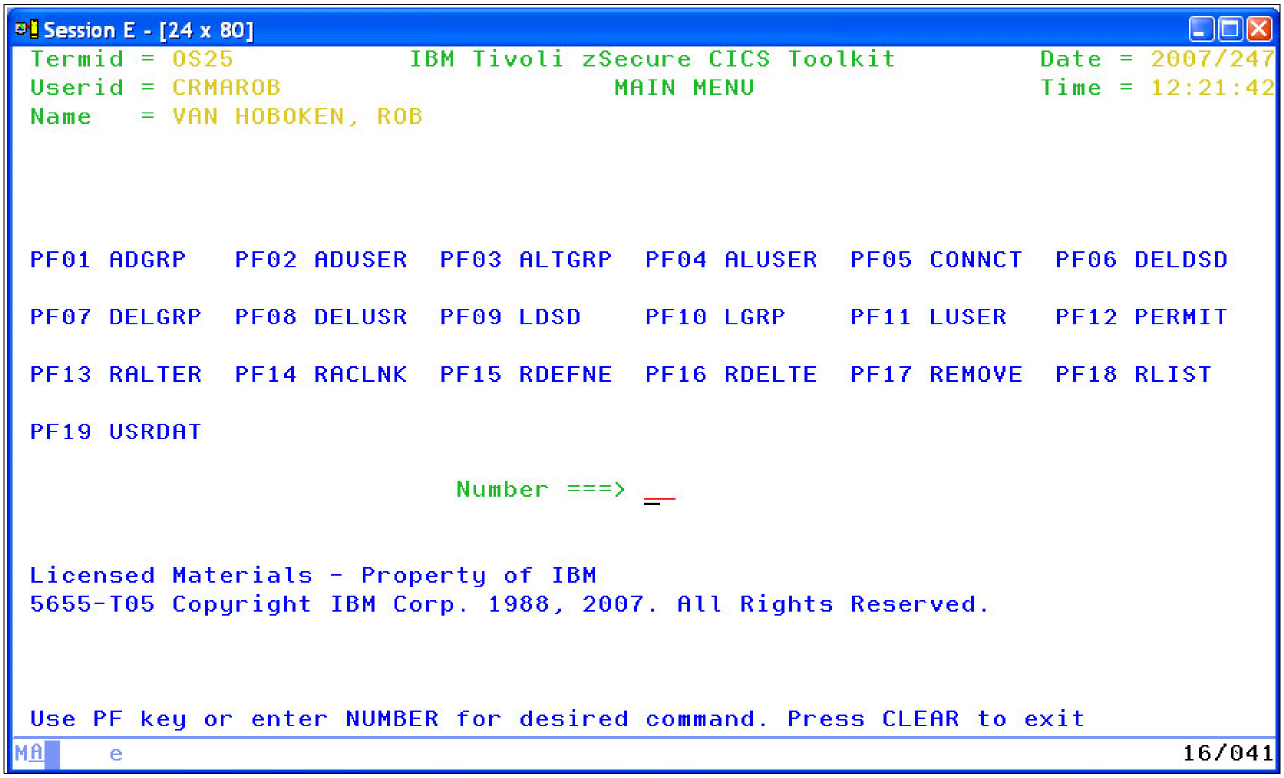


Figure 8-3 IBM Tivoli zSecure CICS Toolkit

8.1.3 Risk and compliance products

The IBM Tivoli risk and compliance products approach is to use agents in z/OS, and other operating systems, that provide information centrally collected by a focal point product that provides consolidated information to security administrators. Such focal-point products are the IBM Tivoli Security Operations Manager and Tivoli Compliance Insight Manager. Note that both of these products are not running today on System z.

Tivoli Security Operations Manager

The Tivoli Security Operations Manager is collecting real-time information, correlating the data with a view to finding policy violations or intrusion attempts, and reporting this. It is part of a Security Information and Event Management (SIEM) Tivoli platform.
Rather than having agents distributed to get the data, it acts as a central collection service and other components route data to it, such as from UNIX syslogs or intrusion detection software. To do so it uses conduits to receive SMTP messages, SNMP traps, and syslog data.

z/OS specific information is provided by the zSecure Alert product that runs in z/OS, as shown in Figure 8-4 on page 212. IBM Tivoli zSecure Alert is further described at “IBM Tivoli zSecure Alert” on page 215.

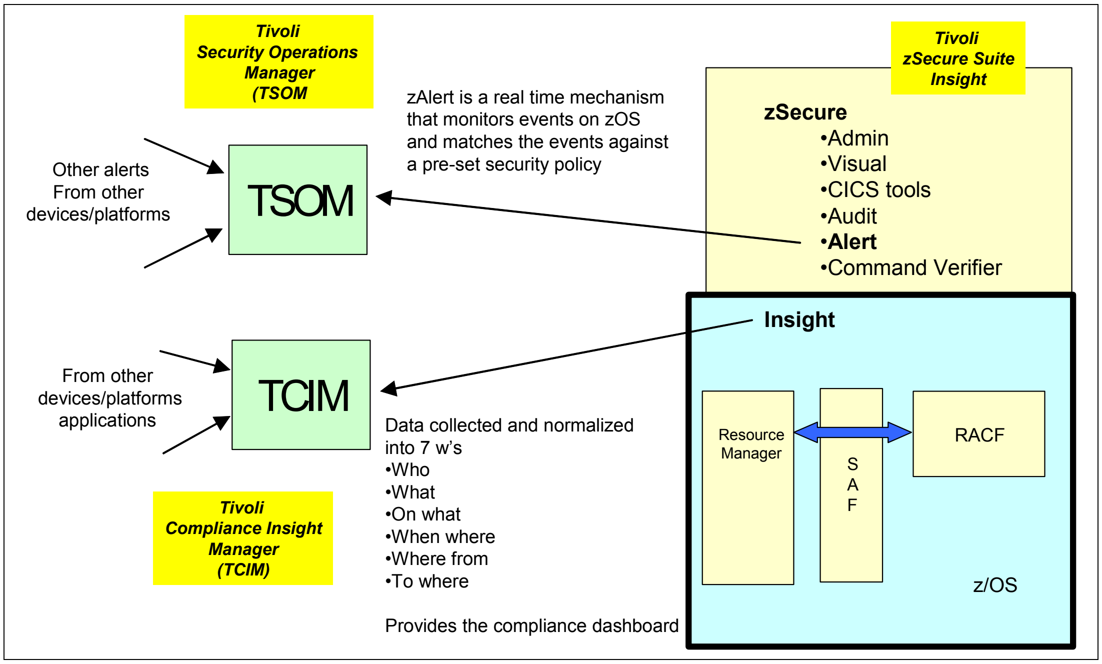


Figure 8-4 IBM Tivoli Security Operations Manager and Tivoli Compliance Insight Manager

Figure 8-5 is an example of a z/OS security alert reported by Tivoli Security Operations Manager. This particular alert is pointing out that a userID that is not assigned the system OPERATIONS authority somehow managed to use that authority which allows access to any datasets on the mainframe (this information surfaces from the analysis of the z/OS RACF generated SMF records).

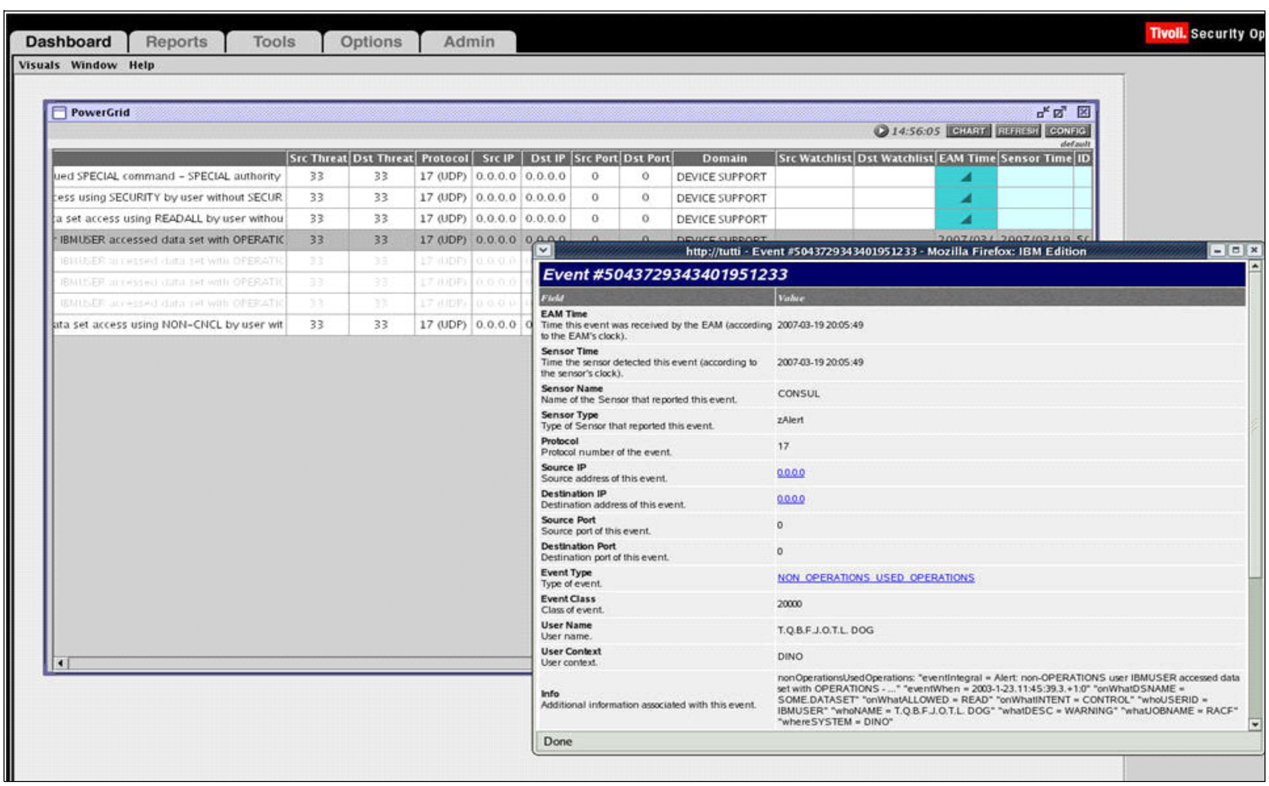


Figure 8-5 BM Tivoli Security Operations Manager display

Tivoli Compliance Insight Manager

Tivoli Compliance Insight Manager is also part of the security information and event management (SIEM) system together with Tivoli Security Operations Manager, but it focuses on compliance functions related to people and system and data access.
It ships with a number of compliance checking modules pertaining to regulations such as HIPAA and SOX.

Tivoli Security Operations Manager is focused on real-time correlation and operations management, while Tivoli Compliance Insight Manager is more focused on compliance and audit. In fact, Tivoli Security Operations Manager can also provide data to be fed into Tivoli Compliance Insight Manager.

The Tivoli Compliance Insight Manager z/OS Actuator (the z/OS Agent for Insight) runs on z/OS using z/OS components such as started tasks and data sets. The Tivoli Compliance Insight Manager uses the event data that are provided within the SMF records of type:

- 0, 2, 3, 4, 5, 6, 8, 10, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 27, 39, 40, 41, 42, 43, 45, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
- 80, 81 (RACF)
- 82 (ICSF – Integrated Cryptographic Services Facility)
- 83 (Security Events)
- 84, 85, 88, 91, 92, 94, 96, 99, 100, 101, 103, 108, 109, 115, 116, 118, 119, 120

It copies this data to a file that is stored in z/OS UNIX Services and then passes the data to the Tivoli Compliance Insight Manager. It can capture and process z/OS (including z/OS UNIX), RACF, ACF2, TopSecret, and DB2 SMF data. It can also process zSecure Alert events, as shown in Figure 8-4.

Figure 8-6 shows the entry pane for the so-called “Compliance Dashboard” that Tivoli Compliance Insight Manager displays. The dashboard shows all activities in the enterprise. The size of each circle indicates the amount of activity (logged events). On the axes the administrator can compare People (Who) with Information (on What).
The policy violations over time are shown on the right and log databases sorted to the needs of the user’s business (by department, regulation or technology) are available.

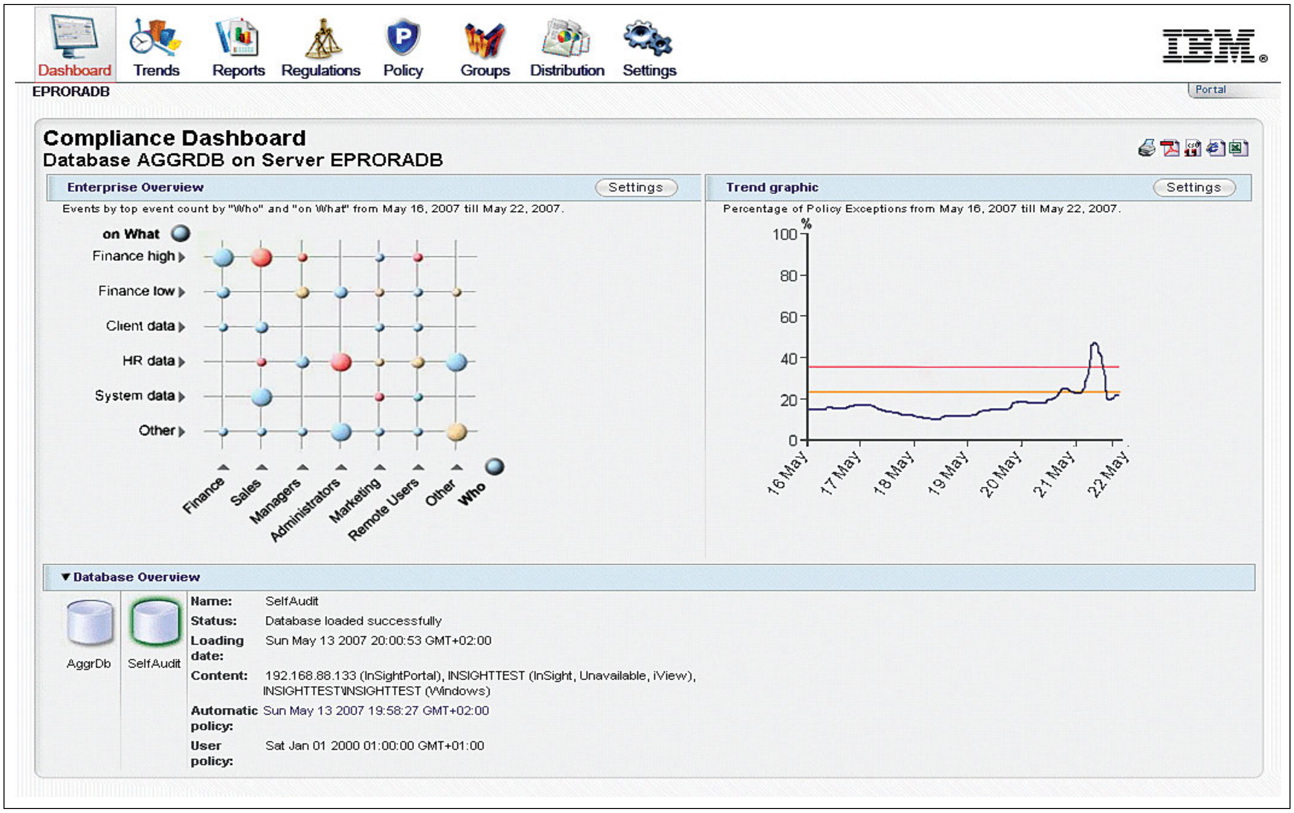


Figure 8-6 IBM Tivoli Compliance Insight Manager Compliance Dashboard.Tivoli zSecure audit products

The second half of the zSecure suite products is related to audit and compliance functions, zSecure Audit, zSecure Alert and zSecure Command Verifier. These three products all run on z/OS and operate on the z/OS security data and commands.

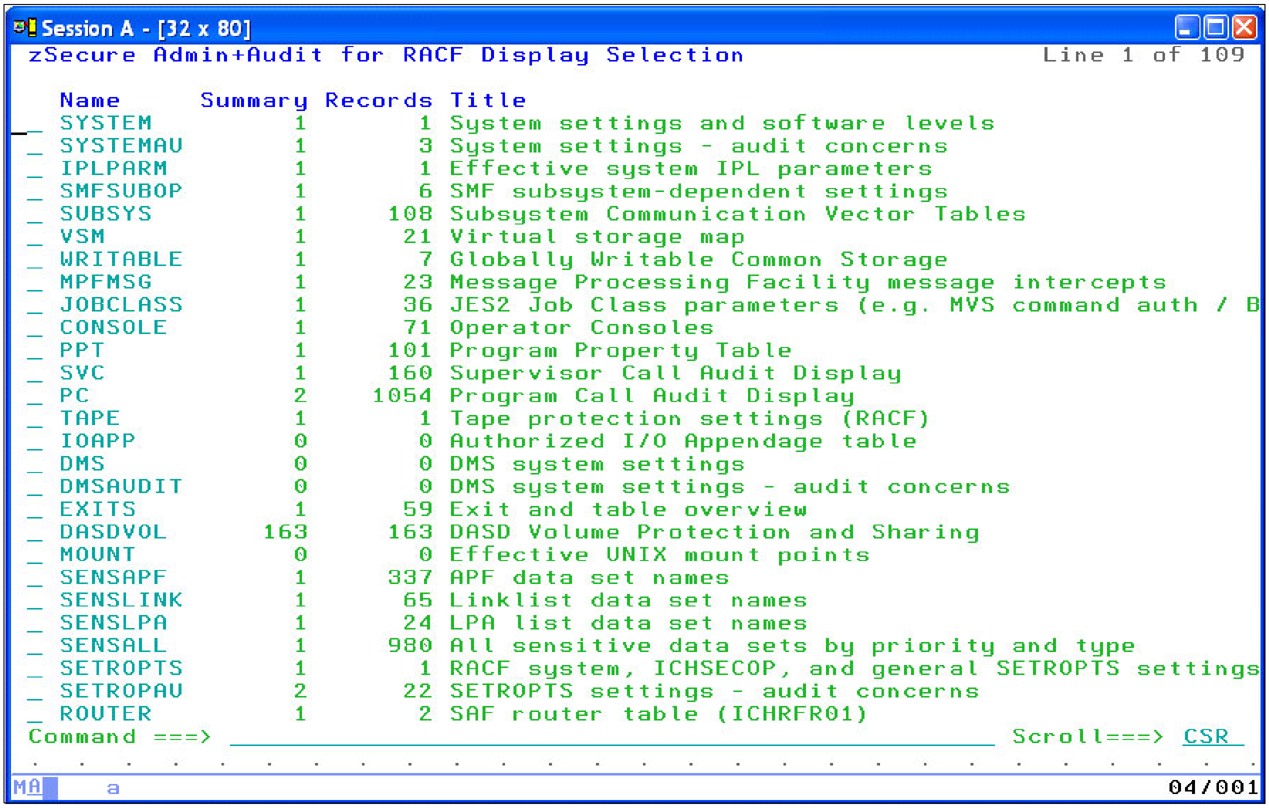
IBM Tivoli zSecure Audit

The zSecure Audit product runs in z/OS and analyzes security data, such as historical SMF data, and security configuration, such as RACF objects and system libraries, to identify and report on any security exposures.

