

z/Auditing Essentials

VOLUME 1

zEnterprise Hardware
An Introduction for Auditors

Julie-Ann Williams
Martin Underwood
Craig Warren
Steve Tresadern



www.newera.com

Foreword

by Barry Schrager

This has been an interesting book for me to review. I have been involved with data security since 1972 when I formed the SHARE Security Project and we developed the security requirements for future IBM Operating Systems. And, when I was not satisfied with the IBM response to those requirements (RACF), I developed ACF2 to prove that the requirements could be accomplished. RACF and Top Secret now also meet those requirements.

But, my whole focus was on data security in an operating environment and I never thought much about the steps before the system was IPL'd and the External Security Manager (ACF2, RACF or Top Secret) was active. This book opened my eyes to the issues involved before the IPL is complete including the LOADxx set of parameters that define where the Operating System Parameters are stored and the layout of the storage devices, how they are mapped – and even added dynamically by activating a new Input/Output Definition File (IODF)! When presenting on security vulnerabilities, I often give examples of system integrity vulnerabilities caused by sharing of storage devices containing production data or libraries between production and development LPARs or systems, but this book brings the whole concept forward, clarifies it and explains how this can be correctly managed.

Although some Auditors may believe this book gives them more information than they needed to know, it certainly should open their eyes to issues that should be looked at for a complete audit. The book provides the big picture of how all the pieces fit together and gives them the basis for asking the right questions – many of which I am sure they did not know.

I really have to give Julie-Ann, Martin, Craig and Steve credit for the details and the explanations that they provide. I am sure it was not easy. Even experienced Systems Programmers would have some difficulty embracing these concepts and Auditors who did not come out of that environment would have a steep learning curve. I cannot wait for Volume 2 which covers the auditing concepts for z/OS in the operating environment.

Barry Schrager – May 2011



Table of Contents

1.1	About this Book	1
1.1.1	About the Book's Sponsor	1
1.2	About the Author(s)	2
1.2.1	Introduction from the Lead Author	2
1.3	About You	5
1.4	Icons Used in this Book	5
1.5	More Detailed Technical Information	5
2	What Auditors Want	7
2.1	Auditing the z/Platform	7
2.1.1	Environment	8
2.1.2	Language	10
2.1.3	Required Skill vs Available Skill	11
2.1.4	Internal vs External	11
2.1.5	Regulatory Compliance	12
2.1.6	The Statement of Work	13
2.1.6.1	Mapping Business Objects & Control Objectives	13
2.1.6.2	Deciding on the Audit Scope	14
2.1.6.3	Audit Cost vs Organizational Benefit	14
2.1.7	The need for continuous improvement	15
2.1.7.1	"No Walruses in the Gulf"!!!	16
2.1.8	Separation of Duties	17
3	Foundation	19
3.1	General Purpose Business Computer	21
3.1.1	The need for System Integrity	21
3.1.2	The Integrity Statement	22
3.1.3	Your z implementation vs IBM's delivery practices	22
3.2	Mainframes - The last 50 years	22
3.2.1	From one to many Virtual Machines	23
3.2.2	Current Trends	24
3.2.3	Current z/Hardware Platform	24
3.2.3.1	Advantages – Total Cost of Ownership	25
3.2.3.2	Environmental – Green Movement	26
3.2.3.3	Managerial – Footprint Consolidation	27
3.2.3.4	Futures	28
3.3	The Parallel Sysplex	29
3.3.1	The need for more Processing Power	29
3.3.2	Massive Parallelism	30
3.4	Operating System Selection	30
3.4.1	What is it?	31
3.4.2	Advantages and Disadvantages	31
3.4.3	z/Platform	31
3.5	The External Security Manager	32
3.5.1	What is it?	32
3.5.2	Advantages and Disadvantages	33
3.6	Application Focus	33
3.6.1	Business Objects	34
3.6.2	Business Controls	34
3.6.3	Disaster Recovery	34
3.6.4	Asset management	35

3.7	Using the IODF as the z/Audit Control Anchor Point	36
3.7.1	Why the IODF is Critical to System Integrity	37
3.7.1.1	Hardware Configuration Definition (HCD)	38
3.7.1.2	Hardware Configuration Manager (HCM)	38
3.7.2	Mapping Business Control Objects to System z	39
4	Processors and Partitions	41
4.1	Logical Channel Subsystems	42
4.1.1	Logical Partitions	43
4.1.1.1	Types and Descriptions	44
4.1.1.1.1	Coupling Facility LPARs – Risk	44
4.1.1.1.2	Operating System LPARs – Risk	46
4.1.1.1.3	LPARs	46
4.1.1.1.3.1	z/OS LPARs	47
4.1.1.1.3.2	UNIX Systems Services focused LPARs	47
4.1.1.1.3.3	Non z/OS LPARs	48
4.2	Workloads	49
4.2.1	Revenue Generating Objects	50
4.2.1.1	Business Control Objects	50
4.2.1.2	Shared Objects	50
4.2.1.3	Legacy vs Contemporary	50
4.3	An LPAR's PHYSICAL Properties	51
4.3.1	Input/Output Control Program (IOCP)	52
4.3.2	Unit Control Word (UCW)	53
4.3.3	HCD/HCM/HMC	54
4.3.4	The Support Element (SE)	55
4.3.5	Power on Reset (POR)	55
4.4	Channel Path Id (CHPID)	55
4.5	Switches - FICON/ESCON/InfiniBand	57
4.5.1	Switch Control Program (SWCP)	58
4.5.2	PORTS – Internal vs External Connections	59
4.5.3	Chaining and Cascading	60
4.6	I/O Devices	60
4.6.1	Control Unit Sets	61
4.6.2	Allow LPAR Access	61
4.6.3	An LPAR's LOGICAL Properties	62
4.6.3.1	Operating System Control Program (OSCP)	63
4.6.3.2	NIP Consoles – Security Considerations	63
4.6.3.3	Esoteric Devices – Allocation Concerns	63
4.6.3.4	Unit Control Block (UCB)	64
4.6.3.5	HCD/HCM	65
4.6.3.6	LOADxx Configuration Member	66
4.6.3.7	Initial Program Load (IPL)	66
5	IPLing a z/OS Logical Partition	69
5.1	Required System Elements	70
5.1.1	Hardware Management Console (HMC)	71
5.1.2	IPL Unit Address	72
5.1.3	LOAD Parm Decoded	72
5.1.3.1	LOADxx Member	73
5.1.3.2	Location	74
5.1.3.3	Example and Keyword Syntax	74
5.1.3.3.1	IODF Decoded – ConfigID as link to OSCP	76
5.1.3.3.2	IEASYM – Path to System Symbols	77
5.1.3.3.3	SYSCAT – Path to System Catalog	77

5.1.3.3.4	SYSPARM – Path to IEASYSxx	78
5.1.3.3.5	PARMLIB – Path to PARMLIB Concatenation	79
5.1.3.3.6	SYSPLEX – The Name of the Parent Sysplex	79
5.1.3.3.7	NUCLEUS – Path to Platform Architecture	80
5.1.3.3.8	NUCLST – Path to NUCLSTxx	80
5.1.3.3.9	Member Filtering – Flexibility, Efficiency	80
5.1.3.4	Risk – Orphan Members	81
5.1.3.5	Nucleus Initialization Process (NIP)	81
5.1.3.6	LOADxx Filtering	82
5.1.3.7	IMSI - Initial Message Suppression	83
5.1.4	SYS1.SVCLIB 79	84
5.1.5	SYS1.NUCLEUS	85
5.1.6	IPL Text	85
5.1.7	The IODF Dataset – OSCP	86
5.1.7.1	HARDWARE	87
5.1.7.2	LPARNAME	87
5.1.7.3	VMUSERID	88
5.1.8	Discovering parameters	89
5.1.8.1	Find NUCLSTxx	89
5.1.8.2	Find IEANUCxx	90
5.1.8.3	Determine the MVS Level	90
5.1.8.4	Find the Master Catalog	90
5.1.8.5	PARMLIB Concatenation	90
5.1.8.6	IEASYMxx Member(s)	91
5.1.8.7	IEASYSxx Member	91
5.1.8.7.1	Parameters	92
5.1.8.7.2	Directors	93
5.1.9	Prevailing PARMLIB Members	93
5.1.9.1	High Risk	94
5.1.9.2	Low Risk	95
5.1.10	Dynamic Configuration Changes	96
5.1.10.1	MVS Operator Commands	97
6	And Finally	101
7	What's Next?	103
8	References	107
9	Glossary of Terms	109
10	Index	113
11	Appendices	115
11.1	CA Technologies Security Management Solutions	115
11.2	IBM Health Checker for z/OS	116
11.3	Key Resources	117
11.4	NewEra Software	117
11.5	Vanguard Integrity Professionals	119

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



1.1 About this Book

This book is designed to work in the same sort of way as the famous “for Dummies” books. It’s not one of the official sequences but the style works so well for explaining complex functionality that it seems the best approach for this subject. It’s the second in the series from the same team, following on from *CICS Essentials - Auditing CICS - A Beginner’s Guide*.

First things first, you should never try to audit anything using just a “for Dummies” book! The aim here is to make the whole process slightly less intimidating and more accessible to people who have already been around the audit industry for a while.

Let’s think about security in general rather than just as it relates to computers for a moment. You can have several, multi-lever locks possibly including biometrics such as finger print scanning to make sure that no one can enter your office building via the back door. However, all of this is pretty pointless if your employees routinely leave the Front Door wide open!