Highlights of zSecure Audit functions are:

- Live analysis of critical information that goes beyond z/OS and RACF analysis
- Customize reports to meet specific needs with flexible report and alert language
- Analyze SMF log files to create a comprehensive audit trail
- Analyze RACF profiles to get fast answers
- Detect system changes and integrity breaches to minimize security risks
- Track and monitor baseline changes for the RACF database
- Integrated remediation with Tivoli zSecure Admin
- Seamless links to enterprise audit and compliance

Figure 8-7 shows an ISPF panel displayed by zSecure Audit.



Name	Summary	Records	Title
SYSTEM	1	1	System settings and software levels
SYSTEMAU	1	3	System settings - audit concerns
IPLPARM	1	1	Effective system IPL parameters
SMFSUBOP	1	6	SMF subsystem-dependent settings
SUBSYS	1	108	Subsystem Communication Vector Tables
VSM	1	21	Virtual storage map
WRITABLE	1	7	Globally Writable Common Storage
MPFMSG	1	23	Message Processing Facility message intercepts
JOBCLASS	1	36	JES2 Job Class parameters (e.g. MVS command auth / B
CONSOLE	1	71	Operator Consoles
PPT	1	101	Program Property Table
SVC	1	160	Supervisor Call Audit Display
PC	2	1054	Program Call Audit Display
TAPE	1	1	Tape protection settings (RACF)
IOAPP	0	0	Authorized I/O Appendage table
DMS	0	0	DMS system settings
DMSAUDIT	0	0	DMS system settings - audit concerns
EXITS	1	59	Exit and table overview
DASDVOL	163	163	DASD Volume Protection and Sharing
MOUNT	0	0	Effective UNIX mount points
SENSAPF	1	337	APF data set names
SENSLINK	1	65	Linklist data set names
SENSLPA	1	24	LPA list data set names
SENSALL	1	980	All sensitive data sets by priority and type
SETROPTS	1	1	RACF system, ICHSECOP, and general SETROPTS settings
SETROPAU	2	22	SETROPTS settings - audit concerns
ROUTER	1	2	SAF router table (ICHRFR01)

Figure 8-7 IBM Tivoli zSecure Audit

Note that zSecure Audit also explains what the exposure is in English words.

IBM Tivoli zSecure Alert

The zSecure Alert product gathers events from SMF and provides real-time monitoring of intruders, system activity, from a security perspective, and system configuration. As with zSecure Audit, this product runs within z/OS.

The highlights of the provided functions are:

- Threat knowledge base with parameters from the user's active configurations
- Broad range of monitoring capabilities, including monitoring sensitive data for misuse on:
  - z/OS
  - IBM RACF
  - z/OS UNIX subsystems.
- Easily send critical alerts to enterprise audit, compliance and monitoring solutions
- Integrated remediation with Tivoli zSecure Admin

Both zSecure Alert and zSecure Audit can send data to Tivoli Compliance Insight Manager for analysis and reporting. There are also other destinations for report and alert data.

IBM zSecure Command Verifier

This product, previously known as zLock in the Consul portfolio, is often listed as an audit or policy compliance tool. However, it can be a very effective delegated/distributed administration control mechanism. It allows profiles to be defined to limit the RACF command arguments that can be specified, including filters on values.

For example, the system administrator may decide that no user can be created with a name of TESTUSER and setup RACF accordingly (zSecure Command Verifier uses the $CAR class of profiles). A delegated administrator attempting to create a user with this name will see the command being refused as shown in Figure 8-8.
This product runs completely within a z/OS system. As it is using an exit, it captures all administrative commands, whether they are done through a command line, a job, or an administrative tool.

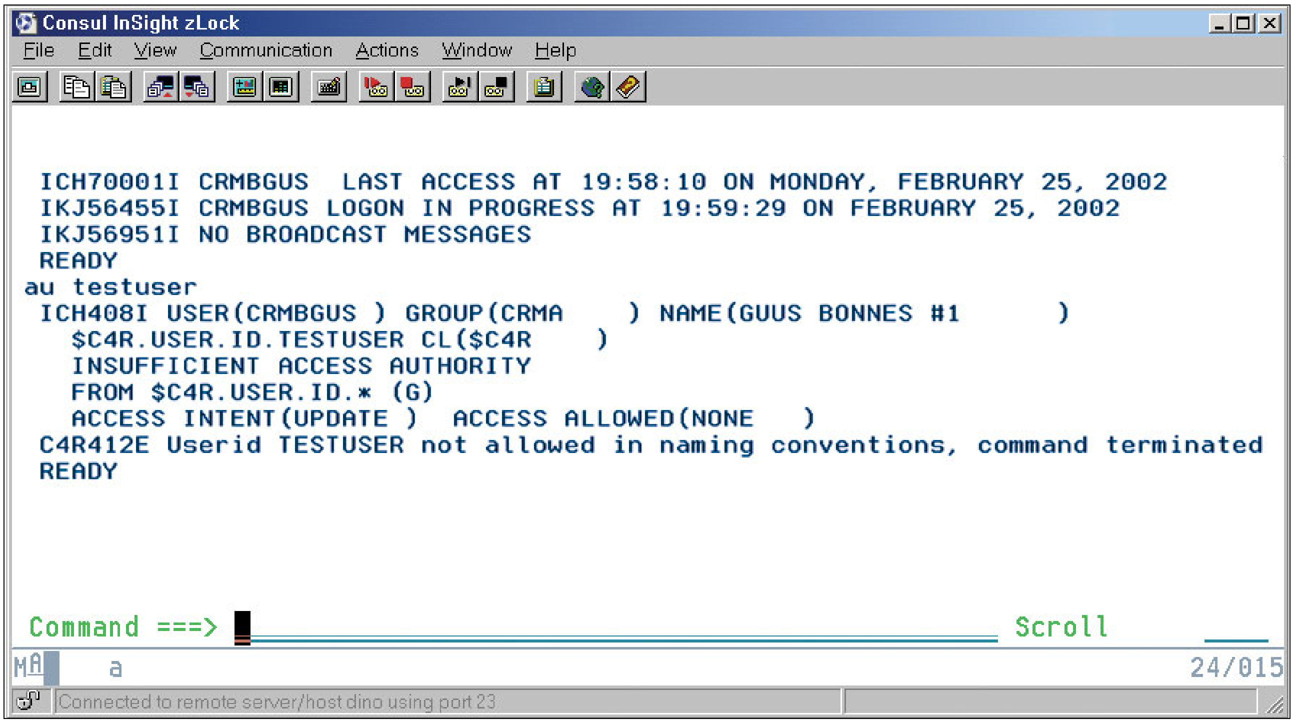


Figure 8-8 IBM Tivoli zSecure Command Verifier

8.2 Java and z/OS Security services

The IBM SDK for z/OS is delivered in z/OS ServerPac or CBPDO or can be downloaded from the IBM web site. It contains:

- The Java Runtime Environment (JRE) that comprises:
  - The Java Virtual Machine (JVM)
  - The production core classes delivered in any Java implementation
  - A set of trusted root certificates
  - Java Runtime Environment tools (e.g. keytool)
- A Java development toolkit with the JAR tool, the Javac compiler and other miscellaneous tools.

The JVM executes in z/OS as a z/OS UNIX application and as such benefits of the z/OS Unix System Services security features. However Java has its own security model which is not directly supported by z/OS security services, the JVM has therefore to create the specific Java security runtime environment that is required for the applications it has to execute.

Some specific z/OS security services are however made available to Java applications executing on z/OS that we are briefly explaining in this chapter.Further details can be found in the IBM Redbook “Java Security on z/OS -The Complete View”, SG24-7610.

8.2.1 z/OS security services available to Java applications

The following z/OS security services are available via Java APIs:

- RACF authentication and authorization though the JAAS (Java Authentication and Authorization Services) API.
- Miscellaneous RACF services via the SAF classes APIs (note that these are not standardized Java APIs as JAAS can be, they are specifically designed for z/OS Java applications):
  - Checking of the userID in effect for access rights to a RACF-protected general resource
  - Extraction the RACF userID from current thread
  - Checking of the active status of RACF or a specific RACF class of profiles
  - Userid and password authentication or user’s RACF password change
  - Checking of a user membership to a RACF group
- RACF USER and GROUP profiles administration through the JSEC API
- RACF passtickets
- z/OS Enterprise Identity Mapping (EIM)
- z/OS integrated hardware cryptography support - We are further developing this support for z/OS Java applications in the next section.

8.2.2 z/OS Java use of the integrated hardware cryptography support

Complex Security services, including cryptographic services, have been available since many years for Java applications as add-ons to the different SDK implementations. Starting with SDK 1.4.0 these functions are integrated in the base Java 2 framework, meaning that the functions are part of the set of common APIs made available to Java applications across platforms (also termed the Java 2 Framework).

The Java 2 framework offers a specific API that gives explicit access to elementary cryptographic services such as data encryption or decryption. This is the JCE (Java Cryptographic Extension) API.
As an example, the JCE API gives access to:

- Digital signatures
- Hashing
- Symmetric encryption/decryption
- Asymmetric encryption/decryption with RSA

More recently a Java version of the PKCS#11 API has been made available. The PKCS#11 API is an industry-accepted standard commonly used by cryptographic applications. The PKCS#11 standard is defined on the RSA Laboratories Web site at:

http://www.rsasecurity.com/rsalabs/PKCS/

The underlying architecture

Following are the elements that are building the underlying architecture:

Service providers

The functions offered by the API are performed by calling Java service classes (also termed "engines") which themselves rely on "service providers" to implement the required functions and algorithms. That is an implementation architecture that support use of plug-replaceable components, as there are cryptographic service providers available from Sun, IBM or other vendors. Each one of these providers is a specific implementation of the JCE or other APIs. As an example IBM is proposing two different JCE providers, coming as.jar files:

- The IBMJCE cryptographic provider
- The IBMJCECCA cryptographic provider

Keystores

As for any implementation of cryptographic services, keys need to be kept and managed in secure repositories. Java uses the concept of "keystore", where individual secret keys are kept under a label, called the *alias*, and encrypted with a password. The repository itself is also protected with its own *store password*.

The keystore concept can be implemented in different way, such as by using for example a flat file, it can also be implemented using other technologies, as it is the case with the z/OS ICSF PKDS VSAM data set. The *keystore type* specifies what is the keystore implementation to consider. The keystore types supported in an installation are also related to the cryptographic providers that are used.

8.2.3 Focusing on the cryptographic providers implementation in z/OS Java

Among the many cryptographic providers delivered in the IBM Java SDK for z/OS, the IBMJCE, IBMJCECCA and the IBMPKCS11Impl are the ones providing the basic cryptographic functions that other providers are invoking.

The IBMJCE provider

The IBMJCE provider implements the JCE services by software only. It provides the following services at the SDK V6 level:

- Digital signatures with the DSA or RSA algorithm, in combination with the hashing algorithms below.
- Hashing with SHA1, SHA256, SHA384, SHA512, MD2, MD5.
- Symmetric encryption/decryption with DES, Triple-DES, AES128, AES192, AES256, PBE, Blowfish, Mars, RC2, RC4.
- Asymmetric encryption/decryption with RSA and ElGamal
- Key agreement with Diffie-Hellman
- RSA with Optimal Asymmetric Encryption Padding (OAEP)
- Hash-based Message Authentication Code (HMAC) with MD5, SHA1, SHA256, SHA384 and SHA512.

The keys used by the IBMJCE provider are generated and managed with the java *keytool* utility. The keytool utility comes with the provider and therefore requires that IBMJCE be specified in the providers list.

•**Note:** for installations with a requirement to use a FIPS-certified provider, IBM delivers the specific IBMJCEFIPS provider. Information on FIPS certification of providers is given at http://cs-www.ncsl.nist.gov/cryptval

The IBMJCECCA provider

The IBMJCECCA provides those JCE services that can be performed by the hardware cryptographic coprocessors through ICSF in z/OS. It actually calls ICSF for the IBM Common Cryptography Architecture services and therefore requires ICSF to be in operation and to be setup to provide these services.

Information on the IBMJCECCA provider can be found at:
`http://www.ibm.com/servers/eserver/zseries/software/java/j5jcecca.html`

The ICSF Cryptographic Ky Data Set (CKDS) and the Public Key Cryptographic Data Set (PKDS) can be used to securely store the symmetric and asymmetric keys to be used with the IBMJCECCA provider. Figure 8-9 is a high level description of the infrastructure layout.

The IBMJCECCA provider supports the following algorithms

- Digital Signatures via RSA and DSA (see the note below)
- Hashing - SHA1, MD2, MD5
- Keystore - Symmetric and Asymmetric keys protected by 3DES
- Symmetric Algorithms - DES, 3DES, PBE, AES
- Cipher Modes - ECB, CBC, CFB, OFB, PCBC
- Asymmetric Algorithms - RSA
- HMAC - MD5, SHA1
- Pseudo random number generation

Note: The DSA algorithm is performed by hardware on the systems z800 and z900 only. Beginning with systems z890 and z990 there is not any more hardware support for DSA in the cryptographic coprocessors.

IBMJCECCA provides the *hwkeytool* to generate and manage keys in a format appropriate to be used in cryptographic hardware operations.

Note: as ICSF is transparently called by IBMJCECCA, RACF protection for ICSF callable services, via the CSFSERV class of profiles, and keys in the CKDS or PKDS, via the CSFKEYS and XCSFKEYS classes of profiles, can also be achieved.

Access control is performed on the basis of the RACF ID of the user that started the Java application.

The IBMPKCS11Impl provider

RSA Laboratories' Public Key Cryptography Standards #11 (PKCS#11) was initially created o exploit smart cards or other plug-installable simple cryptographic devices accessed through a physical interface such as a smart card reader.

In PKCS#11 terminology such hardware devices are called "tokens", and are expected to provide hardware accelerated cryptographic operations and secure storage for sensitive information such as:

- Application specific data
- Digital certificate
- Cryptographic keys

The PKCS#11 API has been implemented in z/OS, with underlying ICSF support, at z/OS V1R9.

Note that to support PKCS#11 on z/OS, the conceptual view of the token is preserved. However, tokens are no longer physical devices but storage areas provided in the ICSF-managed Token Key Data Set (TKDS).

Note: Access to the PKCS11 "tokens" in the ICSF TKDS is protected by the CRYPTOZ class of profiles in RACF.

The z/OS IBMPKCS11Impl provider is delivered in the Java SDK V6 and allows Java applications that invoke the PKCS11 API to benefit of the z/OS integrated hardware cryptography support.

Information on the z/OS IBMPKCS11Impl guide can be found at:

http://www.ibm.com/servers/eserver/zseries/software/java/j6pkcs11implgd.html

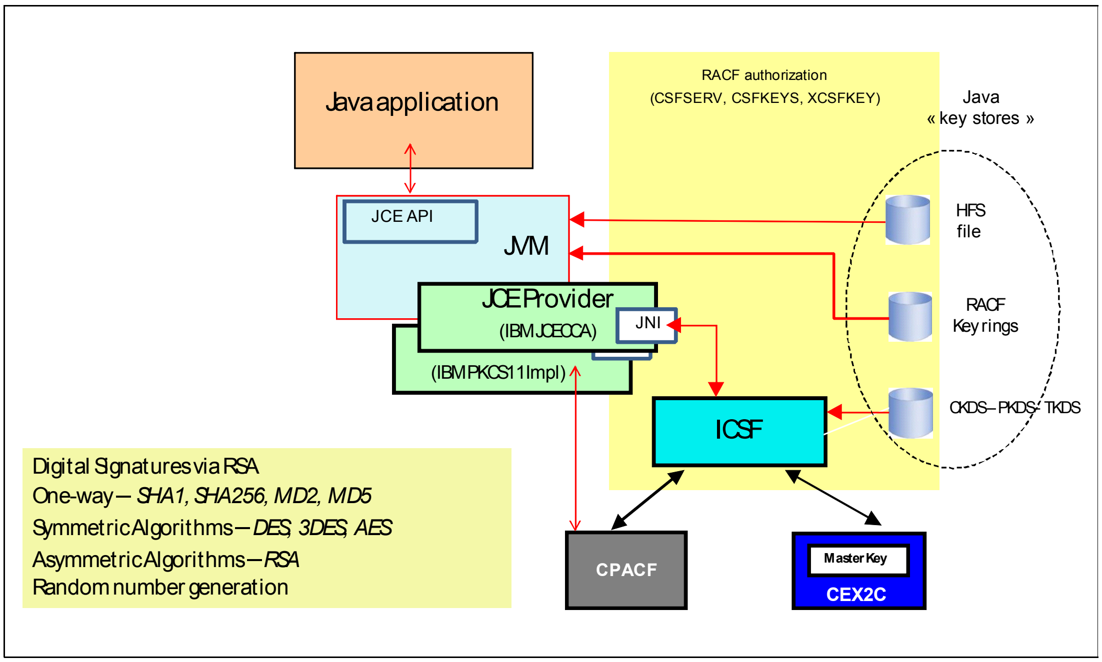


Figure 8-9 z/OS Java hardware cryptography layout

8.2.4 Java Keystores that are supported in z/OS

The keystore concept can be implemented in different ways which are reflected in the *keystore type*. The support of a given keystore type is dependant on the provider in use, and the keystore type is specified when invoking the provider. We are listing below the keystore types which are supported by the different cryptographic providers on z/OS.

IBMJCE

The IBMJCE provider supports:

- The JCEKS keystore for symmetric or asymmetric keys. The JCEKS is implemented as HFS or zFS files.
- The JCERACFKS keystore for asymmetric keys. The JCERACFKS is implemented as RACF key rings that hold RSA or DSA keys.

IBMJCECCA

The IBMJCECCA provider supports:

- The JCEKS keystore as implemented in HFS or zFS files.
- The JCECCAKS keystore, which is actually implemented using the ICSF PKDS for RSA keys and the CKDS for DES or Triple-DES keys.
- The JCERACFKS for RSA or DSA keys in RACF key ring
- The JCECCARACFKS keystore, for RSA keys accessible through a RACF key ring but actually stored in the ICSF PKDS.

IBMPKCS11Impl

The IBMPKCS11Impl provider supports the IBMPKCS11IMPLKS keystore, which is implemented as “tokens” in the ICSF TKDS~VSAM data set.

8.3 WebSphere Application Server and z/OS

z/OS does not provide Security APIs that J2EE or Web Services applications can directly use. It however optionally extends the J2EE and Web services security by providing z/OS controlled resources that back up the J2EE or Web services Security Model. From the applications standpoint, as shown in Figure 8-10, the required services are provided by the WebSphere Application Server run-time, which itself is a set of z/OS started tasks running each one of them with a z/OS RACF userID and operating according to the z/OS Security model.

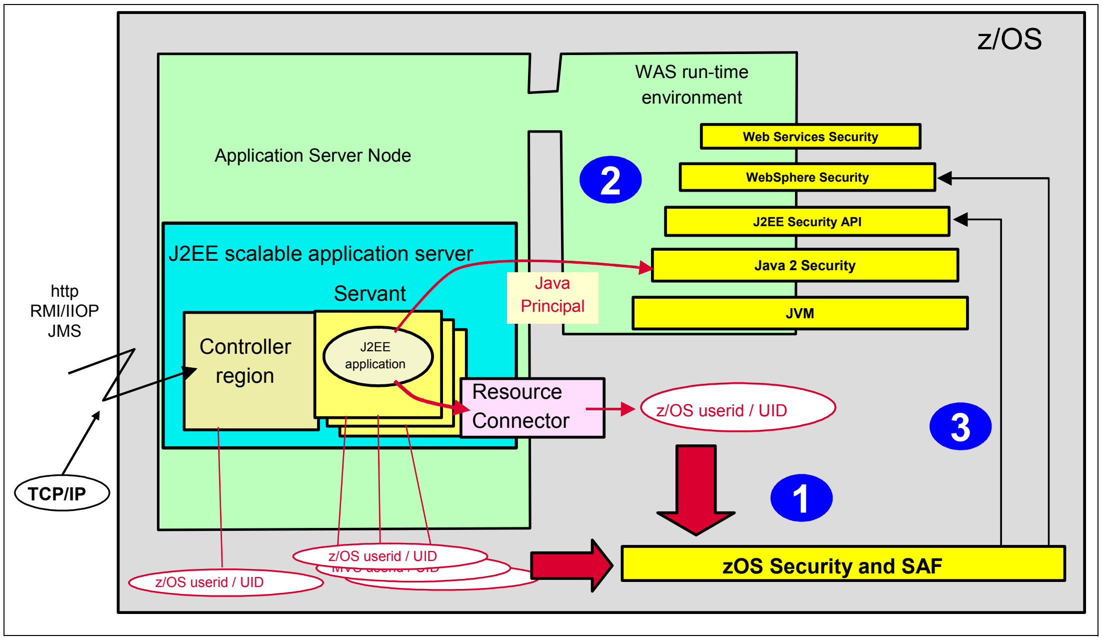


Figure 8-10 WebSphere Application Server for z/OS Security implementation

One can therefore distinguish the following different Security environments in this implementation:

1. The z/OS environment
Here we are using the z/OS regular security services to secure the execution of the WebSphere Application Server Controller and Servant regions address spaces and the JVM they host. This relies on the usual RACF mechanisms for identification of the started tasks and control of their accesses to z/OS resources. As long as the WebSphere Application Server code handles its own specific Security related entities (such as Java and J2EE subjects and their privileges regarding Java or J2EE resources) within its run-time, the z/OS Security mechanisms are not directly involved in the relevant processes.

   Note however that when the user request that WebSphere Application Server processes translates into a request for z/OS controlled resources, typically using JCA component connecting to a transaction server or Database Manager that runs on z/OS, or performing a file access, then the z/OS Security mechanism apply again, on the basis of the RACF userID which is associated to the request issued from WebSphere Application Server.

2. There are different Security models involved in what we call with the generic term of “WebSphere Application Server run-time environment” in:Figure 8-10

   – The z/OS JVM is located at the “bottom” of this security environments stack. It performs the optional Java 2 Security mechanisms on the basis on the code base and Java principal who is making requests to access Java resources controlled by the JVM security policies. The JVM also provides APIs, such as the JCE (Java Cryptographic Extension) cryptographic APIs or the JSSE (Java Secure Socket Extension) API, that applications can use to invoke cryptographic services.
   – The J2EE container of WebSphere Application Server is providing the J2EE security services and APIs that can be used by the J2EE applications components to proceed with callers authentication and resources access control.
   – The WebSphere Application Server itself has its own security services which are making up the underlying infrastructure for achieving secure communications, proper

access to user registries and authorization data, and the secure propagation of J2EE applications originated requests to the environments external to the J2EE container.
- Web Services Security comes as an additional layer when the users exploit the web services technology via the web services support embedded into WebSphere Application Server

Note that RACF profiles can be used to protect some resources specific to WebSphere Application Server for z/OS as explained in the next section.

8.3.1 The complete picture using SAF for authentication and authorization

Figure 8-11 summarizes the support provided by RACF, through the SAF interface, for WebSphere Application Server clients authentication and authorization. There are several conditions to be met for this to work:

- The J2EE principals to consider are actually RACF userIDs.
- The RACF administrator has defined the J2EE roles planned for the deployment of the application as profiles in the EJBROLE class. The list of J2EE principals in the role is actually the access list of the EJBROLE profile.
  If a user is in the access list of an EJBROLE profile, the user has that role. If a group is in the access list of an EJBROLE profile, users in that group have that role. If the EJBROLE profile has UACC(READ), all users have that role.

> **Note:** when using the SAF user registry, WebSphere Application Server recognizes users' membership to groups as per the users to groups connections in RACF.

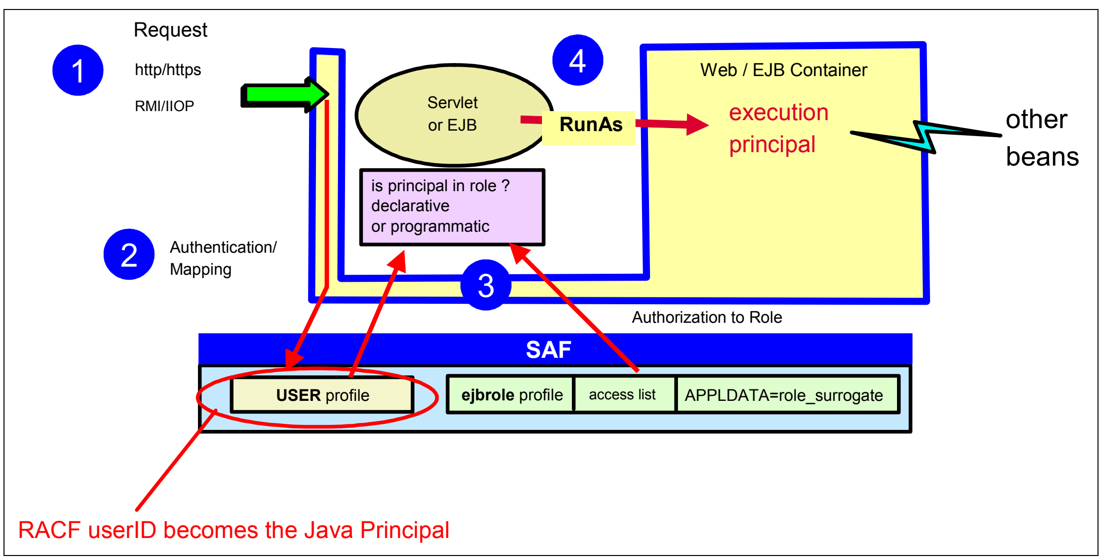


Figure 8-11 RACF support for J2EE authentication and authorization

The sequence of events is as follow:

1. The J2EE client sends a request to WebSphere Application Server for z/OS.
2. The caller authenticates by means supported by WebSphere Application Server JAAS login modules and RACF. These means are basic authentication with userID and password or passticket or digital certificate with RACF identity mapping. On a successful authentication, and mapping if required, the RACF userID becomes the authenticated JAAS principal. The JAAS subject contains also the groups the principal belongs to.

3. When it comes to check whether the principal is in the right role, WebSphere Application Server sends a request for verifying permission of the JAAS principal (RACF userID) to the EJBROLE profile.
4. The RunAs identity is assigned, as per the deployment descriptor, to the J2EE thread, with the capability of assigning a default identity for a given role, as explained below.

WebSphere Application Server supports a form of delegation where a user Identity can be represented as a J2EE role. For example, an application can be established to run with RunAs Role of roleX, as specified in its deployment descriptor, and WebSphere Application Server is also instructed to map a specific principal to roleX. With RACF support for the J2EE roles, the principal (that is the RACF userID) to be mapped to a role is specified in a field of the EJBROLE profile defined for this very role.

8.4 The IBM Tivoli Security portfolio

The IBM Tivoli portfolio of Security products is represented in Figure 8-12 and entails the following products (we are indicating the commonly used acronyms, as of the writing of the book, to designate these products):

- The IBM Tivoli Federated Identity Manager (TFIM) that provides Identity Federation and SOA Security. It is a standards-based, access control solution for federated single sign-on (SSO) and trust management in a web services and SOA environments.
- The IBM Tivoli Security Operations Manager (TSOM) and IBM Tivoli Security Compliance Manager that provide compliance and vulnerability assessment. Tivoli Compliance Insight Manager (TCIM) provides an enterprise with control for security compliance under the form of a graphical interface dashboard. It also monitors in details privileged users activities. TCIM operates on comprehensive auditing data that it collects from the systems it has connectivity to.
  IBM Tivoli Security Operations Manager (TSOM) is a real-time security information and event management (SIEM) platform designed to improve the effectiveness and efficiency of security operations and information risk management. TSOM centralizes and stores security data from throughout the heterogeneous technology infrastructure.
- The IBM Tivoli zSecure suite of products adds a user-friendly layer onto the RACF administrative interface, with additional audit, alert and monitoring capabilities (the zSecure suite is discussed in Section 8.1, “Complementing z/OS RACF” on page 208).
- IBM Tivoli Access Manager (TAM) is a policy-based, access control security solution for e-business and enterprise applications, featuring Web-based single sign-on and distributed Web-based administration. We are discussing TAM further in 8.4.2, “IBM Tivoli Access Manager (TAM)” on page 226.
- IBM Tivoli Identity Manager (TIM) provides a secure, automated and policy-based user management solution that helps effectively manage user identities throughout their life cycle, across both legacy and e-business environments. We are discussing TIM further in 8.4.4, “IBM Tivoli Identity Manager (ITIM)” on page 241.
- IBM Tivoli Directory Server (ITDS) is a set of LDAP-based data repository products that can be hosted by different platforms. They provide robust and advanced LDAP services and directories that can be exploited from any LDAP compliant client application.
  ITDS when running on a distributed platform can be accessed from z/OS hosted applications or middlewares. Likewise the ITDS for z/OS LDAP server can provide LDAP services to z/OS and non-z/OS hosted applications.
- IBM Tivoli Directory Integrator (ITDI) provides real-time synchronization between data sources so that enterprises can establish an authoritative, up-to-date, consolidated data

infrastructure. TDI is discussed further in 8.4.3, “IBM Tivoli Directory Integrator (ITDI)” on page 235.

- IBM Tivoli Key Lifecycle manager (TKLM) centralizes and automates the encryption key management process. It both provisions and serves encryption keys at the time of use to devices such as the storage units part of the IBM System Storage™ self-encrypting offerings. TKLM is discussed in 8.4.5, “IBM Tivoli Key Lifecycle Manager” on page 248.

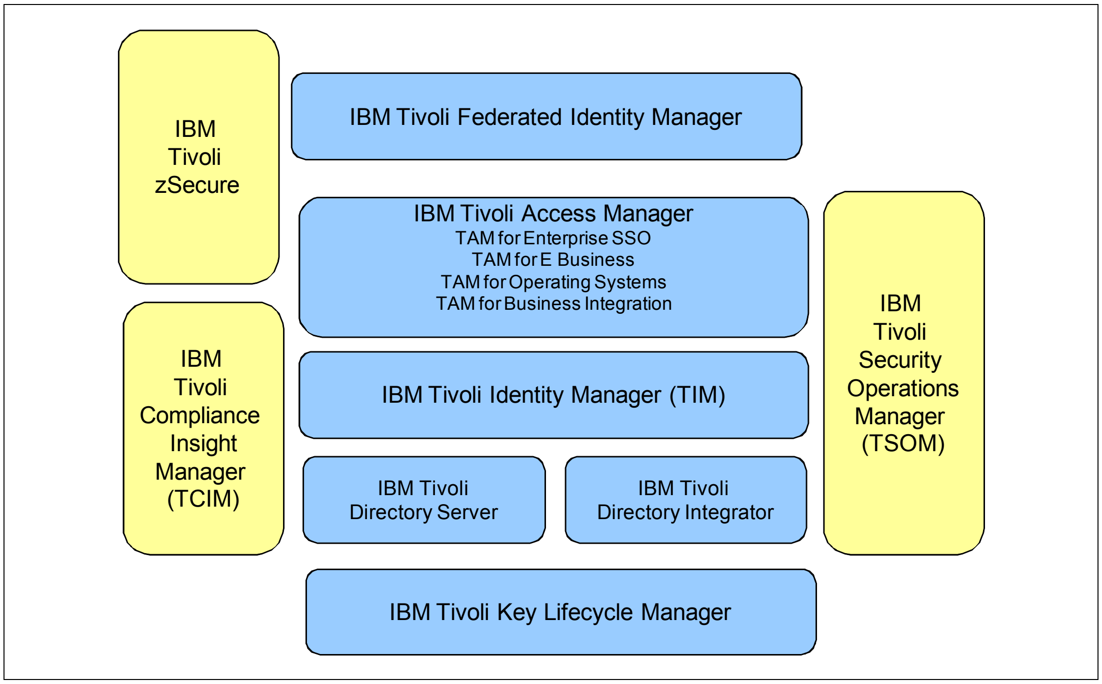


Figure 8-12 IBM Tivoli Security portfolio

8.4.1 Tivoli Security products that can execute on z/OS

Some Tivoli Security products (besides the zSecure suite which is, by design, intended to run on z/OS or z/VM) can be hosted in a z/OS instance and provide services to the different systems in the installation including z/OS. Running these products on z/OS is not a requirement however installations may want to consider the infrastructure robustness and centralized operations advantages that this implementation yields.

The products that can run on z/OS, as of the writing of this book, are:

- TFIM
- TIM
- ITDI
- ITDS
- TKLM

Note: we are focusing in this book on the capability of z/OS to host Security services and products.Products indicated as being supported by Linux can be hosted by Linux for System z and although not being installed in z/OS they can still run in a logical partition of System z.

8.4.2 IBM Tivoli Access Manager (TAM)

Further information on TAM, its use and its optimum placement in the Security infrastructure can be found in the IBM Redbook "Enterprise Security Architecture Using IBM Tivoli Security Solutions", SG24-6014.

The TAM family of products

The following products make up the Tivoli Access Manager family:

- Access Manager for e-business (TAMeb)
- Access Manager for Business Integration (TAMBI)
- Access Manager for Operating Systems (TAMOS)
- Access Manager for Enterprise Single Sign-On (TAM E-SSO)

> **Note:** although Tivoli Access Manager for Enterprise Single Sign-On bears the same naming of the components mentioned above, it does not share the same core components. Therefore the concepts and infrastructure that we are describing below do not apply to TAM E-SSO.

In the context of this book, we are focusing here on Access Manager for e-business (TAMeb) and Access Manager for Business Integration (TAMBI).

The TAM concept

TAM is an authentication and authorization solution for corporate Web, client/server, and existing applications. It implement access control to protected information and resources using a centralized, flexible, and scalable access control solution. It proves to be particularly suitable for building secure and easily managed network-based applications and e-business infrastructure.
TAM supports authentication, authorization, audit and logging, data security, and resource management capabilities.

In the simplest case, TAM's main purpose is to provide an authorization engine that returns a yes or no answer in response to a request by a particular user to perform a given action, or actions, on a given object. The term "authorization engine" is purposely used here as it pertains to a concept where the access control function is offloaded from the requestor's operating system. TAM is running on its own system (which can also very well be the requestor's operating system) and the authorization request and its response are carried over using TCP/IP between the requestor and the TAM server.
Note that TAM is not intended to be substituted to the existing native access control mechanisms implemented in operating systems, it rather aims at protecting resources meaningful only to instances of applications that run on different systems in the enterprise but still need to share the same access policy to these resources.

TAM contains 2 major components:

- A User Registry
  TAM maintains a User Registry where it stores information about each TAM- registered user. The main information stored here is the users TAM identity and their group membership. Other information includes a password, which can optionally be used for UserID and password authentication, and policy information, such as last password change.
  TAM is operating with an LDAP based user registry that has to be implemented by the user. See , "More details on the TAM implementation" on page 229 for a list of supported directory products.

- An Authorization framework
  This is used to make authorization decisions on behalf of applications written to use the TAM authorization API (the aznAPI). Decisions are made after querying the TAM authorization database. The database contains a hierarchical model of the objects, that is the representations of resources, being protected (the "Protected Object Namespace"). Access Control Lists (ACLs) are attached to the objects in the Namespace that define the security policy.

  Figure 8-13 gives a schematic view of the Protected Object Namespace. Objects to be protected are designated by entries in a hierarchical tree and ACLs are assigned to the object, stating which users or groups of users can access the object and for which types of access. Note that an ACL, once assigned to an entry is by default automatically propagated to all objects below the entry in the hierarchy.

  Objects are definition of resources usually at the installation level, such as servers, URLs, applications, messaging queues, etc. of which protection policy has to be shared by several systems in the installation.

  Entries can also be assigned additional protection directives under the form of a PoP (Protected Object Policy).