This is Volume 1 of a 2 Volume set. Volume 1 deals with what we call Front Doors to System z. This means it won’t be talking technical about RACF or any of the alternatives. The focus is on issues before the external security manager is active as well as hardware configuration. Volume 2 will be dealing with the more widely acknowledged, Back Doors to System z. This will be where you see APF Lists and RACF/ACF2/Top Secret Analysis suggestions.

As usual with books from this team, there’s nothing to memorize. There will be no tests.

What you will find are de-jargon-ified explanations of concepts and specific parameters. It is a distillation of a number of people’s personal experiences in the field written in “Clear English”.

1.1.1 About the Book’s Sponsor

For most of my career I have been a Trainer. I like being able to make it easier for other people to understand a subject than I found it when I first learnt.

But my “first love” is mainframe security - tell no one how much of a geek I am – it can be our little secret! I revel in the fact that ours is the most securable, commercially available, platform in history. But I despair when I see the kinds of misunderstanding this generates in the industry.

When NewEra Software offered me the opportunity to set the record straight, I didn’t have a single second thought. I have been evangelizing about this subject where ever I can get an audience to stand still for 5 minutes for the last 30 years!

NewEra Software is one of several providers of z/OS integrity solutions and when they asked me to undertake the assignment I was clear that I would not show bias towards any product. There are a number of different choices a customer can make about vendor solutions and this book needs to stand in any z environment. However there is an appendix at the end of the book which details the solutions offered by NewEra Software along with the other leading players in the marketplace.

1.2 About the Author(s)

Julie-Ann Williams has been messing around with IBM mainframe computers for most of the last 30 years. She has been helping Customers to get ready for external audits since 1987.

As well as being something of a RACF geek, Julie-Ann has extensive experience with web enabling mainframe applications for large IBM Customers and was one of the first people in Europe to implement Domino (Lotus Notes) on a mainframe. She has an unusual blend of skills encompassing detailed, classic mainframe knowledge (don't call it legacy!) as well as "newer" technologies like WebSphere, TCP/IP and UNIX combined with communications abilities and Mentoring. She's pretty sure that the last 2 used to be called having a chat with your colleagues in the good old days.

Julie-Ann took the lead in writing this book ably assisted by a number of Industry Experts including very significant contributions from:

Martin Underwood is an enthusiastic z Evangelist with almost 25 years of experience. His most recent specialism has been in helping z Customers to prepare for all sorts of IT audits. He says he never wants to stop learning and that teaching is the best way to learn. Deep ☺

Craig Warren has been in the z Industry for a quarter of a century. He laughs at those who say the mainframe is dead and his past couple of decades working on bleeding edge z projects gives him good reason!

Steve Tresadern has been working as a kind of missing link between Audit and the technical teams of Systems Programmers and Security Analysts for a major, international financial institution for some years. He enjoys helping the different teams to communicate and long walks on the beach.

1.2.1 Introduction from the Lead Author

I always enjoy the process of writing a new book. First comes the spark of an idea. Then the flurry of research associated with making sure no one else has already done it! Working with NewEra on books with a technical basis is always a pleasure because their team of Peer Reviewers always expresses strong opinions. Once everyone has agreed that there is a need for the publication and that it will not simply be "covering old ground" the actual writing begins.

I work with a fabulous team who all have decades of experience in the industry. We do the actual writing part of the process taking input from friends and colleagues as we go. But the real fun starts once we have a first draft that we can send to the review team.

This time around we had kicked off the project by talking together about a new work to encompass all of the things that need to be considered when auditing a System z environment. But the work is separated into 2 volumes and the first 1 (this book) will deal with issues that have not necessarily **ever** been considered by Auditors before.

The same team who did the Peer Review for last year's highly successful CICS Essentials - Auditing CICS - A Beginner's Guide have contributed to this book and they know the completely obsessive level of detail that I go into! Almost without exception, they came back and told me that they had initially been disappointed by the fact that the book doesn't detail how to audit z/OS.

Rest assured, Volume 2 will contain the obsessional level of detail on auditing z/OS - the operating system - that they (and maybe you) were expecting to be

contained here.

Volume 1 looks at what must be done to ensure any audit you perform on the operating system will still be valid the day **after** completion!

Taking it to absolute basics, there are 2 entrances to most homes and businesses; the Front Door and the Back Door. What we have all been auditing and securing up until now is the Back Doors. You can have a flawless security system on your Back Door but if you don't at least close the Front Door you will still lose your belongings to an opportunist thief. The same is true of any computer system.

Every other platform's audit takes configuration management into account but on z it is somehow considered to be achieved by magic. IBM encourage this thought process by providing "wizards" to help with everything from setting up logical partitions to assigning I/O devices to specific control devices through to the software Healthchecker(s). Expelliarmus!

I am not saying that what this book talks about is currently BAU (Business As Usual aka normal practice) for auditors. What I am saying is that it **SHOULD** be!

An audit colleague of mine, who is an ex-Systems Software Designer and Developer, recently shared this story with me to help illustrate the Front Door configuration issue:

"I ran across an example of this in a recent audit for one of my customers. They had disk drives mapped to multiple LPARs which did not all have the same level of control – one was production, one was development and the last was quality control. They were using ACF2 and had different ACF2 databases for all 3 LPARs. They had decided that they would lock down the access to the production volumes from the test LPAR by using the ACF2 Volume Controls.

The ACF2 Volume Controls protect at the volume level – they use a pseudo dataset name of VOLUME.@volser instead of the actual dataset name to protect the datasets on the volume. This was created originally for Systems Programmers who were building a new system. It allows this type of activity to be secured using VOLUME.@NEWRES rules instead of giving the users update access to actual system datasets with the associated logging, etc.

But, the customer had EKC's Firecall program. The Firecall program was developed to give emergency access to Systems Programmers to avoid them having a second fire-call Logonid or having to involve the Security Administrators when there was a critical situation that needed to be addressed immediately. It was always designed to be used very infrequently and for the changes made to be in place only for a short period of time.

So the Systems Programmers just obtained the NON-CNCL privilege (the RACF equivalent would be OPERATIONS) using Firecall functionality on the test LPAR and merrily updated all the datasets on the production system they wanted. More importantly, they were doing it over 20,000 times per day!! And all the ACF2 logs showed were accesses against pseudo dataset names of VOLUME.@volser. Not good!

There might have been some rationale initially for the customer's systems programmers to map the production volumes to the test LPARs, but they abused it. It certainly would have been absolutely locked down had they done the mapping correctly. Not to mention much more secure."

Most importantly, you should be able to see that not addressing some of these **Front Door issues can lead to non-compliance with**, as in the example above,

Sarbanes Oxley!

So what I want to do with this book is to show you why a problem which has always been present - that of configuration change management being able to open up Front Doors to your z environment - “suddenly” requires greater consideration.

Chapter 1 - You are here!

Chapter 2 – **What Auditors Want** is where I start the journey by outlining the needs of the audit community in System z. I've written this chapter so that it will make sense to both auditors and technical personnel. This will give you the common language that you need to make sure that everyone is speaking the same language **and** that they all understand the same things from what is said!

Chapter 3 - **Foundation** gives me the time to me to show how System z has evolved into the complex environment we see today. But that audit was left behind on that journey is not in question. Most of us are still using check lists that were assembled 20 or more years ago. In fact, what was being done with the platform then bears almost no relationship to what we as auditors see happening today.

Chapter 4 - **Processors and Partitions** gives me the opportunity to explain the fundamental differences between logical and physical elements in a highly virtualized world. This chapter goes down into the fine detail of IODF (Input/Output Definition File) parameters that should be examined and what their purpose is. I also take this opportunity to share experiences with you to help in the decision making process - not to mention identify the Teams who can be of most help to you as a z Auditor.

Chapter 5 - **IPLing a z/OS Logical Partition** may seem like an unlikely topic for a book that is all about Front Doors or Hardware Configuration (and there will be much more detail on the IPL process in Volume 2) but it is relevant here too. At most points during the IPL process it is possible to change the shape of the operating system which gets started. The external security manager (RACF, CA ACF2 or CA Top Secret) doesn't get initiated until well into the proceedings. So it is possible for those changes to go unrecorded. This chapter outlines the points at which extra pressure needs to be exerted by the z Auditor.

And finally the Appendices give you an idea of the types of products that I and my colleagues are using out in the field to help in the fight to build secure Front Doors on System z.

There are points where I have had to resort to deeply geeky, technical language - particularly when I am talking about the parameters which can shape both physical and logical connectivity. I am sorry about that but do please try to bear with me. The relevance should become clear - without this information it would be impossible for me to tell you what you **ought** to be looking at during a System z audit **that you don't currently examine**.

Honest - I am just trying to help...

Julie-Ann Williams

Senior Technical Consultant specializing in System z Security - February 2011

1.3 About You

You are either:

- A z/OS Systems Programmer who is approaching an external audit, possibly for the first time, and wants to know what might/should be looked at.
- A mainframe Security Analyst with a keen understanding of security and a desire to help improve the overall resilience of a z Series installation.
- An IT Auditor who finds them self auditing a z Series installation, possibly for the first time, and wants to know how to ask questions that will actually get useful answers.
- An IT Auditor who has become aware that there is more to z Series security than just RACF but isn't sure where to find out what the potential problems are.
- Anyone with an interest in mapping Business Processes to physical and logical IT resources on an IBM mainframe.
- Not a potential z hacker - this book will **NOT** detail how to exploit any vulnerability discussed! It will detail how to secure against the exploit though.

1.4 Icons Used in this Book

You've seen other "for Dummies" books. You know how this works. Icons are used as short hand ways of saying the same important things.

The following icons indicate expert knowledge that you will need in order to understand how to audit System z:



Don't forget to remember these important points – or at least remember where you read about them! They will help you to understand the background of the system that you are auditing. System z, and its predecessors, has been around for almost 50 years. There are a few quirks which you should know about.



This icon alerts you to a juicy piece of information that will make auditing System z easier. It may be a technical tip or advice to talk to a specific group of people to save time when finding the right information or other gems of hand acquired wisdom.



And finally, there is much talk of Back Doors in relationship to hacking computers. In System z terms this relates to the back end security of the environment - RACF, CA ACF2 or CA Top Secret. This book will define where "Front Doors" exist to the platform. There is little point in locking the Back Door if the Front Door remains wide open! The icon will show you where a Front Door exists within the System z environment.

1.5 More Detailed Technical Information

This is at heart a technical document. I have drawn heavily on the IBM documentation and would urge anyone with a reason to read this book to try these sites too:

z/OS

www-03.ibm.com/systems/z/os/zos/bkserv

www-947.ibm.com/systems/support/z/progportal

www-03.ibm.com/systems/z/os/zos/bkserv/lookat



2 What Auditors Want

The Financial Audit drives the requirement for any Information Systems Audit. This is a fact whatever business you are in.

Whilst the IS Audit folks will have an understanding of the industry and its best practices, they won't necessarily be experts on **your** internal processes. So, any audit of a System z environment will require cooperation between the Auditors and the specialized technicians running it.

Stu Henderson is a high profile player in the z/Audit environment and I borrow from his comments from time to time. When we first started talking about this book he said to me: "Is this going to be addressing just security audits? There are other types of audit: effectiveness, cost-effectiveness, availability, reliability..." And he has a strong point!

There are some ground rules that you must establish before embarking on any audit of the z/Platform. Everything from the physical security of the machine room (including access to Hardware Management Console and system consoles), to the protection of datasets across all installed operating systems and releases should be taken into consideration.

Any auditor tasked with checking the integrity of an organization's z/Platform must be able ask the right questions to extract the required information from the technical staff responsible for running and maintaining the mainframe installation. Once this information has been gathered they must also have access to the necessary skills to be able to check the actual state of the settings against audit requirements, understanding the impact of any variances. This is a uniquely complex set of needs on the z/Platform.

Whilst the rise of personal certification such as CISA, CISM and CISSP has expanded the pool of mainframe security geeks who **can** understand the wider picture, you can't rely on the presence of such qualifications. One of the biggest problems that must be tackled is enabling the technicians to understand what is going on and why auditors ask so many different types of questions in order to gain their buy-in.

This chapter is designed to be used by Auditors to help the on-site z/Technicians understand the need for audit. Reading it should help to ease the language and cultural differences which exist between audit and technical. First though I outline a roadmap of how System z hardware connects to software and where the risks live.

2.1 Auditing the z/Platform

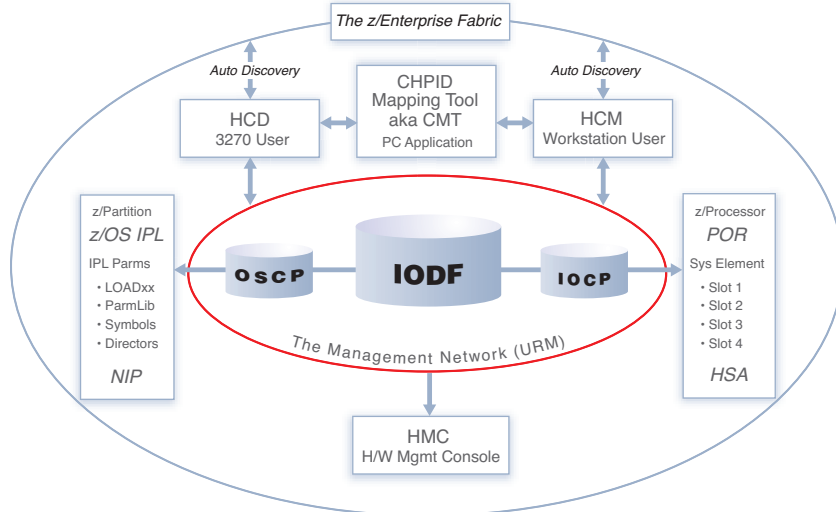
System z is without doubt one of the most complex platforms commercially available in that a single "server" is capable of running many different operating systems and an almost infinite number of application combinations. Right now audits typically only look at a single element – the z/OS operating system in a single logical partition.

There is no doubt in my mind that the situation is exacerbated by some of the new functionality in zEnterprise that has been primarily driven by customer needs – for example the "Plug&Play" like feature in HCD/HCM at z/OS v1.12. For almost the first time, new hardware is driving our need to reassess audit practices.

Before we can do that we have to understand what the impact of these new features, together with their effects on functions available with some of the more established

System z tools, can have on the systems we are trying to control. And before we can do **that** we have to get to grips with the whole platform!

IODF – The Absolute z/Enterprise Control Point



The following information will not provide sufficient detail to bypass the need to take a cooperative approach to auditing System z. What I hope it will do is give you a roadmap to understanding where you need to be looking and why.

2.1.1 Environment

So what's the big deal? OK, it's a mainframe but surely it can't be **that** different?

Virtualization is the big difference. System z mainframes are designed to be partitioned to run **many** discrete operating environments. The hardware is shared though. What this means to any audit is that if you aren't very careful about how you set up the IODF, 2 completely separate services (and potentially many more), with completely separate security environments, can get access to the same data.

The IODF is the central configuration file for the System z environment. It contains much of the same information as PC BIOS settings. Many electronic gadgets use the same idea - up to and including universal remote control devices. And it is not maintained by IT Security. In the System z world, the IODF is generally updated by hardware folks who don't have security at the uppermost of their thought processes.

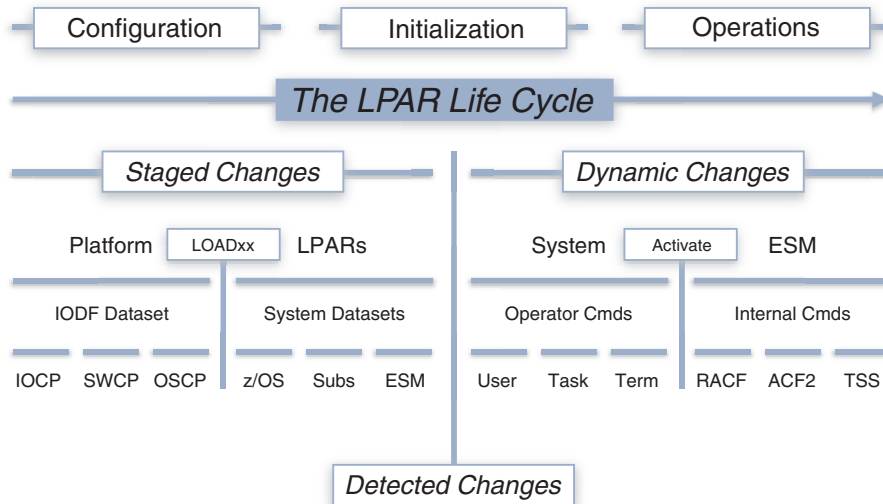
The different logical partitions within a System z environment usually have different purposes e.g. a Test version of the company Production environment. This Test LPAR (Logical PARTition) could be used to make sure that code changes don't interrupt normal business processes. In turn, this means that often access given to Developers on the Test LPAR will be much higher than that imposed in Production.

If the Test and Production LPARs both have connections to the same I/O devices then they can access each other's data. And if someone can read Production data (filled with real, sensitive data) then they can take a copy of it. If they can take a copy of Production data from the Test LPAR then the results of any audit on Production would, immediately, be invalidated!

Auditing System z only at the LPAR level may leave significant problems unexposed. It's a situation which is complicated even further by the ever expanding and contracting number of LPARs running on the specific system you are examining. Clearly the solution is to look at a different set of control points on System z rather

than just the ones we have been used to.

They say a picture is worth a thousand words and who I am to question that judgment? So this new paradigm has to look at the different stages of each LPAR's life. From inception to abandonment the LPAR goes through 3 distinct stages (each usually many, many times) which must be addressed with control points:



Configuration = change either staged or dynamic. It always takes place before an LPAR is started for the first time, but it can also happen anywhere throughout the Life Cycle of that LPAR.

Initialization = the actual IPL process. Dynamic alterations can be made at many points in the IPL and most of them will be outside the influence of defined security (RACF, ACF2 or Top Secret). Staged changes will be picked up during the IPL too.