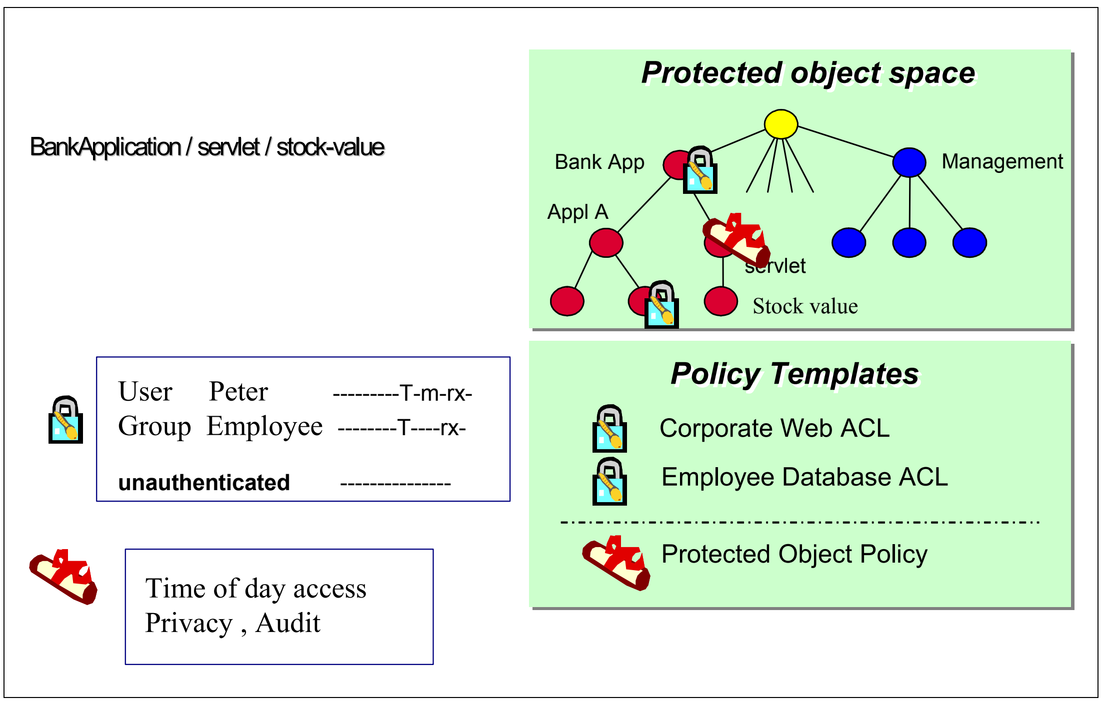


Figure 8-13 TAM protected object name space

A high level view of the TAM implementation is given in Figure 8-14. The core part of TAM, that is the "Policy Server", does not itself protect application-owned resources. It maintains the user registry and the Protected Object database (also called the Object/ACL database or the authorization database).

Application resources are protected using "Policy Enforcers" that reside between the front-end application that the user interacts with and the back-end application, or system, that contain the resources that require protection. The policy enforcer can be seen as a "plug-in" program to be installed in the path between the requesting application and the application or system that hosts the resource. IBM provides policy enforcers and users can also design policy enforcers of their own.

When the policy enforcer receives a request from the front-end application it uses the aznAPI to query the authorization database. It sends via TCP/IP information about the user making the request, the action being requested and the object to be accessed. This results in a yes or no response which indicates whether the request should be accepted or rejected. The response can also contain data pertaining to a PoP, if there is one.

The aznAPI is an Open Group standard for authorization request API intended to be application and platform neutral, isolating requesting applications from the complexity of the access control decision making processes.

Note that all communications between the TAM components can be protected with SSL/TLS.

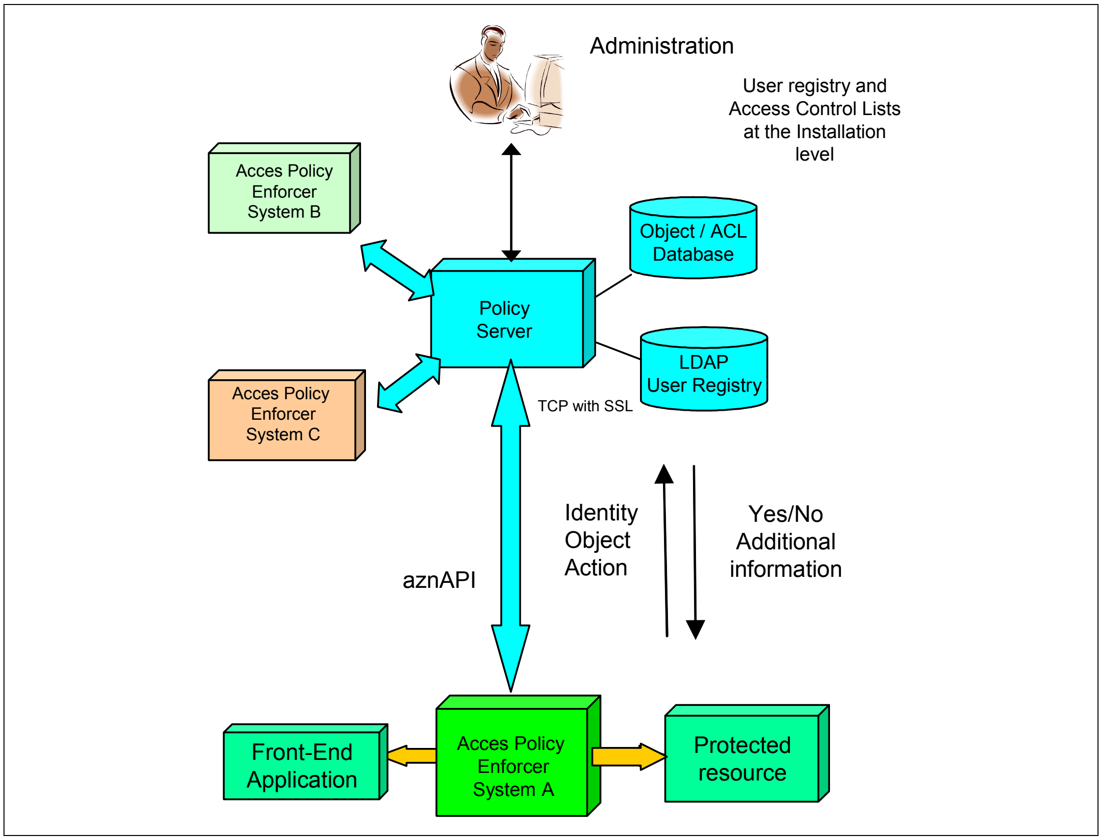


Figure 8-14 The IBM Tivoli Access Manager concept

As shown in Figure 8-14, the TAM Policy Server can serve requests coming from policy enforcers on different systems, thus sharing the access policy in the Protected Object database between applications on these systems.

In some cases installations may install a Policy Enforcer that acts as a proxy agent, that is an independent entity from the front-end and back-end systems. In other cases the Front-end, Back-end and Policy Enforcer might be different parts of the same application.

As a summary, TAM implementations are achieving the following:

- Maintain a central user registry
  - Users and groups
  - Authentication information
- Maintain a model of the Protected Objectspace
  - Hierarchically organized

- Define permitted actions on objects
  - Uses Access Control List templates
  - These are attached to entries in objectspace
- Provide an API for making Authorization queries
- Provide APIs for TAM administration

More details on the TAM implementation

We remained at the conceptual level in the previous sections; we are now providing some more implementation-related details.

User registry

The following LDAP based directories are supported by TAM, as of the writing of this book:

- The IBM Tivoli Directory Server on non-z/OS or z/OS systems. The ITDS for z/OS LDAP server is used with the TDBM or LDBM backend
- Lotus® Domino®
- Microsoft Active Directory
- Microsoft Active Directory Application Mode (ADAM)
- Novell eDirectory
- Sun Java System Directory Server

Note: when using the z/OS LDAP server as the TAM user registry, the schema files supplied in z/OS (schema.user.ldif and schema.IBM.ldif) already include the specific TAM user registry object classes and attribute types.

The TAM Policy Server and the additional components

The Access Manager environment requires certain basic capabilities for administrative control of its functions. Management facilities are provided through the following base components:

- The Policy Server, which supports the management of the authorization database and its distribution to Authorization Services.
- A Policy Proxy Server, which provides a mechanism for Policy Enforcers to access Policy Server functionality without a direct connection to the master Policy Server. The Policy Proxy Server is using caching techniques with a local copy of the Policy Server access control database.
- The pdadmin utility, which provides a command line capability for performing administrative functions such as adding users or groups at the TAM policy server.
- The Web Portal Manager, which provides a browser-based capability for performing most of the same functions provided by the pdadmin utility.
- The administration API, on which the pdadmin utility and the Web Portal Manager are built, enables execution of program initiated level administration tasks and queries.

As of the writing of this book, the TAM Policy Server can run on the following operating systems:

- AIX
- HP Unix
- Linux
- Sun Solaris
- Windows

The Resource Managers

Resource managers are program components that provide Access Manager authorization support for specific application types. The resource manager is responsible for the enforcement of the security policy within an Access Manager environment. The resource manager uses the policy enforcer to call the Tivoli Access Manager authorization service with the credentials of the user making the request, the type of access desired, and the object to be accessed. The resource manager takes the recommendation of the authorization service, performs any additional verification actions, and ultimately either denies the request or permits the request to be processed.

We are describing in the next sections the WebSeal and WebSphere MQ TAMBI resource managers.

TAM for e-business (TAMeb) - WebSEAL reverse proxy

A real example of the front-end, policy enforcer, back-end arrangement is the way in which TAM can be used to protect Web Resources. In this case the front-end is a web browser and the back-end is one or more web enabled servers. The IBM resource manager and policy enforcer, known as WebSEAL, acts as a reverse proxy sitting between the two. This is depicted in Figure 8-15.

Instead of making requests directly to the WEB servers, the client is forced (using a firewall for example) to make requests to WebSEAL. When WebSEAL receives a request it first authenticates the user and then makes a decision as to whether that user is allowed to access the resource they are requesting. Note that WebSeal has a direct connection to the Policy Server LDAP user registry in order to proceed directly with authentication and collection of meaningful user characteristics.

If the user is authorized then WebSEAL passes the original request to the back-end server and then passes the response back to the user. If the user is not permitted an error page is returned to the user and the request is discarded. In the simplest case, neither the browser or the back-end server have to be modified to achieve this functionality.

All the TCP/IP connections shown in Figure 8-15 can be secured with SSL/TLS

As of the writing of this book, WebSEAL can run on the following systems:

- Linux
- Windows
- UNIX

IBM also provides other policy enforcers such as a plug-in for the WebSphere Edge Server or for IBM or vendor provided web servers. The web server plug-in can be installed on the following web server products:

- Apache Web Server on AIX, Linux on System z, and Solaris
- IBM HTTP Server on AIX, Linux on x86, Linux on System z, Solaris and Windows 2003.
- Internet Information Services on Windows 2003
- Sun Java System Web Server on AIX and Solaris

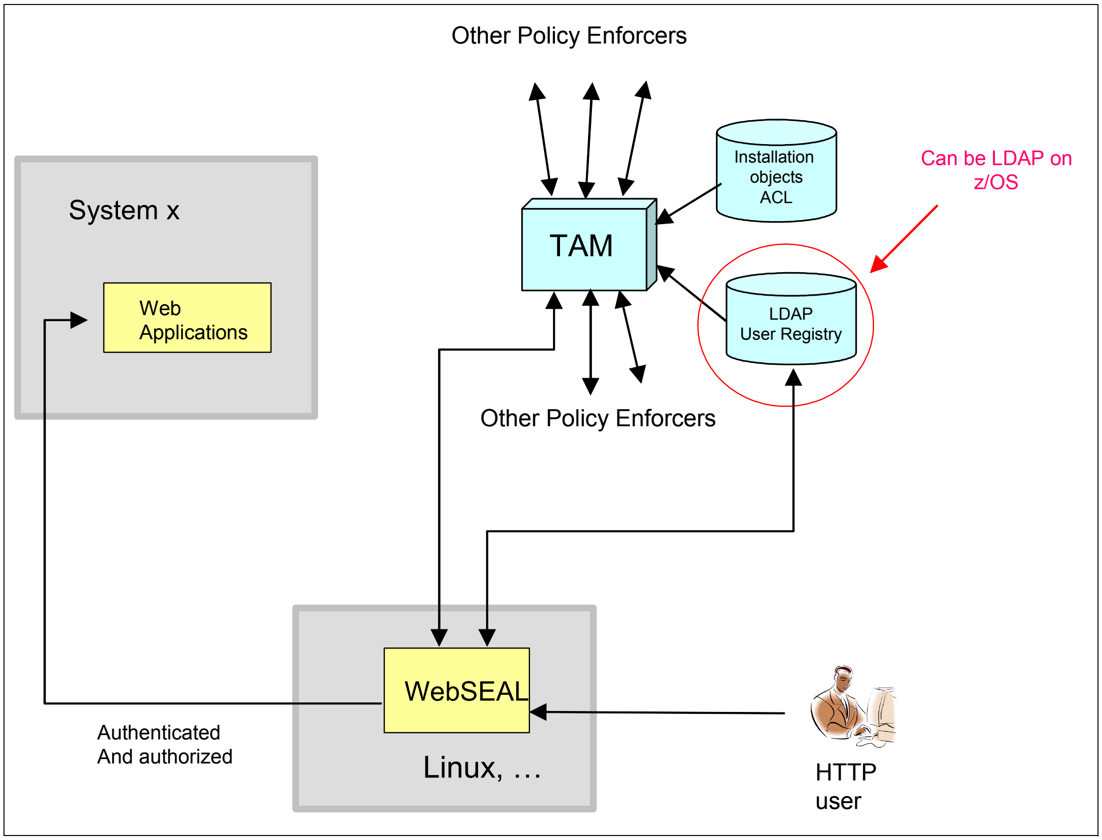


Figure 8-15 TAM and the IBM WebSeal authorization reverse proxy

Junctions

The back-end services to which WebSEAL can proxy are specified via junctions. A junction is a TAM configuration entity which defines a set of one or more back-end web enabled servers that are associated with a particular URL.

Single sign-on

WebSEAL supports several mechanisms for supplying a junctioned server with the identity of the authenticated user, including:

- Providing the user's identity via HTTP header values, which can be read and interpreted by the junctioned server.
- Insertion of an HTTP basic authentication header to provide the junctioned server with login information for the user, including a password. Optionally, this basic authentication header can permit login to the junctioned server with a different identity from the one for the user who is logged in to WebSEAL.
- For junctions that support it (for example, WebSphere Application Server and Domino), insert a Lightweight Third-Party Authentication (LTPA) cookie identifying the user into the HTTP stream that is passed to the junctioned server.
- For junctions that support it, (WebSphere Application Server), the use of a Trust Association Interceptor Plus (TAI++) to forward Tivoli Access Manager credential information and establish trust between WebSEAL and backend application server.

TAM auditing - Common Auditing and Reporting Services (CARS)

Tivoli Access Manager includes IBM's Common Auditing and Reporting Service (CARS) platform, which provides a consistent way to audit and report on data. CARS automates the collection of audit data and provides the ability for enterprises to centrally view and report audit data that are critical for compliance needs, and thus provides a more efficient audit process.

For details on TAMeb auditing and CARS see the “Tivoli Access Manager for e-business 6.0 Auditing Guide”, SC32-2202.

TAM for Business Integration (TAMBI)

Tivoli Access Manager for Business Integration operates in conjunction with IBM Tivoli Access Manager for e-business. The combination of these software applications is provided as a security solution for IBM WebSphere MQ products. TAM BI provides:

- Data protection for MQ messages
  That is secure sensitive or high-value messages, provide integrity and privacy protection for data as it flows across the network and while it is in a queue, detect and remove rogue or unauthorized messages before they are processed by a receiving application.
- Access control for IBM MQ Resources
  That is control which users have access to specific queues, generate detailed audit records showing which messages were expressly authorized and encrypted.
- Central Management of Security Policy Definition and Enforcement
  Define authorization and data protection policies centrally for IBM WebSphere MQ resources that get and put messages to queues, using a Web browser or command line.
- Support for existing and new MQ applications
  Secure existing off-the-shelf and customer-written applications for IBM WebSphere MQ.

MQ uses the Operating System identity that an MQ application is running under to make it’s authorization decisions. MQ relies therefore on the local operating system for authentication and has locally administered authorization.

Although MQ has the capability of doing data encryption and integrity using the security or message exits, these only provide data protection when a message is traversing the network. Channel exits do not protect the messages while they are on the queues.

TAMBI exploits the X.509 digital certificate technology to globally identify users and processes. In most cases this is mapped from the OS Identity. If “real” users are using MQ applications locally then they can be prompted for a digital certificate identity which is independent of the Operating system user. Also, the operating system definitions for the same person on multiple MQ systems can be mapped to a single identity in TAMBI. This means that changes to a users access rights made centrally are immediately enforced on every MQ system in the secure domain. Each TAMBI user is mapped to a digital certificate identity. This means also that this infrastructure can be directly exploited for messages to be encrypted for the intended recipient(s) and/or signed by the sending user or process.

A high level representation of the TAMBI implementation is given in Figure 8-16. When TAMBI is used, it intercepts requests made by the MQ application. This is done in various ways depending on the platform.

Since all requests made by the application now pass through TAMBI it can completely control what actions the application can perform. It can also make changes to the messages sent by the application to include data signing and data encryption.

TAMBI uses the services of TAM in order to make authorization decisions. TAMBI uses digital certificate technology and symmetric key algorithms to perform the signing and encrypting required.

The Object Authority Manager (OAM), which is the native MQ authorization engine is redundant with TAMBI and is not expected to be used. Likewise the Message Channel Agent (MCA) code exits, which are available to provide authorization of exchanges between MQ systems or to encrypt/decrypt the message data as it flows over the network are not expected to be exploited when TAMBI is in use.

The user setup the following policies in the Policy Server authorization data base:

- An Access Control Policy, which gives permissions for the PUT and GET operations.
- A Data Protection Policy that specifies the Quality of Protection (QoP) level. The QoP level can be “none”, “data integrity checking” or “data integrity checking and data privacy”

Auditing can be specified for a queue, then all OPEN, PUT, GET actions are recorded and the audited Information includes:

- TAM identity of application or user
- Date and Time
- Encryption and Signing algorithms used
- Sender (if the message is digitally signed)
- MQ Message ID

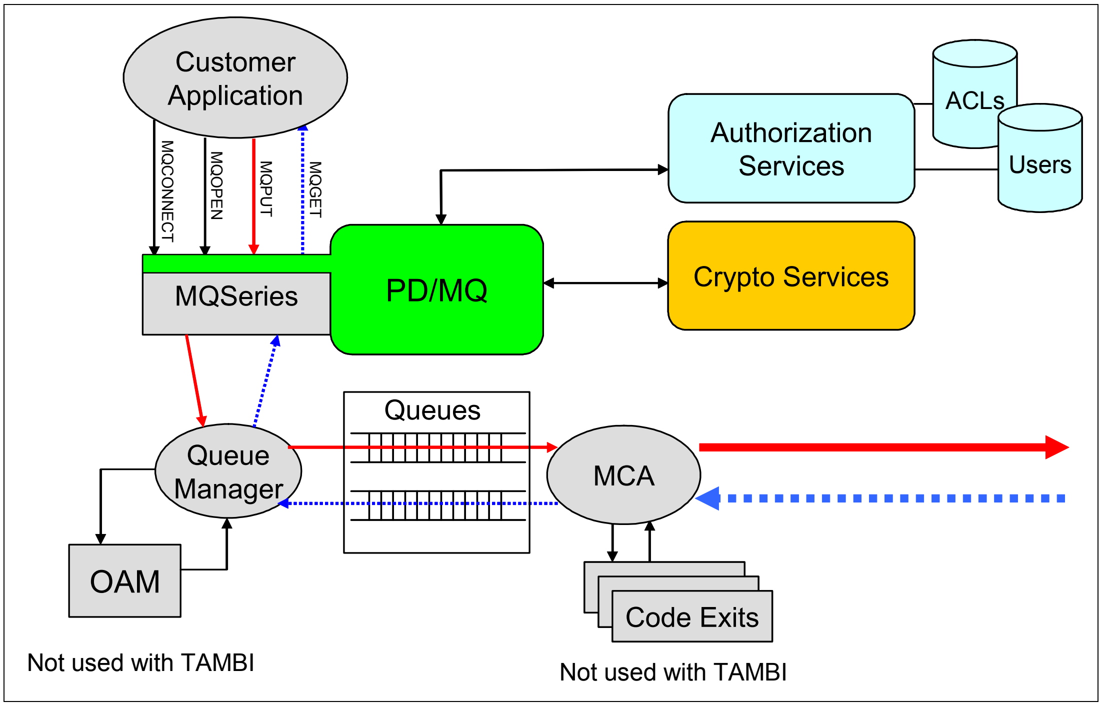


Figure 8-16 TAM BI high level view of implementation

TAMBI for z/OS

TAMBI for z/OS is marketed, as of the writing of this book, as “Access Manager Business Integration Host Edition” (AMBIHE). Further details on AMBIHE can be found in “IBM Tivoli Access Manager for Business Integration, Host Edition Administration Guide”, GC32-1122.