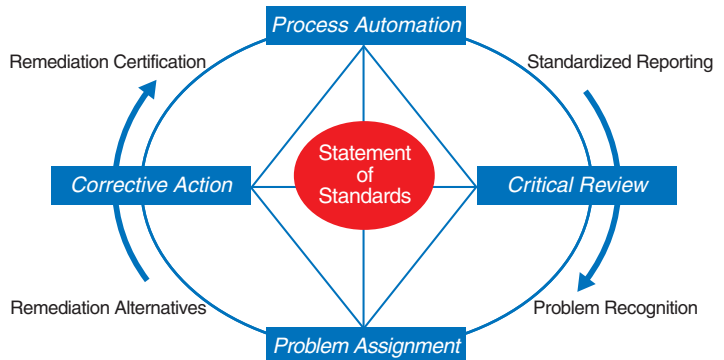
Operations = BAU. The day-to-day running of business processes on an LPAR which is active. Dynamic changes can still be made but security will be applied.

We've got to be auditing **all** of the points listed on the diagram above. That's the hardware, the software that affects the hardware, the operating system in each individual LPAR as well as the external security applied. It's also about both Staged and Dynamic changes. You can't exempt any of the parts.

What I will show you in the rest of this book is how to identify what you need to be looking at during any audit of System z.

Don't expect to be able to just pick up a check list any more. Auditing System z is about fostering a co-operative attitude amongst the various technical teams and getting good input into designing an audit program which is appropriate for the environment under examination. Only with this level of co-operation can you ensure that an audit will pick up points where improvements can be made to ensure that not only is the system/ LPAR/environment secure/compliant NOW, but that it will continue to be after the next IPL.

System Compliance Model – System Control Points – LPAR Monitoring



2.1.2 Language

As in most walks of life, the language we use is critical. All sides of a conversation about audit must understand the same thing from the same words. Systems Programmer language is all about the technology while for Auditors it's more about concepts or practical application of knowledge.

There are 3 primary control terms associated with audit:

1. Risk
2. Mitigation
3. Control

Risk assessment is all about understanding the actual value of the business data. It requires full management buy in and the amount of work involved in identifying risk on the system is typically massive!

Risks fall roughly into 3 categories: **C**onfidentiality, **I**ntegrity and **A**vailability (CIA).

Confidentiality Risks involve the ability of people, outside of those with a genuine requirement, to see the business data. Think about the secret formula for Coke and how much damage could be caused by copies being leaked to the internet.

Integrity Risks are where existing security controls may not be sufficient to guarantee the completeness or correctness of the business data. Think about a hacker covering their tracks by removing information from audit logs.

Availability Risks live in the Systems Programmer's space. These are the dangers which can be caused by changes to the underlying system which can literally cause availability problems. These used to be the "least considered" risk in IT. Now they are only the "least considered" risk on the mainframe! A denial of service can cause as many problems to an organization as someone swiping a copy of your customer database and selling it to the highest bidder.

For example - an availability and/or confidentiality risk: To make an alteration to how a system is loaded at IPL (and its hardware configuration) requires **only** that a person has access to the Hardware Management Console. Until recently IBM were STILL suggesting that the engineer's password not be changed from the default! This user has access to most functionality on the HMC. Any changes made here are before any operating system, or its security, has become active on your z/Platform. In other words if you get access to the HMC you can change the configuration without being challenged or monitored!

Mitigation is what can be done to minimize the identified risk. To follow on from the above example: Part of the mitigation might be to have a CCTV camera pointed at the HMC. Or an alternative would be to use procedures to ensure that the person making the changes to IODF is not the person who uses the HMC - an example of Separation of Duties (aka Segregation of Duties) offering a way to avoid conflicts of interest.

Control is what is applied to the system to keep track of what's going on. Again, there is a breakdown of control elements to 4 specific areas of control:

1. Identity Management
2. Access Control
3. Detective Control
4. Preventative Control

Detective controls are literally those that detect changes to the system in real-time.

The last one, Preventative Control is what Auditors tend to see as top priority. If the damage can't be done in the first place then it never needs to be fixed in a hurry!

One more point in this introduction to Audit for Systems Programmers, etc. that also applies to experienced auditors...

It's all about scope. The scope of controls to be applied to any system **must** be balanced with the actual value of the business data. There is no point in applying military grade security controls to the data on a system if it is all publically available data. That is the point where security becomes a hindrance.

The scope information needed to apply balanced controls can only really come out of the risk assessment phase of a wider project.



2.1.3 Required Skill vs Available Skill

It's no secret the average age of a z/Professional, never mind a z focused auditor, is steadily increasing. As time goes on the number of specialists not yet retired is steadily falling. You could say it's our own fault for having such a resilient, long lived system (together with similarly skilled people to operate them).

Whilst it is true that one does not need to be a specialist in the technology one is auditing, it is necessary to understand how to get the right information from the staff that **ARE**. Most organizations don't have a permanent z skilled auditor on staff whose sole responsibility is to ensure that their installation is fully compliant with **all** requirements.

Required Skills	vs	Available Skills
Audit awareness		Audit averse
Risk awareness		Risk tolerance
z/OS Analysis		Audit Analysis
RACF Analysis		Compliance Specialism
Local data expertise		Departmental expertise
etc.		etc.

So it is critical to provide an aspirational view of the end audit requirement (maybe even a 5 year plan showing direction). As well as being a useful tool to obtain staff buy-in for the process, it can help to provide metrics for measurement of success. This in turn means that a departmental view of compliance against regulations can be supplied to each of the specialist areas. All of which helps to make sure that the value of regular audits is on public view.

CISA, CISM and CISSP certification is becoming increasingly ubiquitous and many organizations are driving their staff in that direction. Sadly though this can mean that education budgets are spent on certification and real technical skills can be allowed to stagnate. New functionality is introduced pretty much with every release of z/OS and the Security Analysts simply aren't being given the time to learn about them.

Be under no illusion - auditing z requires a very specialized skill set. Simply relying on checklists is not appropriate in this most complex of environments.

2.1.4 Internal vs External

The biggest difference between internal and external audits is that external audits bring in skills that may not be available at the site normally. The biggest mistake that organizations can make is ignoring the input from these external specialists! Start treating it as an opportunity for skills transfer and the whole experience will be a lot less scary.

Of course we've all been through audits where the individual did not appear to be so highly skilled (Auditors get audited too!). But any auditor with integrity will acknowledge the source of the information that they use to identify risks (so you know it isn't just something they found on Google!). They will also generally be more than happy to discuss the opinions of the technical staff. Although bear in mind that external auditors are "on the clock" so if we ask you to hold that thought it is most likely just because we are very busy. I know that I always make sure that

I make time to see the folks who had questions before I complete any evidence gathering.

I also like to ask the staff what they think the problems are. I can then gather information to help prove their point. This means that I can be the one to raise a risk report rather than the staff member.

We really do need to get away from the undercurrent of personal responsibility when it comes to risk identification. The majority of organizations that I have worked with have staff too afraid to raise risks in case they somehow get the blame for what is wrong!

Risk Management simply isn't personal - it should be about the business! And it is a process which must be supported by the management team. An entry on the Risk Log should not result in any kind of action against the raiser. We can't expect help when people appear to fear for their careers! And this type of environment leads to a prevailing attitude that security is someone else's responsibility.

Typically internal auditors are seen as something of a nuisance by the technical staff - we are taking up their precious time after all. Once they understand **why** it is so important, attitudes can start to change.

What I want to do is to get the techies to understand that audit can be their friends. We can provide you with the big stick you need to get improvement to the system seen as a priority.

2.1.5 Regulatory Compliance

All of the various compliancy regulations look at one or more of the CIA trio (**C**onfidentiality, **I**ntegrity and **A**vailability). But not all regulatory standards deal with all of the elements. I am hopeful that this brief discussion will explain why auditors ask different questions at different times.

A number of different compliancy regulations can apply to a single business and each will require separate audit activity. The broader your business, the more audit work will need to be done.

There are over 150 different pieces of IT compliancy legislation around the world and there is specialist documentation available with each of them. I'm not going to provide a synopsis for each one here so check the various requirements out on their dedicated web sites.

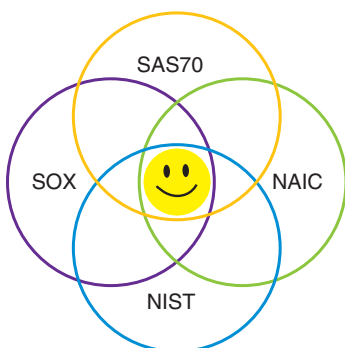
Here are a couple of examples to get the general principal across:

Sarbanes-Oxley is primarily concerned with data integrity. The focus of a SoX audit will be establishing not only who **can** change information but also who **has** changed it. There is no focus on confidentiality.

There is a rash of new SoX work happening in the UK right now as a result of the credit-crunch enforced take-over of a number of financial institutions. These were effectively local banks which used to operate only in the UK. Now they are owned by much larger organizations that do business with the USA and are suddenly liable to SoX scrutiny.

PCI is primarily concerned with confidentiality - or who can see the data. Interestingly, this seems to be the legislation which is having the widest impact right now. It's even beginning to become enshrined in State Law (e.g. Nevada has recently incorporated PCI into its legislature).

However, PCI was written to encompass a PC/server based world and some of the statements do not apply easily to the mainframe environment. There is ex-



planatory documentation available and it details where the requirements are different for mainframe.

One example is the requirement for each server to only run a single application. It would be complete insanity to run a mainframe like this. But the risk that is being mitigated by the action in the PCI legislation doesn't exist within the mainframe environment.

Whilst significant differences do exist between the various compliancy regulations, there are a number of overlap points.

You should always try to develop strategies which will allow you to maximize on those overlap points. Leveraging existing processes and information system controls can help to meet the needs of multiple different compliancy reporting requirements.

But it is rarely sufficient alone. Additional compliancy reporting requirements are an inevitable part of an expanding business.

2.1.6 The Statement of Work

You can start to see why having a Statement of Work is critical before embarking on **any** z/Audit! There are so many different areas which can be audited and so many different elements which could potentially be included that a clear definition of scope is vital!

2.1.6.1 Mapping Business Objects & Control Objectives

Today a typical System z audit is based around technological boundaries - e.g. LPAR boundary or RACF database. This simply doesn't align with the requirements of the majority of IS (Information **S**ystems) Audit activity which focus in on the Confidentiality, Integrity and Availability aspects of a line of business. In fact, it doesn't even align with the majority of the control objectives such as Identity Management, Access Control and both Detective and Preventative Controls.

One of the first things that needs to be done before an efficient Statement of Work can be assembled is to identify the relationships between parts of an organization's IS environment and their IT infrastructure. For example: The relationship between Confidentiality and Integrity of Business data is fairly simple to tie back into the external security manager policies in place at the organization.

It's not always an easy map to achieve. And it is vital that a full assessment of the value of the business data has been carried out before this process can be effective.

I find it can be quite helpful to have the external security administrators assign SECLEVEL values to the data in the external security manager once it has had its value confirmed. This makes it much simpler to group the information together for audit and reporting purposes.

But auditing System z can't just be about the data available from a single LPAR. If that data can be exposed just by making some changes in the IODF what is the point in the audit? And that is what is so new about this book - everything that has gone before has focused on LPAR level analysis. I am suggesting that this is not enough!

Auditors, Systems Programmers, Hardware Specialists and Security Analysts must learn to work co-operatively. No single discipline can be held responsible for all System z security. Disappointingly in a lot of cases, having an internal department which is named for anything to do with security (e.g. Service Security) can lead to everyone else feeling it's not their job! Security awareness training is



a must in all areas of the business. And don't forget that compliance does not, necessarily equal security.

For example: Access Control is a critical part of most businesses and is a firm focus for audit. Individual users are not responsible for the overall degree of access control employed at an organization. But they should be held responsible for any activity which takes place with a userid assigned to them. A vigilant user base makes it a lot harder for a hacker to take advantage of holes in the processes.

2.1.6.2 Deciding on the Audit Scope

It is vital to define the boundaries of what is going to be audited. It would be an impossible task to try to perform an audit (or any other piece of work) without understanding the scope of what is to be done.

Sometimes it's an easy decision to make. For example: If you are being audited to PCI requirements there is a specific list of 10 things to check. But even then it may not be clear cut. Any previously identified risks will have been entered in the Risk Log and should not be re-audited (unless the mitigation has been superseded e.g. by implementation of technology). So every Statement of Work should incorporate a definition of scope.

As an aside: I'd advise maintaining a separate Risk Log for each department. This just makes it easier to monitor progress and identify areas where security may not be taken quite so seriously. To the techies reading - you will be audited on the use of your Risk Management Process as part of any audit.

There are many opportunities to use the output from an audit for advancement of skills in an organization. Multiple reporting options against the Risk Log represent just one example.

But let's go back to scope...

There are an alarming number of z/OS sites who have not done a full analysis of the business value of their data. It's an expensive thing to do and the platform has suffered from lack of belief in its longevity for some decades. These 2 factors combined have allowed organizations to get away without performing this critical analysis.

Under such circumstances it is better to audit at the individual LPAR/Sysplex/RACF database level than not to do so at all. Do make sure that you select an appropriate boundary for the way that the system has been implemented. And be careful of "scope creep" when carrying out the audit. I'm all for technicians raising potential risks with me during an audit. But sometimes one just has to report it rather than divert time to research a new area. Sorry guys (and gals) but we're on the clock too. And lastly, remember that a partial audit is only ever going to be just that.

2.1.6.3 Audit Cost vs Organizational Benefit

Up until recently this has been quite a hard thing to quantify across skill groups. For example, trained auditors will understand the benefit of a specific audit. But sometimes we're not very good at communicating that to our technical counterparts. Please bear with us - we promise to try harder.

At the technological sharp end (e.g. Systems Programmers and Security Analysts) it can be hard to see the value of the spend - particularly when you can think of so many other things to do with the money!

Recent events (2010) in the Gulf of Mexico have brought to the fore the need to keep current with technological advances. Whether that is a deep water drilling

rig in a new geographical location or a z/OS site starting to exploit UNIX System Services is irrelevant. Think about the impact a serious hacker breach could have on your organization's reputation. I don't want to find myself in a position where I can sympathize fully with the former CEO of BP! I'd like to think that a jolly good audit would have flushed out the root of the problem or at least identified the reuse of old procedures in these circumstances as a risk.

Audit is how the business reports back to its shareholders to demonstrate conformity with the various applicable pieces of legislation. Our primary focus is financial. At the end of the day, the information that we gather from IT is only part of the input to reports which allow investors (and potential investors) to have a stab at predicting the economic performance of the business.

Shareholders (both current and prospective), Regulators and the Financial Community rely on Audit Certification of Economic Performance for making informed Financial and Investment decisions.

In other words: It's not personal! You should take no implied criticism of the way you work. Questions from auditors aren't loaded and you should feel free to share information with us. If you think that something poses a risk to any of the CIA criteria we may be able to apply the pressure to management that you need to be able to apply a technological mitigation aka **fix it!**

2.1.7 The need for continuous improvement

The IT world is a continuously changing and growing entity. This can lead to the audit process lagging behind these improvements. When the audit process catches up with the changes in technology and processes this can lead to a large number of audit failures being discovered, with little or no way of tracking whether any breaches have occurred. We all know that closing the stable door any time after the horse has bolted is of little or no use, but closing any audit gaps as soon as possible not only makes sound business sense, but in some cases can be a legal and regulatory responsibility.

There are methods to ensure that your audit processes keep step with changes to IT systems and technology. Central of these is by performing a regular review of audit plans and requirements to ensure that any newly introduced processes or systems are addressed. Any process changes that are required will then be spotted sooner.

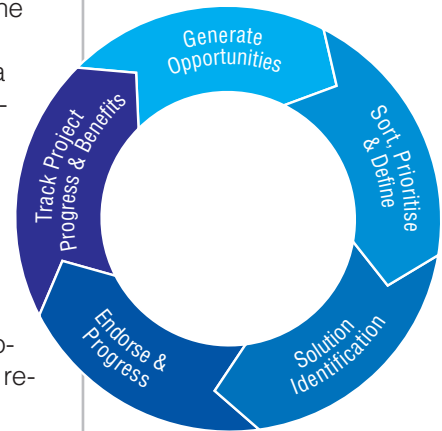
One small but important point... Cleaning up old/redundant data is vital for easing the audit work load. This process should be incorporated into Continuous Improvement.

As soon as these additional audit points are found to be necessary, they should be checked to ensure that your organization still meets both internal and regulatory requirements. Once these checks have been made and any necessary changes have been made to processes, these new audit checks can then be incorporated into the next scheduled audit and any failures will be picked up. Remedial action can then be taken to rectify any of these failures, reducing the time-span when these can be exploited.

By using this method of Continuous Improvement of your audit schedules and plans you can ensure that audit failures will be kept to a minimum. Thus ensuring that all of the easily avoidable audit points are addressed **before** they become bona fide audit failures that are raised forcing action to be taken within the auditing organization's specified time-frame.

Any changes that have to be made to your audit processes will be achieved in a

Continuous improvement causes us to look at upstream process not downstream damage control



CIP, Kaizen, Lean, Six Sigma etc.

more gradual and achievable manner. This will make sure that the changes made will precisely reflect any alterations to your organization's processes.

If this method of continuous improvement to audit processes is not used, the changes will still have to be made to the audit process. Any changes will then have to be made on a large scale. When these changes are made conflicts can arise between technical departments or business units and audit departments which have the potential to paralyze the workplace. It is not unusual for this friction to be caused by the rush to fix audit problems often resulting in the worst technical solution to the problem being used as it's the quickest to implement!

By using a continuous improvement process, your organization will be able to approach any audit in the knowledge that actions that could have been taken with all of your current in-use technology to avoid any failures, have been taken in a timely manner. It will ensure that you are ready for any changes to processes, hardware and software technologies that your organization uses, both now and in the future.

2.1.7.1 "No Walrus in the Gulf"!!!

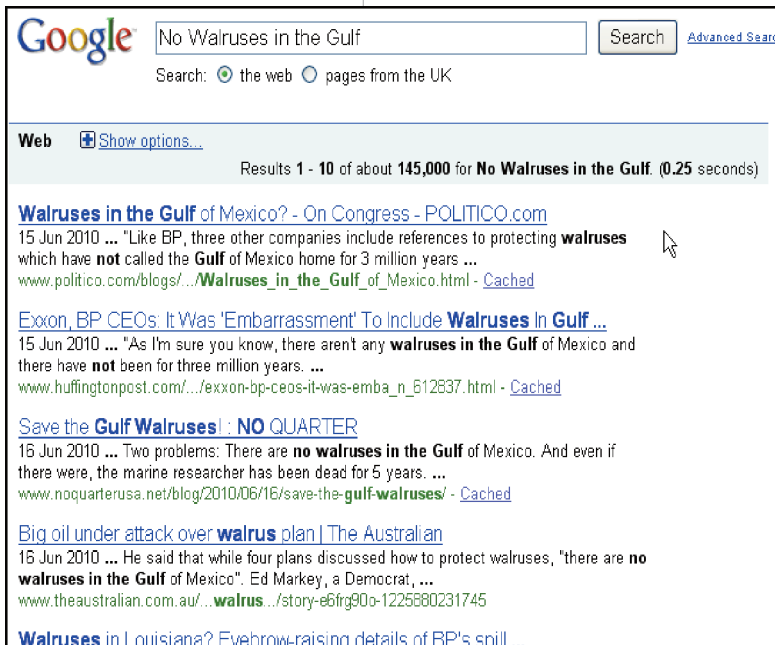
Disasters happen. There is no avoiding them completely. That's why they are called disasters not planned-outages. All we can do is try to mitigate as many of the risks as possible. But if we don't update what those risks are from time to time, we'll get caught out in much the same way as BP has during the recent disaster in the Gulf of Mexico.