As for the other TAMBI implementations, AMBIHE is designed to enforce two specific access rights which are whether an application is authorized to put and/or get messages to a queue. AMBIHE also supports the following options for data protection policies:

- NONE — No data protection.
- INTEGRITY — Sign message data to allow verification.
- PRIVACY — Sign and encrypt message data for integrity and confidentiality.

Note that the AMBIHE cryptographic functions exploit the System z cryptographic hardware coprocessors, whenever possible.

The auditing of access to protected resources is also part of each policy. If auditing is enabled, GTF (Generalized Trace Facility) records are provided that document the success or failure of attempts to open and close queues and put and get messages. The specific audit options are:

- NONE — Records any unsuccessful intercepted API operations.
- Any other specified option — Records OPEN, CLOSE, PUT, and GET operations on protected WebSphere MQ queues.

AMBIHE uses public and private keys to perform its data-protection functions. Certificates are managed by RACF and associated with RACF userids. AMBIHE can utilize digital certificate credentials generated by most popular third-party Certificate Authorities, including VeriSign, Entrust, Baltimore, and Netscape, in addition to the self-signed certificates it can generate itself. These credentials are stored and secured using RACF.

Figure 8-17 is a high level view of the AMBIHE implementation. AMBIHE comes with the Policy Director Authorization Services (PDAS) component that has to be installed in z/OS. It comprises a z/OS UNIX version of the TAM pdacld (Policy Director ACL daemon) remote authorization engine. pdacld maintains a replica of the authorization database on the z/OS host but, rather than providing authorization services to remote clients as in the distributed environment, it mirrors the policy information to a cache in z/OS where it can be accessed by the local resource managers.

When PDAS is installed, the SAF interface is extended to provide support for the following SAF callable services that are specific to AMBIHE:

- aznCreds - Which is a z/OS version of the azn_id_get_creds and azn_creds_delete aznAPI functions.
  The identity supplied with the call can be a TAM user identity or can be a RACF userID in a z/OS ACEE, that is a security context, control block.
- aznAccess - Which is the z/OS version of the azn_decision_access_allowed aznAPI functions.
  The supplied identity can be, as above, a TAM user identity or a RACF userID in a z/OS ACEE, that is a security context, control block.
  Protected resource attribute values and protected object policy (POP) values may also be optionally returned
- R_cacheserv - This callable service allows to store and retrieve information from a cache, actually a data space, managed by RACF. In the context of PDAS it is exploited by pdacld to mirror the Policy Server authorization data base.
- R_proxyserv, which exploits the z/OS LDAP server PC-callable support to retrieve TAM user registry information.

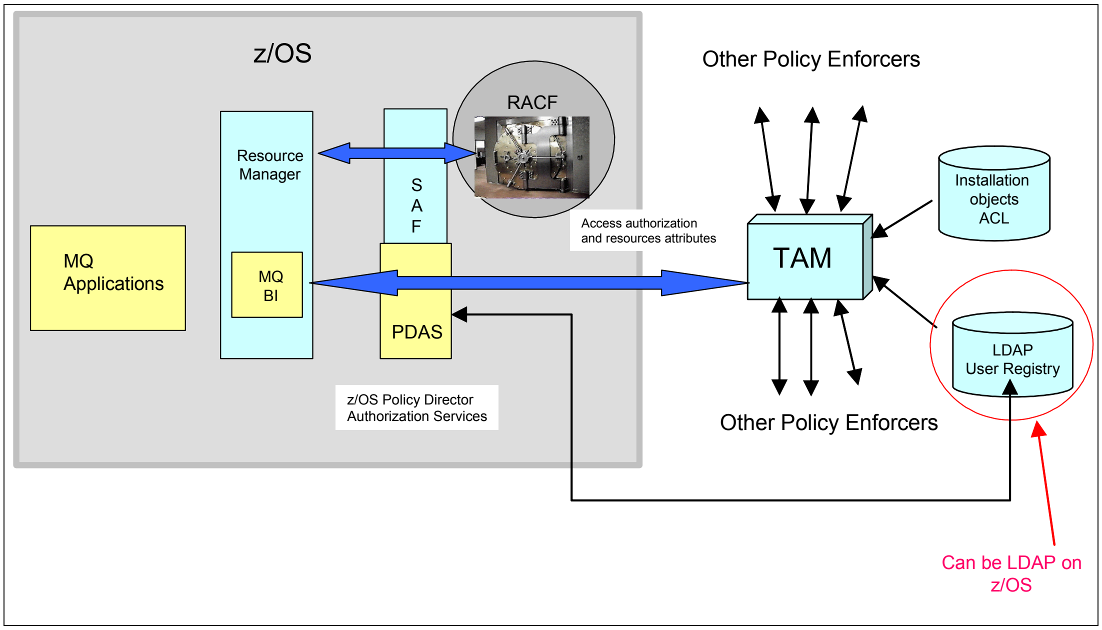


Figure 8-17 High level view of AMBIHE implementation in z/OS

In addition to the auditing data collected at the TAM Policy Server level, PDAS operations are audited and recorded in SMF type 180 and 80 records. Details on the SMF records contents can be found in “z/OS Security Server RACF Macros and Interfaces”, SA22-7682, for SMF type 80 records and “IBM Tivoli Access Manager for Business Integration Host Edition Administration Guide”, GC32-1122 for SMF type 180 records.

8.4.3 IBM Tivoli Directory Integrator (ITDI)

Further information on ITDI, its use and its optimum placement in the Security infrastructure can be found in the IBM Redbook “Robust Data Synchronization with IBM Tivoli Directory Integrator”, SG24-6164.

ITDI is an open architecture based solution to synchronizes and exchanges information between applications or directory sources of many different types and data formats. It operates with an AssemblyLine methodology that builds a compound information object from connected information sources, performs modifications on received data, or creates new entries altogether and adds/updates/deletes the new information object to the assigned destinations.

It is particularly well adapted to:

- serve as a flexible synchronization layer between an installation’s identity structure and the application sources of identity data, thus eliminating the need for a centralized data store
- help ease the process of deploying an enterprise directory solution by connecting to the identity data from the various repositories throughout the organization
- manages data across a variety of repositories providing a consistent directory infrastructure that can be exploited by a wide variety of applications.
- create authoritative data spaces needed to expose only trustworthy data to advanced software applications such as Web services

Data Integration-relevant concepts

- Data sources and targets - These are the data repositories, systems and devices that feed data into, or get data from the dataflows.
- Data flows - These are the threads of communications and their content and are usually drawn as arrows that point in the direction of data movement. Each data flow represents a dialogue between two or more systems.
- Events - Events can be described as the circumstances that dictate when one set of data sources communicates with another. One example is whenever an employee is added to, updated within, or deleted from a user registry.

Conceptual view of ITDI operations

The conceptual view of ITDI operations is shown in Figure 8-18, where a schematic "assembly line" shows the collection of data coming from and LDAP and ODBC directory, which are eventually processed and reformatted as a JMS message and a Lotus Notes® database update.

ITDI includes connectors to support numerous protocols, parsers to interpret and translate information from a byte stream into a structure information object, and hooks to enable the definition of certain actions to be executed under specific circumstances. It also exploits an Event Handler framework that adds to the flexibility by providing the ability to wait for, and react to, specific events that have taken place in the infrastructure

ITDI can typically be used as a solution to:

- continuously maintain records in one or more databases, based on information in other data sources such as files, directories and databases.
- migrate data from one system to another, or to synchronize with pre-existing data where systems cannot be replaced or shut down.
- automatically transform files from one format to another.
- add supplementary identity data to LDAP directories when deploying user registry, provisioning, and access control solutions.
- react to changes to data (such as modification, additions, and deletions) in the infrastructure and to drive this information to systems that need to know about it.
- integrate geographically dispersed systems with multiple choices of protocols and mechanisms; such as MQ, HTTP, secure e-mail and Web Services.

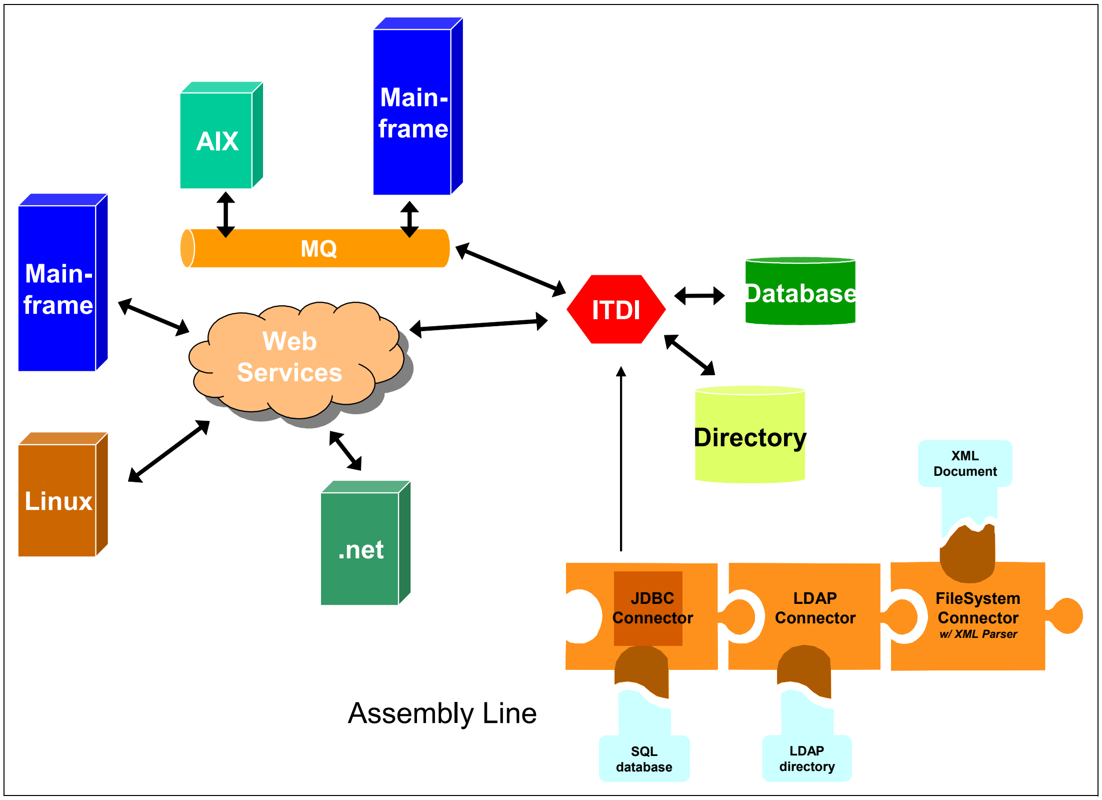


Figure 8-18 ITDI concept of operations

Supported operating systems

ITDI can execute on the following operating systems:

- AIX
- HP Unix
- Linux
- Sun Solaris
- Windows
- i5/OS®
- z/OS

Synchronizing data with ITDI

When implementing a synchronization solution, the result is an environment where shared data looks the same for all consuming applications. This is because changes are propagated throughout the synchronized network of systems, molded in transit to fit the needs of each consumer. Each data source is kept up-to-date, maintaining the illusion of a single, common repository. Each application accesses its data in an optimal manner, utilizing the repository to its full potential without creating problems for the other applications.

Tivoli Directory Integrator-based synchronization solutions are typically deployed in one of the three following manners, although combinations are also frequently used to enable the various data flows that entire solution requires:

- Batch - In this mode ITDI is invoked in some manner (through its built-in timer, command line or the Tivoli Directory Integrator API), and is expected to perform some small or large job before either terminating or going back to listening for timer events or incoming API calls. This is often used when synchronizing data sources where the latency between change and propagation is not required to be near real-time.

- Event - ITDI can accept events and incoming traffic from a number of systems, including directory change notification, JMX, HTTP, SNMP, and others. This mode is typically used when ITDI needs to deal with a single, or a small number of data objects.
- Call-reply - This is a variation of the event mode, but the difference is that the originator of the event expects an answer back. IBM products use the ITDI API to call Tivoli Directory Integrator, and solutions in the field often use HTTP, MQ/JMS and Web services to invoke an ITDI rule and get a reply back.

Data synchronization security

It is important to identify the security requirements of the data that users will be synchronizing. Most of the requirements become apparent while identifying the nature of the data and planning the data flows. The following two questions can be asked to further identify these requirements.:

1. Does the entire data transmission between sources have to be secure for all data? Solutions for securing the data transmission involve utilizing technology such as SSL/TLS protection of the communications.
2. Are there specific data attributes that must be encrypted? Many times this involves the password attribute. ITDI provides several encryption methods and the ability to encrypt any attribute. It is not limited to just the password attribute.

Non-data synchronization scenario

While ITDI can deal with a large number of synchronization scenarios, its core is a general purpose integration engine that can be used by other systems in real-time, providing these systems with quite interesting capabilities. An example of such a deployed solution might be: a mainframe application sends MQ messages that ITDI picks up, then accesses other data systems in the enterprise, performs some operations and transformations on the set of data and responds back through MQ to the mainframe.

ITDI component structure

ITDI is a Java based system where the core system provides most of the functionality. The core handles log files, error detection, dispatching, and data flow execution parameters.

There are four main types of components, which serve to provide an abstraction layer for the technical details of the data systems and formats, that users are to exploit: AssemblyLine, Connectors, Parsers, Function Components, and EventHandlers.

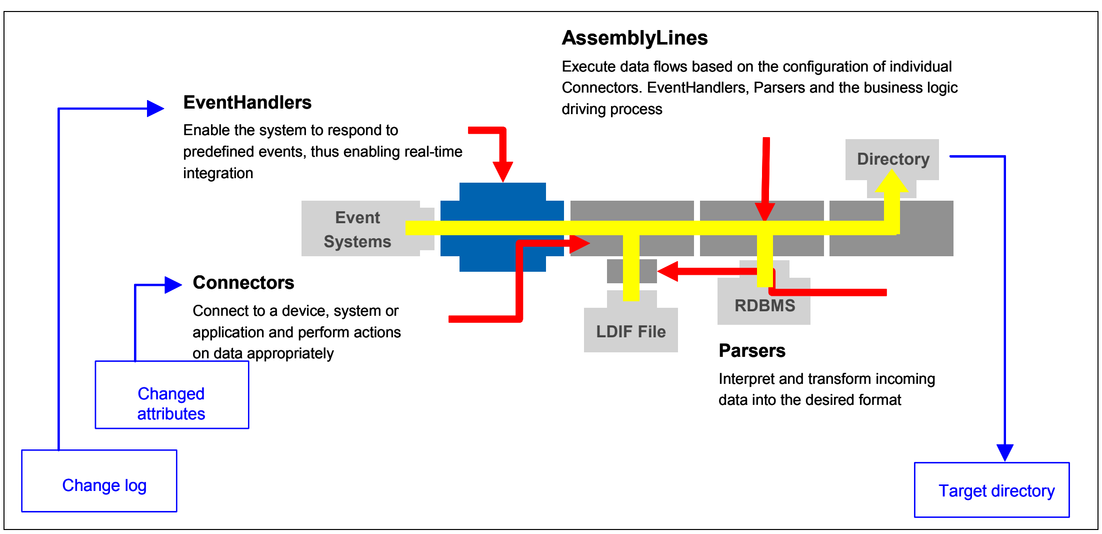


Figure 8-19 ITDI components

Assembly lines

An AssemblyLine (AL) is a set of components strung together to move and transform data. It is the 'unit-of-work' in ITDI and typically represents a flow of information from one or more data sources to one or more targets. Data to be processed is fed into the AL one Entry at a time, where these Entries carry Attributes with values coming from directory entries, database rows, emails, Notes documents, records or similar data objects. Each Entry carries Attributes that hold the data values read from fields/columns in the source system. These Attributes are renamed, reformatted and/or computed as processing flows from one component to the next in the AL. New information can be 'joined' from other sources and all or parts of the transformed data can be written to target stores or sent to target systems as desired.

Connectors

Connectors are providing access to data while “abstracting away” the details of some system or store, giving the user the same set of access features.
There are basically two categories of Connectors:

- The first category is where both the transport and the structure of data content is known to the Connector (that is, the schema of the data source can be queried or detected using a well known API such as JDBC or LDAP).
- The second category is where the transport mechanism is known, but not the content structuring. This category requires a Parser to interpret or generate the content structure in order for the AssemblyLine to function properly.

> **Note:** when running on z/OS, ITDI can use the z/OS TSO Command Line Function component to interact directly with z/OS components such as RACF.

Parser

Unstructured data, such as text files and bytestreams coming over an IP port, can be handled by passing the bytestream through one or more Parsers. ITDI is shipped with a variety of Parsers, including LDIF, Directory Services Markup Language (DSML), XML, comma-separated values (CSV), SOAP, and fixed-length field. As with Connectors, the users can extend and modify these, as well as create their own.

Events handler

They enable the system to respond to pre-defined events, thus enabling real-time integration.

Hooks

Hooks enable developers to describe certain actions to be executed under specific circumstances or at any desired points in the execution of an AssemblyLine.

Scripts

The ITDI implementation gives the ability to extend most of its integration components, functions, and attributes through scripts or Java. Scripting can be installed anywhere in ITDI to add or modify the components of an AssemblyLine.

Function components

A function component is an AssemblyLine wrapper around some function or discreet operation, allowing it to be dropped into an AssemblyLine as well as instantiated, or invoked, from a script.