Go and Google "No Walrus in the Gulf".

When I did this back in July 2010 I was returned 145,000 hits!

Almost all of them talked about the underlying breach in the risk management that was exposed at BP (although none of the oil companies gets off scot-free).

Apparently it was quite normal practice to cut and paste Spill Response Plans from pre-existing rigs. Most of the older, shallower rigs were in Alaska and the northern part of the world. Walrus are an indigenous life form up there and have to be specifically catered to in case of a bad oil spill. And it's a very good thing to be environmentally aware if you are an oil company!



These *Spill Response Plans* are such complicated, technical documents that the cost of reviewing the plans for each individual rig outweighed the perceived benefits. No one had thought that a rig which went exponentially deeper than anything which had gone before and was in a completely different geographical location changed that balance. They have been proved to be wrong.

In much the same way, it is normal practice to use the same checklist to audit System z customers around the world. This is an incredibly complicated, technical environment and it could potentially add to the cost of audit and compliance to update for the 21st Century.

But if we don't start to take a fresh look at not only the risks but also any mitigation incorporated in Risk Logs, we could find ourselves in a similar position to the oil industry. And I'm

not sure if many of my huge, international, financial sector clients could take a multi-billion dollar fine and survive at the moment!

What I am discussing in this book is a set of risks which are not new. But they have been able to be overlooked for a long time. The complexity of the environment has grown to a point that ignoring them any longer could be seen as inviting disaster on a grand scale.

2.1.8 Separation of Duties

One final parting shot before I go back to talking primarily to the Auditors amongst you...

One of the things that all auditors really obsess about is making sure that conflicts of interest cannot be used to break with defined security policies. This Separation of Duties is one of the key concepts of internal control but it is one of the most difficult (not to mention, often expensive) controls to achieve. It is all about implementing appropriate levels of checks and balances on the activities of individuals and has its roots in financial accounting systems.

Separation of Duty as a security principle is mostly about the prevention of fraud and errors. This objective is achieved by disseminating the tasks and associated privileges for a specific business process among multiple users. The principle is demonstrated in the traditional example of separation of duty found in the requirement of 2 signatures on a check. (This doesn't work so well as an example across most of Europe where only the account holder's signature is required on a paper check.)

For example that means, no matter how much you are trusted as a z/Technician, no Systems Programmer should be able to issue commands and/or make changes to the external security manager. Sorry guys - it's not personal!



3 Foundation

Evolution and Revolution - IBM first announced MVS as the operating system to run its “big iron” way back in April 1964. In the 46 years since, it has morphed into z/OS running in increasingly complex configurations on System z with the very latest zEnterprise hardware. The platform has bucked the “Technological Discontinuity” trend identified in the Stages Theory of Computer Growth, described in an early Harvard Review article¹. It has done this by continually evolving to serve Business better.

The Stages Theory of Computer Growth postulates that the complicated nature of computer technology produces a body of knowledge about the effective management of IT within an organization. Out of which emerges 4 stages of organizational learning:

Stage I: Initiation - *Characterized by limited investment and contained experimentation for proving the value of the technology in the organization.*
Proof of Concept.

Stage II: Contagion - *A period of high learning in the organization whereby technology proliferates in a relatively uncontrolled manner.*
Adoption of Technology.

Stage III: Control - *Uncontrolled growth eventually leads to inefficiency, which creates a demand for controls that slow the growth to a more manageable rate.*
Analysis of Use.

Stage IV: Integration - *The accumulated learning leads to a balance of managed controls and growth.*
Business Strength Management Processes.

I believe that System z has reached Stage IV in the process. And that now is the time to use the wisdom gathered over the preceding 5 decades to apply business strength management processes to the whole of this most critical of General Purpose Business Computers not just focusing on one element - the external security manager. Business Objects and Controls can be successfully mapped to z/Components and this document sets out to explain how.

30 years later, in 2003, another article was published in the Harvard Business Review. Written by Nicholas G Carr the piece entitled “IT doesn’t matter!”² took a more contemporary look at the topic and drew some very interesting conclusions. He talked about the impact that availability and affordability has had on IT. Attitudes over the last 30 years have shifted significantly too and where IT used to be looked upon as a low value “*proletarian tool*” which no self-respecting Manager would ever touch, now it is commonplace for chief executives to talk openly about the strategic value of IT.

The danger of this pervasive implementation and use of IT is that what gives an organization a competitive edge is having or doing something no one else can. IT is becoming so “normal” it is seen as simply a cost of doing business. This can lead to IT becoming almost invisible.

As Carr said, “*When a resource becomes essential to competition but inconsequential to strategy, the risks it creates become more important than the advantages it provides.*” The time to fully assess those risks is now.

In today’s 24x7 world even a brief interruption in IT services can spell disaster for an organization. How many stories have hit the press globally detailing just

such mishaps? No one ever intended to get hacked - but the technical details don't make it past the news publication's Editors. I'd like to "borrow" from Carr one more time:

"Worrying about what might go wrong may not be as glamorous as speculating about the future, but it is a more essential job right now."

One of the trends currently unfolding and very likely to continue is that of outsourcing z work to third parties. The state of IT discussed by Carr and others has clearly accelerated this offloading process. So discussion of the residual responsibility for oversight of third party providers should prove helpful.

As these trends continue it becomes increasingly clear that today's well managed IT enterprise will rely on an accurate mapping of its IT resources into its organizational objects. Such a resource object mapping should be viewed as the Primary Anchor Point for any responsible IT Audit.

The System z environment is becoming increasingly complex (with 5 different GA operating systems available currently and more in the pipeline) and critical to Business (some 80% of the world's business data lives behind a mainframe). But there are some damaging myths that surround it at the moment. I seek to dispel those myths and offer insight to the technical issues. For example that System z is the most securable platform available commercially is undisputed. However, it is not delivered that way. Over reliance on defaults and lack of applied knowledge can leave the platform as open to "Hacker Attacks" as any other.

Over recent years the System z audit has become something that is carried out by check list. What I hope to show is that this is no longer sufficient - if it ever was. It's time to start using the skills that exist in the industry to properly secure this incredibly flexible platform.

An accurate Object Map is The Primary Audit Control Anchor Point - These concepts, and more, will be discussed in Clear English allowing anyone to tap into some of the thought processes needed to provide effective, repeatable Audit Compliance on this very complex platform.

Taking input from a broad group of specialists working in the industry today, this document will be of use to anyone tasked with ensuring continuous compliance on System z. In an age dominated by cloud computing and virtualization which is seeing the available skill base being eroded at an unprecedented rate we need all the help we can get!

Specific points I am going to address include:

- Why checklist auditing of System z is not appropriate.
- Why taking a broader view of the problem is vital to anyone working in System z.
- How Audit can be a tool for Systems Programmers to get support for improvements.
- Why a review of the process control mechanisms surrounding the IODF must become mainstream practice.
- What makes it so important to review System z risks now?
- The implications of unrecognized or undisclosed residual obligations on IT outsourcing strategy.
- Why suggestions which might not be viewed as popular initially can reduce workloads on everyone.



3.1 General Purpose Business Computer

Let me take you back to the early days of computing. It was an exciting time when new uses for newly discovered technology were being found all the time. Machines were designed for specific purposes with little or no thought as to how to connect them together. General purpose computers were designed to solve a large variety of problems. In other words they can be given different programs to solve different types of problems.

General purpose computers can process business data just as easily as they process complex mathematical formulae. They can store large amounts of data and the programs necessary to process them. Most businesses in the 21st century use general purpose computers because of this versatility.

1953 saw the development of IBM's 701 EDPM, which was acknowledged as the first commercially successful general purpose computer. Inventor Thomas Johnson Watson Junior, son of the then IBM CEO Thomas Johnson Watson Senior, wanted to contribute what he called a "defense calculator" to aid in the United Nations' policing of Korea. In addition to the technical challenges he faced he also had to convince his father that the new computer would not harm IBM's then profitable but incompatible punch card processing business.

Only nineteen 701s were ever manufactured! They were put to use in a wide assortment of fields from atomic research and aircraft companies to assorted government agencies such as the United States Weather Bureau. From these humble beginnings the IBM 701 EDPM machines effectively paved the way for the complex, powerful, general purpose machines we know today as mainframes.

3.1.1 The need for System Integrity

1973 was a pivotal year not only because it saw the birth of Disco but also the publication of the Stages Theory of Computer Growth! Business was starting to think seriously and long term about the impact of all this new, available technology on them. And IBM grasped the opportunity to give its loyal customers the backup they needed to make buying mainframes a solid decision. A marketing campaign at the time even suggested that *"No one ever got fired for buying IBM"*.

As organizations began to rely heavily on their computer systems (24x7 operations were still a long way off but the systems were now vital to the business) it became apparent that making sure the computer system could keep running was going to be critical. At the same time applications were being rolled out globally and so it was not possible to achieve stability by just doing nothing to your systems.

The SHARE Security Project was formed (in 1972 and was at least partly responsible for IBM issuing their statement of integrity) with the mission of developing the security requirements for future IBM Operating Systems. After much discussion, the project came to the conclusion that:

There can be no System Security without Operating System Integrity.

Everyone was concerned that all the effort put into developing and enforcing good data security could be bypassed if the formal interfaces of the operating system could be bypassed and there was nothing the installation management could do to prevent it.

An Operating System Integrity exposure can be exploited in a manner that would allow a user to gain control in an authorized state. Once the user is in the authorized state, he could modify his identity and authorizations. This, in turn, would mean that the Security System would allow the user to do whatever he

wanted with no controls or journaling of those events.

Ten years later (we're up to 1983 now) the movie "War Games" was released and all of a sudden a lot more people became aware of Back Doors to computer systems and what hackers could do with them!

Security and system integrity became buzz words of the time and we were most worried about computer programs being able to bypass normal controls by doing clever things with the operating system. IBM took a leap forward in consumer confidence by announcing that they would always help fix problems that customers identified with their systems. We were buoyed up by the thought that we would never be alone if we chose IBM mainframes.

3.1.2 The Integrity Statement

IBM first issued their MVS™ System Integrity Statement in 1973. Subsequent statements for OS/390® and z/OS have stood for over 3 decades as a symbol of IBM's confidence in and commitment to the z/OS operating system.

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.

IBM's long-term commitment to System Integrity is unique in the industry, and forms the basis of z/OS' industry leadership in system security. z/OS is designed to help you protect your system, data, transactions, and applications from accidental or malicious modification. This is one of the many reasons IBM System z™ remains the industry's premier data server for mission-critical workloads.

See the full text on the IBM web site at:

http://ftp.software.ibm.com/eserver/zseries/zos/racf/pdf/zOS_System_Integrity_Statement.pdf

3.1.3 Your z implementation vs IBM's delivery practices

IBM's mainframes have earned their reputation as being the most securable platform commercially available (with ratings up to B2 levels of security). However, this is **not** the way it is delivered to the customer.

As the uses that customers have put mainframes to have increased in complexity and the number of skilled technical individuals has decreased, IBM has tried to offset by making the whole process simpler. The result is a delivery practice which is extremely open and easy to use. Good for the first time implementers but this is a long way from a secure platform as delivered.

3.2 Mainframes - The last 50 years

The cyclical nature of the computer industry can be highlighted by what has happened in the mainframe market place. The 1950s saw the introduction of the mainframe computer with IBM being the sole manufacturer of the hardware.

There was an explosion of OEM (literally **O**ther **E**quipment **M**anufacturer) vendors producing their own systems. Some were even supplied together with their own base operating systems.

This group initially consisted of **B**urroughs, **U**NIVAC, **N**CR, **C**ontrol Data, **H**oneywell, General Electric, and RCA. When General Electric and RCA removed themselves from the market they became known within the user community as “IBM and the other **BUNCH**”. The other **BUNCH** competed mainly in the Government IT procurement markets, where systems are often still seen as just another commodity.

The 1980s and 90s saw a period of extreme activity in this OEM market. This was part of a rapid expansion of technology phase with the use that mainframes were put to growing all the time. The decade from the middle of the 80s was a golden age for us mainframe geeks.

A lowering in demand following one of the early predictions of the death of the mainframe led to increased competition for the mainframe hardware market. This in turn led a number of the manufacturers withdrawing from the market.

At the same time customers were migrating towards server based technologies. This was seen as a cheaper alternative that allowed them more control over system configuration. During this time it was widely reported that the mainframe market would eventually shrink. InfoWorld's Stewart Alsop even predicted that the last mainframe in the world would be unplugged in 1996.

This prediction is particularly ironic as it was around this time that corporations began to discover new uses for their existing mainframe systems. With reduced networking costs (and now a fully functional TCP/IP stack running on z) we saw the start of a trend to move back to a more centrally controlled computing system. Combined with the explosion of online business many found that they needed the mainframe back end for transaction processing and data storage. Now we have returned to a world where IBM is the only hardware manufacturer with the latest zEnterprise introducing *A New Dimension in Computing!*

The trend towards a more centralized computing ethos continues to this day. In the 10 May 2010 edition of NEWSWEEK, an article was published that focused on the moves that organizations were taking towards a more controllable, centralized computing strategy for their business. Virtualization has become a key word when discussing IT strategies today and the mainframe can play an enormous part in this.

The development of Linux for System z, which arrived at the end of the 1990s allowed customers to reduce the amount of hardware that they were required to maintain and license, and in late 2000 IBM introduced the 64bit z/Architecture. Throughout the following decade hardware shipments followed a steadily climbing trend.

3.2.1 From one to many Virtual Machines

The concept of Logical Partitions (LPARs) was developed by IBM in the late 1960s and was introduced commercially in 1972 for use with the VM operating system. These concepts now form the basis of the latest trend of “virtualization”.

The PR/SM Hypervisor was also introduced which allowed multiple LPARs to share access to physical resources such as Storage, DASD and CPUs without the need to use VM. This allowed customers to maximize the utilization of their expensive hardware by hosting multiple business streams on the same physical hardware. The results of tasks completed on one LPAR could be shared with another or alternatively each LPAR could be kept standalone - the common

choice becoming a mixture of the 2 as dictated by business needs.

The latest IBM zEnterprise can be configured to run many hundreds of virtual servers, while still managing the more traditional data processing tasks under z/OS. The hardware can be shared between the LPARs which can lead to higher data availability for both the business and their customers alike without necessarily a reduction in security.

3.2.2 Current Trends

The world is currently working towards a greener more efficient mainframe computer where the ability to run multiple versions of multiple operating systems on a single hardware installation is no longer just an option, but a business requirement.

LPAR and PR/SM configuration changes can be made dynamically, allowing the addition of new devices, or changes to their configuration to be made without impacting availability. Roll out of new operating environments can be done in just minutes!

When delivered, IBM mainframes sometimes have more processing power and memory installed than the customer has paid for. IBM introduced Capacity on Demand to allow this additional processor capacity and memory to be made available for a fee when required, whether temporarily or on a permanent basis. Depending on the specific contract with IBM, these changes may be made dynamically by the customer.

The scalability of the mainframe computer has led to a partial exemption from the technology cycle where x86 servers may need replacing as workload increases. With the mainframe, additional processing power can be installed as hardware if it is permanently required for the organization. If the need for additional processing capability is occasional, or cyclical, this requirement can be met dynamically, purely by allocating the additional resources “on the fly” to the LPAR while the task is running, and switching it back on completion.

Some workloads will remain suited to distributed server systems. The new zEnterprise System with an attached zBX allows the parts of these systems that are better suited to distributed platforms to be run on server blades. This work can be run on the distributed server platform, while sharing storage and configuration tasks with the mainframe environment.

Distributed servers will still be used in the data-center, but the BladeCenter extension allows some of these servers to come under the System z umbrella. One example of this is to run the application tier of an Oracle instance on the zBX blades, while the database tier is run on virtualized Linux for System z platforms hosted on the z196. While the application tier could still be run on distributed servers, using the blades in the zBX will not remove any functionality, and all communication between the application tier and the database is performed under the control of the zEnterprise System. This also allows for the scaling of resources available as the database grows in size.

The new zEnterprise machines are actually delivered with all (up to 60) available LPARs pre-configured. They also provide a sort of “Plug and Play” type of capacity for FICON (**F**iber **C**ONnectivity) attached devices! This represents a dramatic change from previous iterations of the mainframe hardware. It also represents a golden opportunity to our wily hackers!

3.2.3 Current z/Hardware Platform

The latest iteration of IBM’s Big Iron is the zEnterprise. Combining massively

improved performance over its predecessor with built-in support for the IBM zEnterprise BladeCenter® Extension (zBX) it represents a world first in technology offerings. It was designed to boost scalability, performance, security, resilience, availability and virtualization.

IBM says:

IBM® zEnterprise™ System is a first-of-a-kind workload-optimized multiplatform technology offering. The system consists of the IBM zEnterprise 196 (z196) central processor complex (CPC), the IBM zEnterprise Unified Resource Manager, and built-in support for the IBM zEnterprise BladeCenter® Extension (zBX) Model 002. The IBM zEnterprise 196 is designed with improved scalability, performance, security, resiliency, availability, and virtualization. The z196 Model M80 provides up to 1.6 times the total system capacity of the z10™ EC Model E64, and all z196 models provide up to twice the available memory of the z10 EC. The zBX infrastructure works with the IBM zEnterprise 196 to enhance System z® virtualization and management to deliver an integrated hardware platform that spans System z mainframe and POWER7™ technologies. The IBM zEnterprise Unified Resource Manager, delivered with the z196, is designed to deliver end-to-end virtualization and management along with the ability to optimize technology deployment according to individual workload requirements.

Today's mainframe is usually to be found at the center of an organization's IT strategy. With its ability to store and process high levels of data the mainframe has regained its reputation as the most efficient and cost effective method of data-processing and with the ability to run multiple operating systems this data can be easily shared across an entire organization.