Example of use case

Figure 8-20 is an example of an ITDI use case, where an installation wants to synchronize, both ways, an LDAP user registry and the RACF data base. In this example the LDAP user registry is actually the z/OS LDAP server with the TDBM or LDBM backend. The access to the RACF data base is also achieved with the LDAP protocol via the z/OS LDAP SDBM backend for a discussion of the z/OS LDAP and its backends).

We also show in Figure 8-20 the LDAP user registry as being actually a TAM policy server user registry.

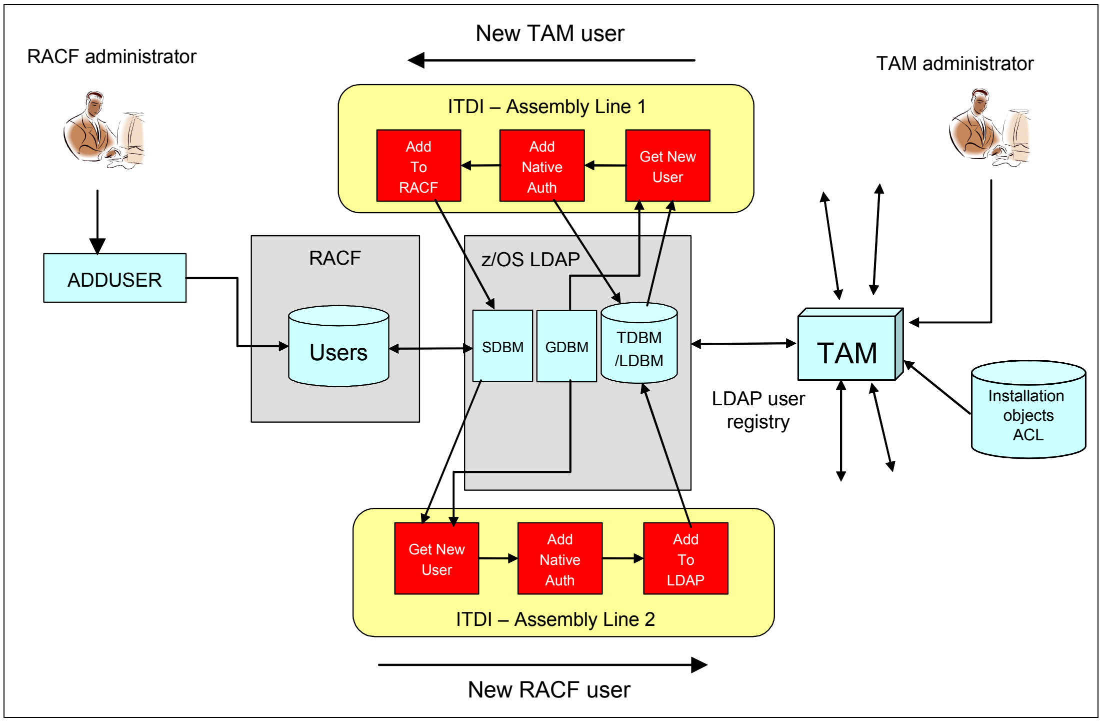


Figure 8-20 ITDI use case example

A new user is created in the TAM user registry

- On detection of the LDAP directory entry creation event, ITDI starts the AssemblyLine 1.

> **Note:** the event detection can be done through the GDBM backend of the z/OS LDAP server

  AL 1 fetches the user data from the LDAP directory and also establishes in the newly created user entry the object class and attribute necessary to exploit the native authentication feature of the z/OS TDBM or LDBM backend.

- The ITDI AL1 then reformats the user attributes and their value fetched from the TAM user registry into object classes and attributes that are supported by the z/OS LDAP backend.
- The ITDI AL1 creates a RACF profile for the new user through the SDBM backend, by using the LDAP protocol and commands.

A new user is created in the RACF data base

- On detection of the RACF “directory entry” creation event (actually the creation of a new USER profile, but as seen through the SDBM), ITDI starts the AssemblyLine 2. AL 2 fetches the user data from the RACF database.

> **Note:** The event detection is done through the GDBM backend of the z/OS LDAP server.

- AL 2 reformat the RACF user data into object classes, attributes and values that match the LDAP schema of the TAM user registry. This includes the attributes for the TDBM or LDBM z/OS LDAP native authentication.
- AL 2 creates the new user entry in the TAM user registry.

Auditing

Although providing functional logging and tracing facilities, ITDI does not provide built-in auditing functions in the Security meaning of the term, however AssemblyLine(s) can be assembled that would feed audit or log files.

8.4.4 IBM Tivoli Identity Manager (ITIM)

Further information on ITIM, its use and its optimum placement in the Security infrastructure can be found in the IBM Redbook “Identity Management Design Guide with IBM Tivoli Identity Manager”, SG24-6996.

ITIM provides a secure, automated, and policy-based solution that helps effectively manage user privileges across heterogeneous IT resources. It implements:

- Comprehensive request-based provisioning for end users, managers, or delegated administrators to easily request (with approval workflow) user access to roles, accounts or fine-grained Access Entitlements such as shared folders and Web portlets on distributed systems.
- Streamlined self-service interface for end users that can be easily customized by the using organization and integrated with corporate portals, to help improve user productivity and reduce administrative cost.
- Comprehensive reporting and auditing facilities.

The entities managed by ITIM

In order to implement the concepts involved in Identity Management, the ITIM functional architecture is based on the following entities that are to be managed:

Users

Users are persons, in the common meaning of the term, of which identity has to be managed across an organization by ITIM. Note however that a user might not be part of the organization itself but, for business reasons, needs to have a registered identity in the organization.

Accounts

The term “account” here refers to the collection of information, mainly dictated by the technology of the user registry in use, that together make up an identity that the technology can exploit.

Attributes

These are the additional meaningful information associated to the user that the exploiting organization wants to find in the account.

Passwords

All accounts have passwords. Account passwords can be centrally managed by their owners or administrators using the Identity Manager Web interface. Passwords are managed in a secure environment. ITIM provides two options for the passwords it manages: passwords can be synchronized or not.

Password synchronization is the process that helps users maintain a single password that is subject to a single password policy, and changes on a single schedule across multiple systems. The synchronization can be applied to all accounts associated with a user or with selected accounts. This is mostly one-way synchronization as Identity Manager sets the password and pushes it to the managed targets. Note that there are few target systems where local password changes can be reflected back to ITIM so that they can be propagated to the other managed systems - The ITIM RACF agent does not provide reverse password propagation.

When the password synchronization property is enabled, there is only one global password that a user uses for all the applications managed by Tivoli Identity Manager. If an account is being set up for first time, password synchronization does not apply; there is only one account, and therefore, one password.
If a user has more than one account, password synchronization affects the following user's or administrator's actions:

- Creating a new account
- Changing a password for an existing account
- Provisioning an account
- Resetting an expired or forgotten password for an existing account
- Restoring an account that was suspended

Administrators can always change passwords for selected accounts by using the service account management, but this would imply that a user will have different passwords across platforms or applications.

There is a process where Identity Manager generates a random password. This can be displayed to an administrator or mailed to a user.

Notes:
1- when RACF is part of the ITIM managed system, the password length and syntax specified in the ITIM password policy must conform to the RACF rules and syntax.
2- The ITIM RACF adapter does not reflect to ITIM a RACF password change that is performed at RACF. If password reconciliation and re-synchronization is required in that case, the installation has to setup the infrastructure described in "Complementing ITIM with ITDI" on page 246 to feed ITIM back with the new RACF password.

Group memberships

The user's membership to a group is granted by ITIM by specifying a group attribute in accounts

Managed systems and applications

ITIM manages users on many managed systems. These include operating systems, such as many flavors of UNIX and Windows servers, and applications, such as databases and business applications.

ITIM management entities

Identity Manager uses the following entities in its identity management processes:

Organizational tree and roles

The organizational tree (org.tree) defines the structure for the organization that ITIM is being deployed into in order to provide the Identity Management services. The tree consists of:

- An organization - there is normally only one organization at the top of the organizational tree.

- One or more locations - These are locations defined by the business.
- One or more organizational units - These are teams or departments as defined by the business.
- One or more business partner organizations - These are business partners as defined by the business.
- One or more administrative domains: These are Identity Manager groupings for administration.

There is no technical difference between locations, organizational units, or business partner organizations. They just use different icons in the administrative interface and allow the org.tree to be modelled as the administrators wish.

All people are attached to the org.tree at a single point. A policy is attached to points in the org.tree. This policy can control the provisioning of accounts, account user ID generation and password strength. Thus, there can be a corporate-wide password policy defined at the organization level in the organizational tree and a specific password policy that applies to a specific branch or department of the organization.

Identity Manager roles

Identity Manager roles, or “organizational roles”, are also attached to points in the organizational tree, defining the scope of specific access rights within the Identity Manager product.

Users are assigned to roles based on responsibilities defined within the organization and role members are provisioned to resource(s) via a Provisioning Policy.

> **Important:** ITIM does not create or delete groups on managed target systems, nor does it manage ACLs or resource access on the managed targets. This must be performed by the local administrators or application owners using the native system or application tools.

Policy

Identity Manager employs four types of policy: provisioning policy, password policy, identity policy, and service selection policy.

- Provisioning policy - A provisioning policy is used to define what accounts can be created for a user and on which target systems. It can, optionally and automatically, create the accounts on those systems. It can also be used to define a specific approval workflow process that has to be applied to the accounts.
- Password policy - A password policy sets parameters that all passwords must meet, such as length, type of characters allowed and disallowed, and so on.
- Identity policy - An identity policy defines how a user's ID is created. Identity Manager automatically generates userIDs from the identity policy.
- Service selection policy - A service selection policy extends the ability of provisioning policies by provisioning a specific instance of a service based on personal attributes.

Workflow

ITIM can be setup to execute workflows, that is internal automated processes, that can be used, for example, to customize account provisioning using entitlement workflows, and provide automated or semi-automated life cycle management, such as adding, removing, and modifying people and accounts in ITIM using operation workflows. Workflows can also be used to get approval before starting a provisioning process.

Audit logs

ITIM logs the events that occur during specific transactions such as (this list is not comprehensive):

- Add, modify, suspend, restore, delete person.
- Add, Modify, Suspend, Restore, Delete Account.
- Change Password.
- Add, modify, delete provisioning policy.

Reports

ITIM provides several types of reports that can be used for administration and tracking of activities.

ITIM key functions and logical architecture

A summary of the ITIM key functions is:

- User self-service - for user to maintain their account passwords and other personal information
- Password management - Through ITIM, users and administrators can centrally manage and synchronize their passwords across all their accounts (that is across systems and repositories where users are registered).
- Manage people and accounts - System administrators have the ability to manage an organization's employees (people) from a central location.
- Apply policy to people and account management - Policies are used to determine and enforce compliance of people and their accounts managed by Identity Manager. They are also used as the basis for automation of account provisioning and de-provisioning, account ID creation, and password strength checking.
- Apply workflow to people and account management - Workflows are a technical representation of specific business processes in ITIM and can be used to complement account provisioning and life cycle management activities, such as adding, removing, and modifying people and accounts in ITIM and managed resources across the environment.

The ITIM logical architecture

ITIM is a Java application with a logical architecture as shown Figure 8-21. It is composed of three main functional layers:

- The Web User Interface Layer
  The Web User interface is the interconnecting layer between that of the user's browser and the identity management application layer.
- The Application Layer
  This is the core of ITIM and runs on application server. The Application subsystem contains all modules that provide provisioning specific capabilities, such as identity management, account management, and policy management.
- The Service Layer
  This Core Services subsystem contains all modules that provide general services that can be used within the context of provisioning, such as authentication, authorization, workflow, and policy enforcement. This also includes communication with the adapters residing on the managed services and to directories for storage of information.

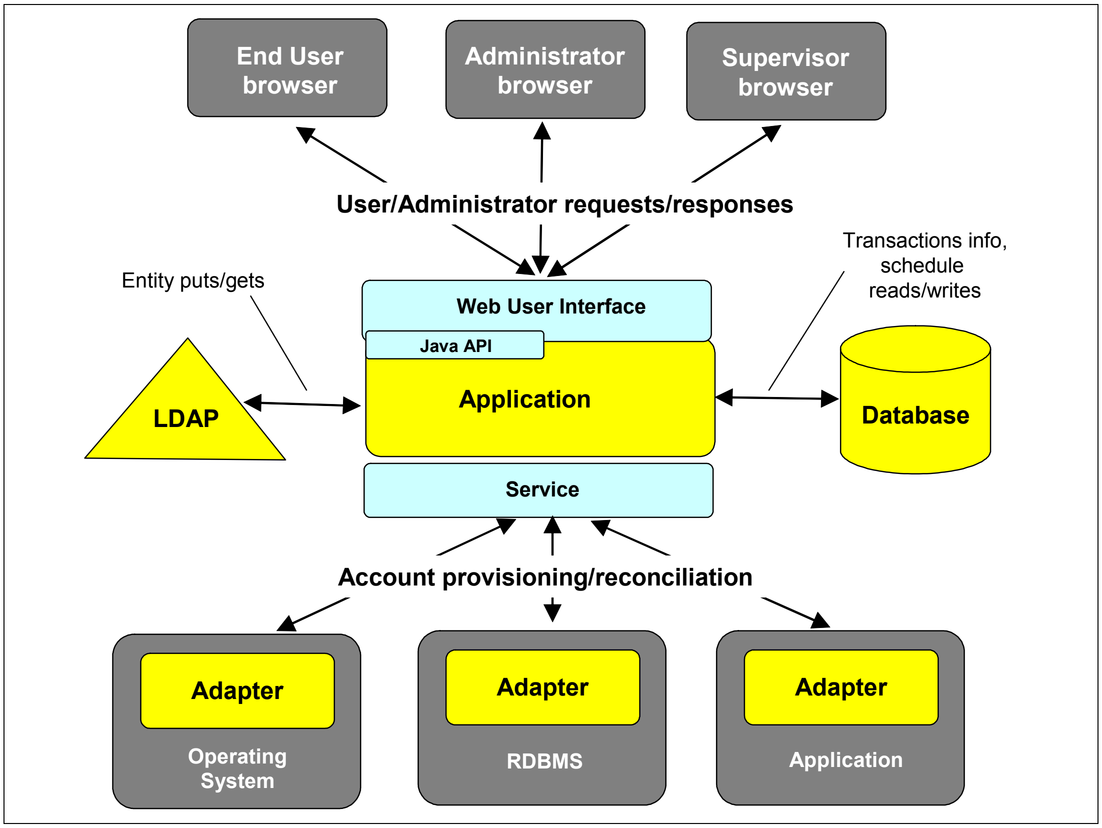


Figure 8-21 ITIM logical architecture

The IBM Tivoli Identity Manager system uses an LDAPv3 directory server as its primary repository for storing the current state of the enterprise it is managing. This state information includes the identities, accounts, roles, organization chart, policies, and workflow designs.

A relational database is used to store all transactional, reporting and schedule information. Typically, this information is temporary for the currently executing transactions, but there is also historical information that is stored indefinitely to provide an audit trail of all transactions that the system has executed.

ITIM can be hosted by the following platforms:

- HP-UX
- IBM AIX
- Red Hat Enterprise Linux
- Sun Solaris
- SUSE Linux Enterprise Server
- Windows 2003 Server
- z/OS

ITIM RACF adapter

ITIM is connected by TCP/IP to the z/OS instances it provides identity management services to. The ITIM commands sent to the adapter and responses to these commands are exchanged using DAML (Directory Access Markup Language) messages between ITIM and the RACF adapter installed in z/OS, as shown in Figure 8-22.

The RACF adapter is itself composed of two parts to be installed in the z/OS system:

- A z/OS UNIX based TCP/IP agent to directly interface with ITIM.
  The ITIM agent is a z/OS started task listening for requests coming from ITIM. When such a request arrives the agent uses APPC to trigger a command processor program in z/OS. Note that the TCP/IP communication between ITIM and the RACF agent is protected with SSL/TLS.
- The RACF command processor
  It is actually a set of APPC programs that issue RACF commands, or start RACF utilities, and send responses back to the ITIM agent.

Figure 8-22 is also showing examples of RACF userID being used for the execution of the agent or the command processor programs:

- ITIAGNT is the RACF userID assigned to the agent started task. It has been defined with the SPECIAL attribute if ITIM is to manage all the users in the RACF database, or with a GROUP SPECIAL attribute should ITIM be in charge of only certain users in the RACF database.
- ADMINX is an existing RACF userID that can be optionally specified at ITIM. In this case the RACF commands are run under these userID.
  The purpose of this facility is to support managing sub-populations of ITIM administrated RACF users, using specific administrators' userIDs.
  If this facility is used then ITIAGNT doesn't need any specific attribute in itself, however it should be defined as a surrogate of ADMINX.
  If this facility is not used, then the RACF command is executed under the ITIAGNT userID.

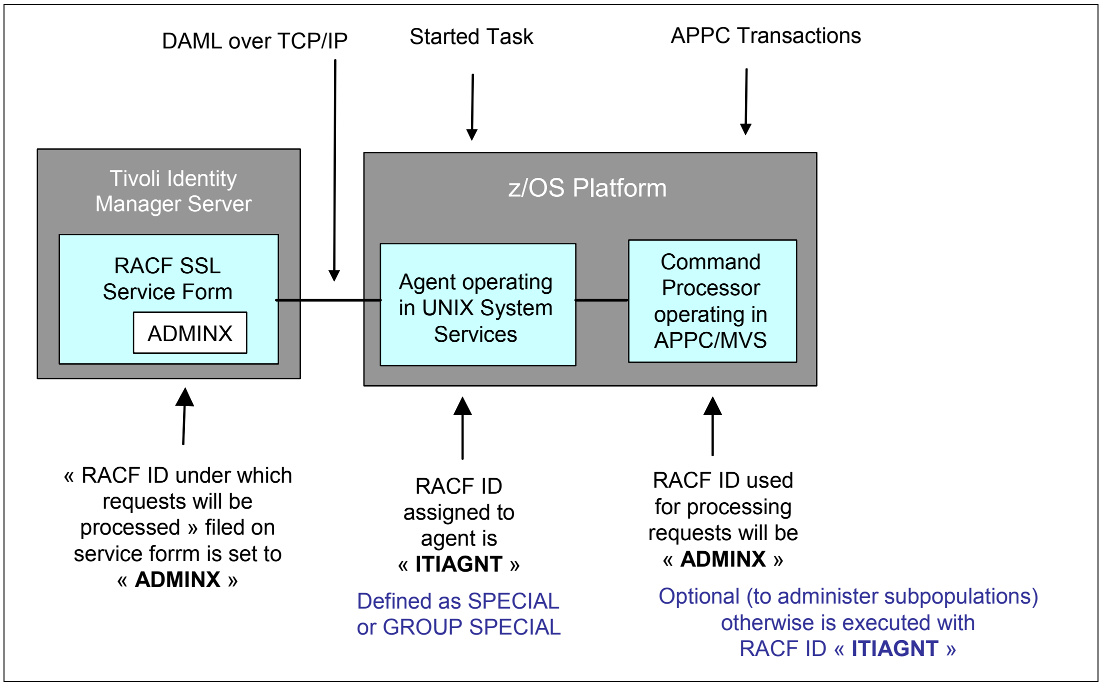


Figure 8-22 ITIM RACF Adapter

ITIM LDAP adapter

There is an ITDI-based LDAP adapter that can be installed on different platforms, including z/OS.

Complementing ITIM with ITDI

We have seen that ITDI could also be used alone to synchronize passwords. However it is strongly recommended that, whenever password synchronization is part of an identity management strategy, ITIM be used to distribute synchronized passwords. ITIM has the complete view of the accounts to be managed that ITDI does not have.

As already mentioned, the ITIM RACF adapter does not allow to reflect local changes to a RACF password back to ITIM. Assuming that an installation requires that all passwords in the installation be re-synchronized to the new RACF password, the infrastructure shown in Figure 8-23 provides this function.

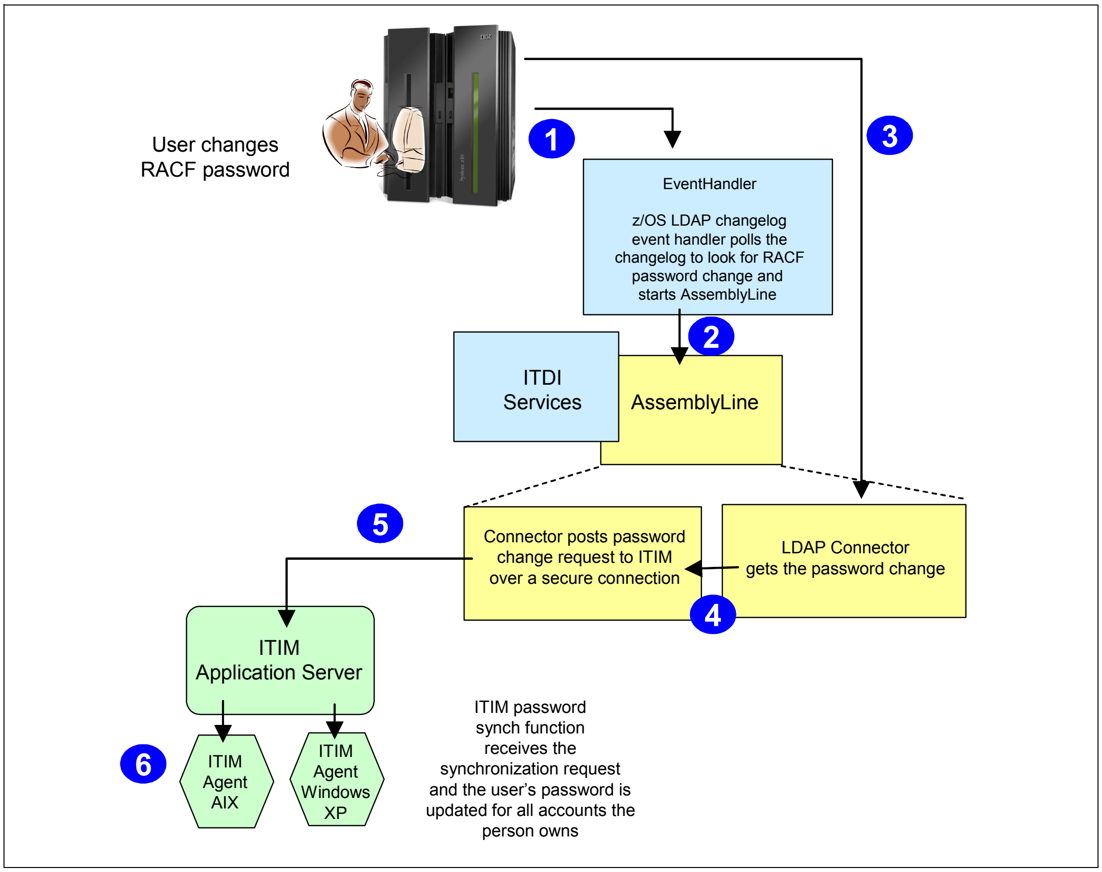


Figure 8-23 Organization's accounts re-synchronization on a new RACF password

The event flow described in Figure 8-23 is:

1. The user changes his password in RACF. This password change gets recorded in the z/OS LDAP GDBM changelog and a password envelope is created. The Directory Integrator EventHandler for z/OS LDAP detects the password change.
2. The EventHandler calls a Directory Integrator AssemblyLine that is composed of two connectors.
   - The first connector securely retrieves the encrypted password attribute and decrypts it with the supplied API in Directory Integrator.
   - The next connector builds the XML structured request to ITIM to request the password change. The connector does an HTTP post to the ITIM password synchronization servlet to make the password change request.
3. ITIM receives the password synchronization request for its use and checks to see what other accounts are eligible for password synchronization. Password change requests are

then automatically initiated for the eligible systems, for example, AIX, Windows NT®, Active Directory, and so on.

8.4.5 IBM Tivoli Key Lifecycle Manager

The IBM approach to key management revolves around IBM Tivoli Key Lifecycle Manager (TKLM), a product announced in 2008 that will be enhanced in phases. From an initial focus on key management for tape and disk encryption, IBM plans to expand TKLM into a centralized key management facility for managing encryption across a range of deployments.

The large number of symmetric keys, asymmetric keys, and certificates that can exist in the enterprise can be managed by TKLM. As of the writing of this book The TKLM product is an application that performs key management tasks for IBM encryption-enabled hardware such as the IBM System Storage DS8000 Series family and IBM encryption-enabled tape drives (TS1130 and TS1040). TKLM is providing, protecting, storing, and maintaining encryption keys that are used to encrypt information being written to, and decrypt information being read from, an encryption-enabled disk. Symmetric keys are used for LTO 4 encryption drives, and asymmetric keys are used for DS8000 and TS1100 tape drives.

TKLM operates on a variety of operating systems. Currently, the supported operating systems are:

- AIX 5.3 and AIX 6.1 - 64 bit
- Red Hat AS 4.0 x86 - 32 bit
- SUSE Linux Enterprise Server (SLES) 9.0 and 10 x86 - 32 bit
- Solaris 10 Sparc - 64 bit.
- Windows Server 2003 - 32 bit.
- IBM z/OS V1.9 and following releases (TKLM hosted in the System Service Runtime Environment for z/OS)

TKLM is designed to be a shared resource deployed in several locations within an enterprise It is capable of serving numerous IBM encrypting enabled hardware regardless of where those devices reside.

Tivoli Key Lifecycle Manager components and resources.

Figure 8-24 is a high level view of the TKLM architecture. IBM Tivoli Key Lifecycle Manager is divided into two major components, *Key Serving* and *Key and Certificate Management*.

The *GUI for key management* is the primary user interface into IBM Tivoli Key Lifecycle Manager and provides all the functionality for day to day key management. The *Audit* component creates and sends audit information into an external *Audit Log* and the *Backup/Restore* facility provides for backing up of the IBM Tivoli Key Lifecycle Manager keystore and associated metadata to a single protected file. The *Key/Certificate metadata and data repository function* maintains the metadata associated with the keys (such as key identifier, the tape serial number, and so on) and stores it in a DB2 database. Keys are generated and key grouping maintained by the *Key generation, Key group managemen*t component, which also manages key group rotation. *Certificate management and monitoring* handles certificate management, for example, the creation of certificates or the obtaining of them from third parties, checking the validity of certificates (that they have a valid root certificate and have not expired) and manages their renewal.

IBM Tivoli Key Lifecycle Manager can be implemented to only serve keys to pre-defined storage devices and the *Authorization* function validates these devices as necessary. It is IBM's stated intention to provide an API (the *Providers Service Provider Interfaces -SPIs*) to enable communication with other key managers.

The Key Serving component is responsible for obtaining the correct key from the keystore using the *Key Store API* and delivering it securely to the encrypting storage device via the *Providers SPIS* . In addition, by implementing the *Reference Implementation,* third parties are able to have their devices participate in IBM Tivoli Key Lifecycle Manager key management.

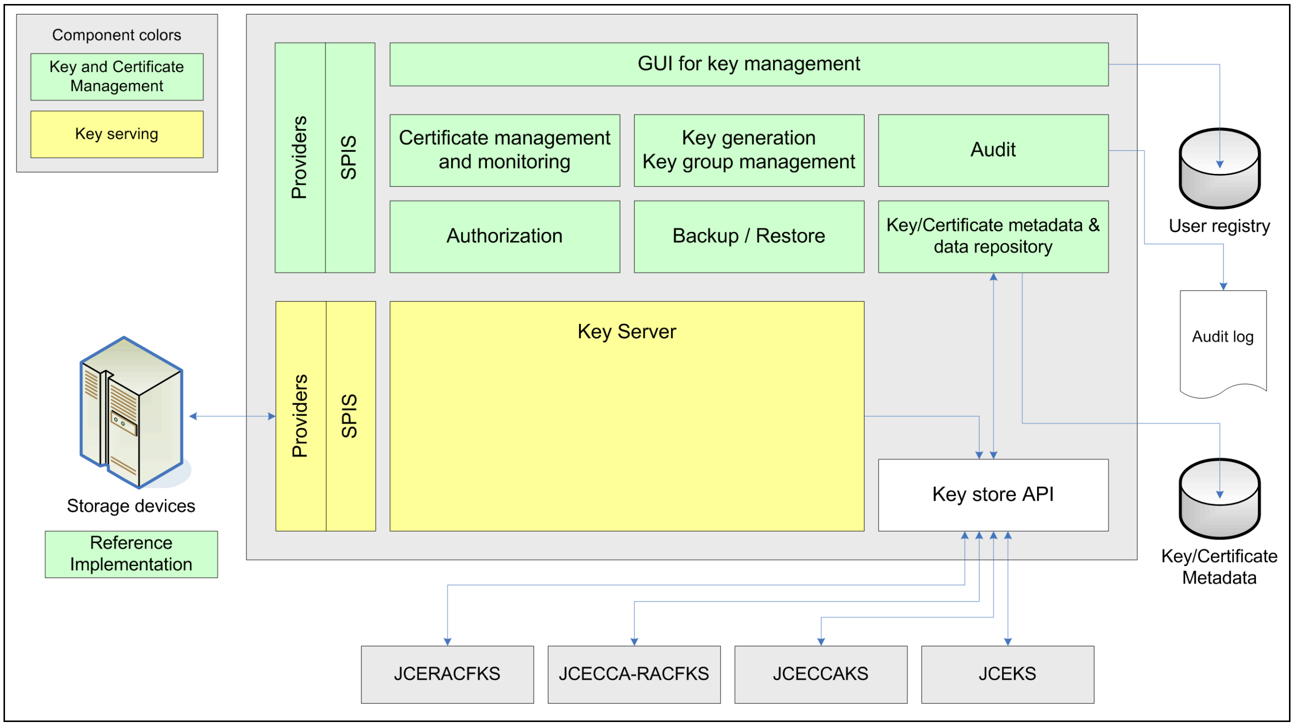


Figure 8-24 TKLM architecture.

TKLM and DS8000

With the DS8000, Tivoli Key Lifecycle Manager is used to handle serving keys to the encrypting disk drives. The TKLM does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. In addition to the key serving function the TKLM also brings the following additional functions, which can also be used for IBM encryption enabled tape drives:

- Lifecycle functions
  - Notification of certificate expiration through the Tivoli Integrated Portal (TIP)
  - Automated rotation of certificates
  - Automated rotation of groups of keys
- Usability features
  - Provides a graphical user interface (GUI)
  - Initial configuration wizards
  - Migration wizards
- Integrated backup and restore of TKLM files
  - One button to create and restore a single backup packaged as a .jar file

> **Note:** For the DS8000, an isolated primary TKLM key server is required and must be deployed on an IBM System x® running SLES 9.0 with storage not provisioned on the DS8000. Additionally, secondary key servers can be deployed on any of the previously mentioned platforms.

To perform these tasks, TKLM relies on external components. The Tivoli Key Lifecycle Manager solution includes the Tivoli Key Lifecycle Manager server, an IBM embedded WebSphere Application Server (eWAS), and a database server (IBM DB2).

The solution also incorporates the Tivoli Integrated Portal (TIP) installation manager which provides simple to use installation for Windows, Linux, AIX, and Solaris.

In TKLM, the Drive Table, LTO Key Group, and metadata are all kept in DB2 tables. The TKLM DB2 tables enable the user to search and query that information much easier. Note that the keystore, configuration file, audit log, and debug log are still flat files.

TKLM also relies on the following resources:

- Configuration file

  The TKLM has a an editable configuration file with additional configuration parameters that is not offered in the GUI. The file can be text-edited, however the preferred method is modifying the file through the TKLM command line interface (CLI).

- Java security keystore

  The keystore is defined as part of the Java Cryptography Extension (JCE) and an element of the Java Security components, which are, in turn, part of the Java Runtime Environment. A keystore holds the certificates and keys (or pointers to the certificate and keys) used by TKLM to perform cryptographic operations. A keystore can be either hardware-based or software-based. TKLM supports several types of Java keystores, including the keystores specific to z/OS Java that we describe in 8.2.4, “Java Keystores that are supported in z/OS” on page 220.

- Cryptographic Services

  TKLM uses the IBM Java Security components for its cryptographic capabilities. TKLM does not provide cryptographic capabilities and therefore does not require, nor is allowed to obtain, FIPS 140-2 certification. However, TKLM takes advantage of the cryptographic capabilities of the IBM Java Virtual Machine in the IBM Java Cryptographic Extension component and allows the selection and use of the IBMJCEFIPS cryptographic provider, which has a FIPS 140-2 level 1 certification. By setting the FIPS configuration parameter to ON in the Configuration Properties file either through text editing or by using the TKLM CLI, the user can make TKLM use the IBMJCEFIPS provider for all cryptographic functions.

9

Security exploiters

In this chapter we describe the security characteristics and capabilities of four major z/OS subsystems, provides, describes, discusses, or contains the following:

- DB2
- CICS Transaction Server
- IMS
- Websphere MQ

These four products are heavily used by many IBM customers. All four products run on z/OS and make use of the security and integrity features of that operating system, adding transaction processing, message processing and both hierarchical and relational database capabilities to the platform.

9.1 DB2

This section outlines the security capabilities of DB2 which for over 25 years has been IBM's flagship relational database.

9.1.1 What is DB2?

DB2 is a relational database system with a proven record in availability, scalability and reliability. It is now optimized for SOA, CRM, and data warehousing.

IBM DB2 for z/OS is the leading enterprise data server, designed and tightly integrated with the IBM System z mainframe to leverage the strengths of IBM System z, and to reduce total cost of ownership through process enhancements and productivity improvements for database administrators and application developers.

New structures, such as the ability to make changes to data definitions without disrupting online performance, continue to enhance availability and scalability in DB2, and bottlenecks have been removed to ensure the position of DB2 as a performance leader. DB2 for z/OS provides the core infrastructure for the next wave of Web business applications and SOA initiatives.

9.1.2 What security capabilities does it have?

DB2 runs as an APF authorized product and establishes itself as a formal subsystem. Thus it is able to use those SAF (and RACF) interfaces which enable it to perform user authentication and trusted access control.

DB2 for z/OS security capabilities can be divided into four broad areas:

- authentication
- authorization
- encryption
- auditing

Because for many years z/OS has been designed to run multiple applications simultaneously on the same server, these capabilities are mature, proven technologies.

Authentication

Authentication is the first security capability encountered when a user attempts to use the DB2 for z/OS product. The user must be identified and authenticated before being allowed to use any of the DB2 for z/OS services. DB2 for z/OS uses the z/OS Security Server (RACF or equivalent) for authentication and authorization to access any DB2 subsystem.

Authorization

Authorization is the next security control encountered. When an application gains access to a subsystem, the user has been authenticated and access to DB2 for z/OS is checked using RACF. DB2 for z/OS then controls access to data through the use of identifiers associated with the authenticated user. A set of one or more DB2 for z/OS identifiers, called authorization IDs, represent the user on every process that connects to or signs on to DB2 for z/OS. These IDs make up the SQL ID. The SQL ID and role, if running in a trusted context, are used for authorization checking within DB2.

Access to DB2 for z/OS requires the use of packages. Packages are required to execute SQL statements. They also have an owner ID or role associated with it. The owner may be different from the SQL ID or role executing the package. To execute any SQL statements bound in a package, the SQL ID or role associated with the package must have the execute privilege on the package. The package owner is used for privilege checking for any static SQL statements in the package. When executing a dynamic SQL statement, the SQL ID or role must be authorized to perform the action against DB2 not the owner. This allows DB2 for z/OS to perform as much authorization checking when the package is created and not every time it is used. Also this approach eliminates the need to authorize all users to all objects used in a package.

Encryption

Encryption can be employed to keep information confidential when it is transmitted between a DB2 for z/OS subsystem and a DB2 for z/OS client or when it is stored on disk.

For data transmission confidentiality, DB2 for z/OS provides two options: native data stream encryption supported in the database protocols and Secure Sockets Layer (SSL) supported in the network layer.

The native data stream encryption uses DES to provide a level of performance over SSL. For SSL, DB2 for z/OS exploits z/OS Communications Server's Application Transparent Transport Layer Security (AT-TLS). This facilitates the use of SSL encryption of data during data transmission between DB2 for z/OS systems on behalf of DB2 for z/OS.

For data-at-rest encryption, there are also two options: the native encryption and decryption column functions provided by the DB2 for z/OS and the IBM Data Encryption for IMS and DB2 Databases tool used to encrypt rows.

Audit

The audit facility integrated into z/OS can be turned on to track user actions in DB2. Auditors can collect log and trace data in an audit repository, and then view, analyze, and generate comprehensive reports on the data using the IBM DB2 Audit Management Expert for z/OS. You can selectively filter SELECT, INSERT, UPDATE, and DELETE activity by user or by object, and export these filters for use on another DB2 subsystem.

MLS and row-level security

You can take advantage of mandatory access control in DB2 to protect table data based on the security labels of the rows. When a user accesses a row or a field in the row with an SQL statement, DB2 for z/OS calls RACF to verify that the user is allowed to perform the type of access that is required for the SQL statement. The access is allowed only if the user has the requested access right to all of the rows containing fields that are accessed as part of the SQL statement. For all fields that the SQL statement accesses, DB2 for z/OS checks the security label of the row containing the field and denies access when the user's security label does not dominate the security label of the any one of the rows containing the fields.

Trusted Contexts

A powerful security enhancement in DB2 9 for z/OS is the introduction of trusted contexts, a feature that supplies the ability to establish a connection as trusted when connecting from a certain location or job. Having established a trusted connection, it provides the possibility of switching to other user IDs, thus giving the opportunity of taking on the identity of the user associated with the SQL statement. In addition, it is possible to assign a role to a user of a trusted context. The role can be granted privileges and can therefore represent a role within the organization in the sense that it can hold the sum of privileges needed to perform a certain job, application, or role. These two constructs together supply security enhancements for a variety of different scenarios ranging from any three-tier layered application such as SAP to the daily duties of a DBA maintaining the DB2 for z/OS subsystem.

9.2 CICS TS

This section outlines the security capabilities of Customer Information Control System (CICS) Transaction Server, which for over 40 years has been one of IBM's transaction processors for System z and its antecedents.

9.2.1 What is CICS?

IBM supplies the total infrastructure that companies need to model, develop, deploy, and manage their critical business applications. As part of this infrastructure, CICS TS v4 exploits the qualities of service associated with the z/OS platform.

CICS is a transactional application server that enables execution of a complex and demanding mixed-language application workload while efficiently integrating with SOA enterprise solutions.

CICS provides a run-time for application in COBOL, PL/I, Assembler, C, C++, and Java, thus delivering a language-neutral approach. This ensures existing skills can be used to build and deploy new applications.

CICS provides open standards-based connectivity consistent with the preferred Web services implementation while preserving the expected Qualities of Service of the trusted applications at an unparalleled cost per transaction.

CICS provides a complete systems management solution to enable management of resources through a single point of control that is consistent throughout the enterprise.

9.2.2 What security capabilities does it have?

Architecture

CICS TS makes use of a couple of the more unusual parts of the System z hardware security architecture.

Much of CICS TS runs as a user program in Key 8. (That is it does not run in supervisor state and does not run in system key; in normal operation it does not run with APF authorization either). It loads and runs user transactions which require a level of isolation both from it (the CICS code) and from one another. In order to achieve the first requirement it runs the transactions in Key 9, which allows CICS to modify the transactions storage but not vice versa.

In order to provide for the separation of transaction storage from another transaction storage it makes use of a hardware capability called Branch in Subspace Group (BSG). This allows each transaction to run in a sub-address space area of 1 megabyte.

Further levels of separation can be achieved using separation of Application Owning Regions (AORs) for separate security zones. This make use of CICS Multi Region Option MRO.

CICS makes use of its own SVC for performing many of its security functions.

CICS and SAF

In order to access CICS resources, users will sign on to CICS providing a userid and authentication credentials (such as a password). At this point CICS invokes SAF (and RACF) to create protected control blocks to represent the user. These control blocks are passed to each point where access control needs to be performed. The primary point is the access to the transaction to be executed.

Transactions are defined as 4-byte names and access can be grouped by using the RACF profile grouping facilities.

Further checks can be made by making use of other RACF resource classes. Checks are made against the transaction requestor against various resource types such as

- LUs
- CICS commands
- DB2 entry
- CICS transient queues
- Enterprise Java Beans
- File control to VSAM and BSAM files
- Unix files
- CICS logs
- various EXEC CICS commands
- CICS application programs
- DL/1 PSBs
- CICS Document templates
- CICS transactions
- CICS temporary storage destinations
- Surrogate user authority.

Checks can also be made explicitly within transactions using the CICS API for security calls (QUERY SECURITY).

HTTP clients

An HTTP client can provide HTTP basic authentication information (via a userid and password). The transaction that provides the requested services will be associated with that userid.

SSL

A client program that communicates using Secure Sockets Layer can supply a certificate which can then be mapped to a userid (see 7.4.8, "Digital Certificate Management" on page 163).

Identity Propagation

If users outside of CICS perform authentications via some other method and then pass requests into CICS to be executed by a generic userid, it is possible for these external identities to be associated with the transaction and for auditing to be performed which shows the original service requestor.

WEB 2.0

Web service clients can also pass authentication data in the form of a security token within a SOAP message. CICS provides support for username tokens and x.509 certificates, and con operate with a Security Token Service such as Tivoli Federated Identity Manager to provide more advanced authentication of Web Services.

9.3 IMS

This section outlines the security capabilities of Information Management System (IMS) Transaction Server, which for over 40 years has been one of IBM's transaction processors for System z and its antecedents.

9.3.1 What is IMS?

IBM Information management System is a highly robust transaction and hierarchical database management system with a long-standing reputation for superior availability, performance, capacity and integrity for critical online data and applications. IBM IMS includes two major components—the IMS Database manager (DB) and the IMS transaction manager (TM).

9.3.2 What security characteristics does IMS have?

As you might expect with any software which has extremely high availability characteristics, IMS has a wealth of security controls built into the heart of its software. In addition IMS leverages the unique characteristics of System z hardware and z/OS software, making use of separation by storage keys and separation of function by address spaces.

IMS runs its own code in Key 7 while running user transactions in Key 8. It has a Control Region and multiple dependent regions, where work is carried out. These dependent regions can be

- – Message Processing Program (MPP) regions,
- – Batch Message Processing (BMP) regions,
- – IMS Fast Path (IFP) regions,
- – Java Message Processing (JMP) regions,
- – Java Batch Processing (JBP) regions.

Each of these Message Processing Regions runs in its own address space. Multiple transactions do not co-exist in each Processing region thus eliminating any potential for cross-contamination or transaction interference.

Those regions designated as batch regions (BMP and JBP) interact with the control region to gain access to IMS resources which can be simultaneously used by online transactions.

9.3.3 What security capabilities does IMS have?

As with any large software component of this nature, IMS security is achieved through a combination of controls in IMS and use of SAF calls to invoke RACF services. IMS performs work on behalf of users, so each user requires authentication. Once authenticated, IMS can determine whether a user is allowed access to a given resource or is allowed to perform a specified function.

User authentication to IMS is via userid and authentication credentials and is performed using SAF (and RACF). Once the user identity is established the standard protected control block (the ACEE) is created. This is used to determine authorities to other resources such as the transaction name. Transaction names in IMS can be up to eight bytes in length. They can be grouped using RACF grouping classes.

If IMS is being accessed by a SNA network then each physical terminal can be identified and protected.

IMS has an Automated Operator (AO) interface, which can be used to issue IMS commands. The commands that can be issued through this interface can be protected via RACF resource class.

IMS data can be encrypted if required by making use of the package known as Data Encryption for IMS and DB2. If required, applications can perform encryption directly using the services of ICSF or by accessing the CPACF facilities using native hardware instructions.

Auditing

IMS has an extensive and sophisticated log which is used during transaction backout. Security violations are logged here, as well as RACF and its use of SMF. There are options to ensure violations are presented at IMS master terminals if required.

For more information about IMS and its security features the reader is referred to, *IMS System Administration Guide,* SC18-9718.

9.4 Websphere MQ

This section outlines the security capabilities of Websphere Message Queuing (MQ). This product was introduced by IBM in 1992 and was previously known as MQ Series. The name was changed to Websphere MQ in 2002. We shall simply refer just to MQ.

9.4.1 What is Websphere MQ?

MQ is a family of products which provide Message Queuing capabilities for System z applications and for applications on many other platforms. It is concerned with secure and guaranteed message delivery, from one application to another. Applications may reside on the same platform, or may reside on different computing environments which are physically separated by large distances. Platforms supports include,

- z/OS
- Linux on System z and other platforms
- OS/400®
- Various flavours of Unix (HP-UX, AIX, Solaris)
- Microsoft Windows

MQ provides queue managers for connection. It refers to the communications paths as Channels. The Channels can use a variety of protocols including SNA. However, in more recent times most channels use TCP/IP.

9.4.2 What security capabilities does it have?

Perhaps here we should try and consider what capabilities MQ needs.

In addition to the more normal issues of identification and authentication of users of its services, there are added questions of identifying and authenticating the servers to which it will communicate, and subsequently questions surrounding

- Integrity of the data transmitted
- Privacy of the data transmitted
- Non-repudiation of delivered messages

In this chapter we examine the use of MQ on the z/OS platform, but many of the issues here will apply to other platforms where MQ is run.

On the z/OS platform MQ runs as an authorized subsystem, and makes use of multiple address spaces to separate work streams. As it is authorized it is capable of creating and manipulating secure control blocks representing user identities.

MQ on z/OS has very high availability characteristics due to its use of shared queues and sysplex capabilities.

User Identification and Authentication and Access Control

Callers of MQ services have already used SAF (and hence RACF) to perform user identification and authentication. Hence MQ can make use of the user identity to determine authority to access queues or issues commands to MQ.

Server Identification and Authentication

In order for each server to be sure it is communicating with the correct target server it is necessary for each server to identify itself. This can be accomplished using public key infrastructure certificates.

In practice this is achieved using System SSL to perform encrypted data transfer.

Data Integrity

The integrity of data transfer by MQ is maintained by making use of hash functions within SSL.

Confidentiality

Confidentiality is achieved by the use of encryption schemes within SSL. All the messages blocks within a message can be encrypted.

Non-Repudiation

Non-repudiation with 100% certainty is difficult to achieve in any context. However it is possible to have MQ messages digitally signed. However it may be preferable to implement non-repudiation at the application level.

10

Solution Pattern example

This chapter contains an example of implementing a security solution on System z based on a solution pattern from the IBM Security Blueprint.

10.1 The overall views

Figure 10-1 is a graphical overview of the Security services and APIs available in z/OS. There are roughly three categories of Security services that implement:

- Platform Level Security
  RACF typically provides a set of services used to achieve Platform Level Security.
- Transaction Level Security
  Typically Transaction Level Security is provided via the support of secure protocols such as SSL/TLS or Kerberos, and the ancillary functions that are necessary for the setup of the protocol execution environment.
- Network Level Security
  These are the Security services available in the z/OS Communications Server.

There are additional APIs and services which are not today embedded in z/OS. These are typically the services and Security APIs pertaining to new and specific security models, such as the Java, J2EE or Web Services Security models. These services and APIs are provided by middlewares such as the IBM SDK for z/OS and WebSphere Application Server that run on z/OS and actually establish the applications security environment in their run-time. There is also a z/OS support for the OpenSSH protocol support which is provided as a non-priced program product to be installed in z/OS ("IBM Ported Tools for z/OS", Program Number 5655-M23).

There are three services that the z/OS platform provides for users who happen to be z/OS or non-z/OS users:

- The z/OS PKI Services
  The z/OS PKI (Public Key Infrastructure) Services is the z/OS implementation of a PKIX Certificate Authority.
- The z/OS LDAP Directory server
  z/OS does not require an LDAP server to be available for its own internal operations. The z/OS LDAP server can be exploited by users on other systems to remotely access z/OS services via the LDAP protocol or can be used as in any other LDAP implementation as network-accessible data repository.
- The z/OS Kerberos Key Distribution Center
  z/OS can act as a kerberos KDC, that is it can provide Kerberos tickets to clients. These tickets can be used for authentication to Kerberos-enabled applications that run on z/OS or other platforms. Optionally session keys in the tickets can also be used by the applications to perform data integrity or encryption.

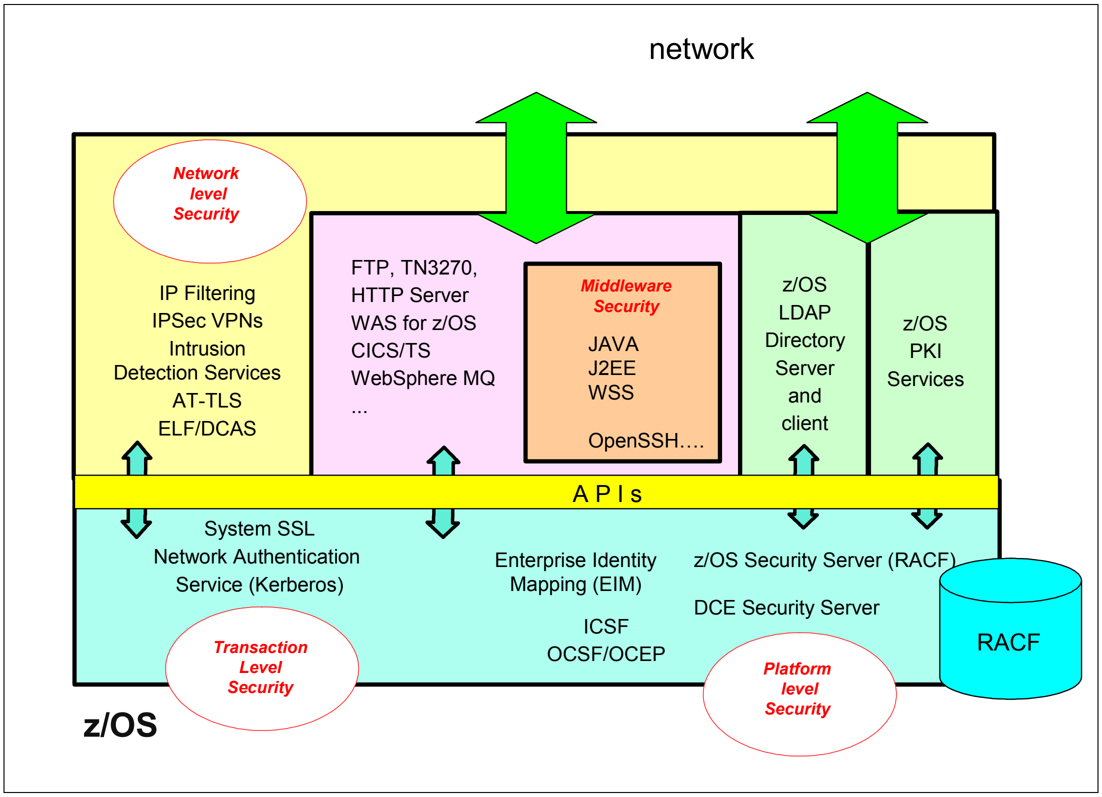


Figure 10-1 z/OS Security Services And APIs

Another view of the z/OS Security services and APIs can consist of looking at the protocols and data formats, that is the standards, used to exchange Security related information and to establish secure communications with other systems. These standards are summarized in Figure 10-2. Note that all these standards are widely adopted by the Industry, thus achieving a seamless insertion of z/OS within a network of distributed systems a quite realistic expectation.

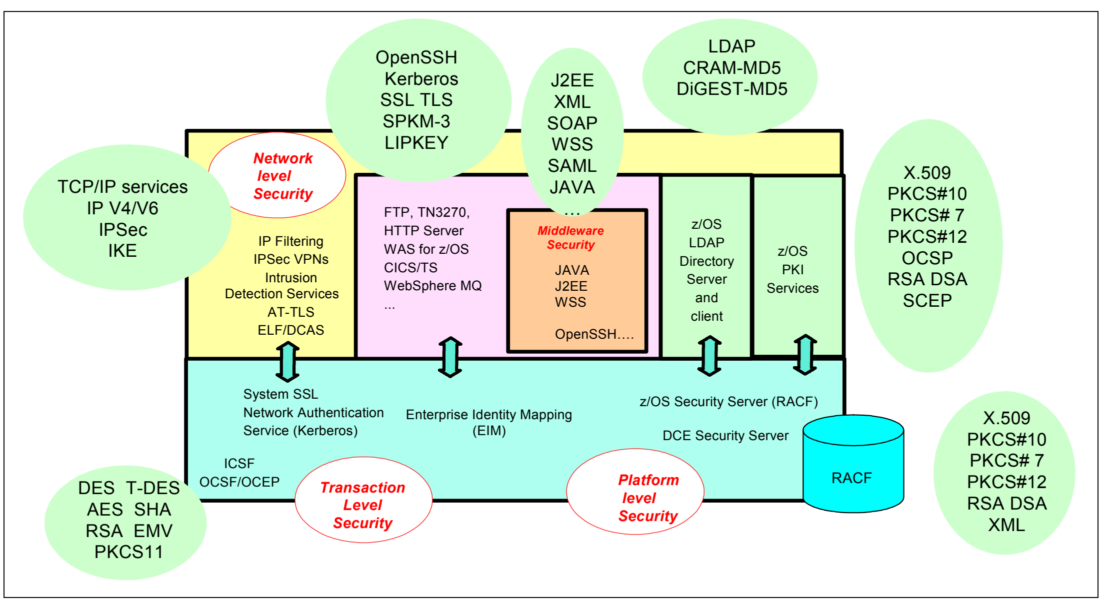


Figure 10-2 Security-related standards supported by z/OS

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get Redbooks" on page 263. Note that some of the documents referenced here may be available in softcopy only.

- *Introduction to the New Mainframe: Security,* SG24-6776
- *Security on IBM z/VSE*, SG24-7691
- Introducing the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security, REDP-4528

Other publications

These publications are also relevant as further information sources:

- *z/TPF - The evolution of transaction processing to open standards.,* G299-0768
- *z/TPF Security,* GTPS-7MS5

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

`ibm.com/redbooks`

Help from IBM

IBM Support and downloads

`ibm.com/support`

IBM Global Services

`ibm.com/services`

Index

X

Z

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

IBM Redbooks **Security on the Mainframe**

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

IBM Redbooks **Security on the Mainframe**

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

IBM Redbooks **Security on the Mainframe**

(0.5" spine)
0.475"<->0.873"
250 <-> 459 pages

IBM Redbooks **Security on the Mainframe**

(0.2"spine)
0.17"<->0.473"
90<->249 pages

(0.1"spine)
0.1"<->0.169"
53<->89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

IBM Redbooks

Security on the Mainframe

(2.5" spine)
2.5"<->nnn.n"
1315<-> nnnn pages

IBM Redbooks

Security on the Mainframe

(2.0" spine)
2.0" <-> 2.498"
1052 <-> 1314 pages

Draft Document for Review March 29, 2010 3:36 pm

Security on the Mainframe

IBM Mainframe Security Concepts

IBM Security Blueprint and Framework

Operating System and Application Security

This IBM® Redpaper Redbooks® publication documents the strength and value of IBM's security strategy with System z hardware and software. In an age of increasing security consciousness, IBM's System z provides the capabilities to address the needs of today's business security challenges. This publication will explore how System z hardware is designed to provide integrity, process isolation and cryptographic capability to help address security requirements. We highlight the features of z/OS and other operating systems which offers a variety of customizable security elements within the framework of the Security Server and Communication Server components. We discuss z/OS and other operating system and additional software which leverages the building blocks of System z hardware to provide solutions to business security needs. This publication's intended audience are technical architects, planners and managers who are interested in exploring how the security design and features of System z, the z/OS operating system and associated software addresses current issues such as data encryption, authentication, authorization, network security, auditing, ease of security administration and monitoring.

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

SG24-7803-00 ISBN