Reliability, coupled with the fact that System z can be configured to be the most secure platform available gives the ability to allow 24 hour access to data in an efficient and timely manner. This is particularly useful for banking and government organizations where dealings across time zones are common place.

With the ability to 'Hot-Swap' hardware within the mainframe, it is now feasible to upgrade your hardware to meet climbing demands, add new storage and make changes to the base operating systems all without affecting system or data availability to your customers.

3.2.3.1 Advantages – Total Cost of Ownership

One area where the modern mainframe leads the way is in offering the ability to drive down the total cost of ownership (TCO) for an organization. However, in driving down this cost using the methods described below it is possible to open up the potential for Front Doors, where access to systems and data is granted purely in the drive to consolidate systems or increase efficiency.

The zEnterprise takes advantage of high capacity data-processing ability at a high utilization rate. The ability to automate even the most complex of system management tasks, together with the reliability of the hardware allow for a reduction in the TCO. Automation of any task related to system configuration can lead to resources being made available where they are not required. Additional checks must be included in the customer's processes to ensure that this has not occurred.

System z also drives lower TCO by excellent utilization (up to 100%) and also by combining specialty engines, superior systems management, and excellent overall economics.

The specialty engines introduced by IBM allow certain types of code to be run outside of the main z Series CP using a different charging model for the use of



these specialized processors. When compared to using the control processor, there is no MSU increase when work is processed by specialty engines. This can reduce the overall cost to the customer significantly due to IBM's pricing and licensing model for these engines.

While the actual processor (from a hardware point of view) is no different for these engines, the way that the information is handled and processed alters due to the microcode that is used to run the processing. This microcode can lead to much more efficient data handling, increasing the amount of data that can be processed in the available time.

The first specialty engine released by IBM was the ICF or integrated coupling facility, which allowed multiple processors to share and update data. This functionality is vital for a parallel sysplex to be able to function.

The IFL (Integrated Facility for Linux) processor was the next to be unveiled. It was introduced in September 2000 to encourage large Linux users to migrate their workload to the newly re-branded zEnterprise Platform. In 2004 IBM followed up with the zAAP (System z Application Assist Processor) engine which is designed to switch Java and XML workloads away from the Central Processor.

The specialized zIIP (System z Integrated Information Processor) engine was introduced by IBM in 2006 initially with just the ability to offload DB2 processes from the core CP. However this support was soon extended to additional IBM processes such as TCP/IP IPsec, communications server. Some general XML processing can be handled by the zIIP rather than the zAAP engine. DB2 v8 and above is able to exploit zIIP capabilities for specific data and utility workloads, with v9 being able to use an increased amount of parallel processing. This allows it to use the zIIP engine more, further reducing the load on the Control Processor.

The zIIP engines are now exploited by a number of 3rd party vendors including CA Technologies, BMC and DataDirect Technologies.

These specialty engines allow customers to farm out particular processing tasks, freeing up the main processor for more traditional data-processing tasks. This can significantly improve the performance of tasks, as the CP can switch to other processes, while the specialty engine deals with its assigned section of the code.

While each installation is unique, reductions in the total cost of ownership can usually be achieved in:

- Reduced downtime costs
- Lower cooling costs
- Less power usage
- Faster workflow times
- Fewer software licenses required
- Reduction in hardware footprint

3.2.3.2 Environmental – Green Movement

Many organizations are now pushing (and indeed are being driven by legislation) to reduce their carbon output. A popular option being used to accomplish this is the consolidation of existing Linux server farms currently run on x86 architecture, to run under z/VM on the mainframe. It has been estimated that the average x86-Linux installation is only actively utilized at approximately 5%, whereas the mainframe, with its ability to release hardware resources when they are required has a utilization rate of around 90%.

It is not just the reduction in power required to run the mainframe that is affected

by this consolidation process. Migrating installations to Linux for System z dramatically reduces the amount of physical space required. By reducing this space requirement, your cooling requirements are also substantially lower, leading to additional power savings in the running of your air conditioning systems.

There are however additional security and audit checks that need to be put in place when implementing Linux for System z (or for that matter any virtualized system). Devices can be configured as shared and data stored under these virtual systems may be accessible from other VM LPARs or from the z/OS platform without security having been configured yet. There is no need for a potential hacker to search for a back door to access data if the front is left unlocked and wide open.

This massive reduction in power requirements for your business is not always the starting point for the planning of a greener IT installation. When approaching their power supplier with future requirements, some organizations have found that the supplier is just not able to offer the level required to run their traditional server farm. And this is where the mainframe has come to the rescue, offering the ability to run hundreds of virtual servers on a single piece of hardware, together with a more efficient use of power, offering a higher level of performance per kilowatt.

In the 2008 Red Paper “Going Green with IBM Systems Director Active Energy Manager”³ IBM announced their plan to consolidate 3,900 of their own servers onto approximately 30 System z mainframes, stating that they estimated a reduction in energy requirements of approximately 85% by doing so. This was announced as part of IBM’s Big Green initiative.

3.2.3.3 Managerial – Footprint Consolidation

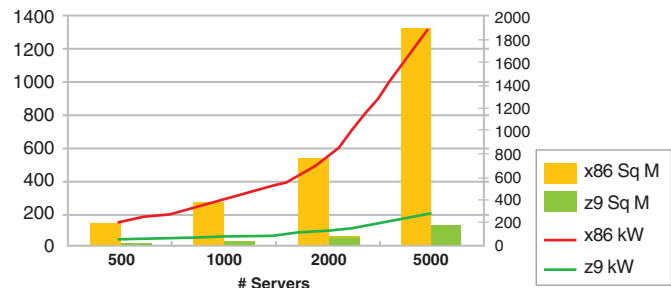
Server and footprint consolidation is now a high priority for many organizations, allowing reduction in administrative overheads, data center space and power requirements.

I was at the meeting where IBM proudly announced the new zEnterprise Platform on July 22nd 2010. This new zEnterprise model is designed with footprint consolidation at the forefront of its considerations. IBM stated:

IBM System z is taking a bold step into the future. For the first time it is possible to deploy an integrated hardware platform that brings mainframe and distributed technologies together - a system that can start to replace individual islands of computing and that can work to reduce complexity, improve security, and bring applications closer to the data they need.

IT today is all about creating an infrastructure that is dynamic and scalable as well as being flexible enough to satisfy both the needs of mission-critical work and the development and deployment of new workloads. This infrastructure must be able to make sense of data, a company's most valuable asset, with insight rather than hindsight, and it must allow a business to use IT to gain a competitive edge. In building such systems, multiplatform solutions have become the norm for handling computational acceleration and specialized processing. As these heterogeneous systems grow, the end-to-end management can become a burden on resources and the IT budget. A new technology is needed that can go to a new dimension in computing, where smarter systems and smarter software work together to address the needs of the business.

Dramatic savings in power, heat and space...



...up to 90% saving with tradable carbon savings

Today IBM is introducing IBM zEnterprise System - a system that combines the gold standard of enterprise computing with built-in function to extend IBM's mainframe-like governance and qualities of service to special-purpose workload optimizers and general-purpose application serving. End-to-end management is enabled for this heterogeneous environment by the IBM zEnterprise Unified Resource Manager, which provides energy monitoring and management, goal-oriented policy management, increased security, virtual networking, and data management, consolidated in a single interface that can be tied to business requirements. The IBM zEnterprise System is comprised of the IBM zEnterprise 196, the IBM zEnterprise Unified Resource Manager, built-in support for the IBM zEnterprise BladeCenter Extension (zBX), and optimizers or IBM blades.

The IBM zEnterprise 196 is designed with improved scalability, performance, security, resiliency, availability, and virtualization. The new 96-way core design delivers massive scalability for secure data serving and transaction processing for large-scale consolidation. As environmental concerns raise the focus on energy consumption, the IBM zEnterprise 196 central processor complex (CPC) is designed with new energy efficiencies that can reduce energy use and save floor space when consolidating workloads from distributed servers. For clients looking to build green datacenters, optional water cooling and high-voltage DC power allow a bold step into the future of cooler computing and increased energy efficiency without changing the system footprint.

The IBM zEnterprise Unified Resource Manager manages System z ensembles -- collections of one or more zEnterprise System nodes in which each node is comprised of a z196 and its optionally attached IBM zEnterprise BladeCenter Extension (zBX) Model 002. An ensemble can consist of a single z196 with no zBX attached, or 2 to 8 CPCs where at least one of the z196s has a zBX attached. The resources of a zEnterprise System ensemble are managed and virtualized as a single pool of resources integrating system and workload management across the multisystem, multitier, multiarchitecture environment.

This is a brave new world we find ourselves in. We must adapt and learn or else, rather than being part of the solution, we really will find ourselves as irrelevant in the 21st Century.

3.2.3.4 Futures

Very strong rumors (backed up by public comment from IBM) suggest that the future of the mainframe looks destined to see the introduction of x86 instruction emulation. This would allow the installation of x86 server and desktop software, both Windows and Linux under z/VM, where applications could be delivered directly to the end-user's desktop from the mainframe. This would allow the mainframe to compete in this still emerging market.

From a security and configuration point of view the mainframe offers a much more centralized focal point of control. This can ease the introduction of new systems, ensuring that from an audit point of view any changes that are made meet internal policy requirements, together with any mandatory legal requirements.

The technology cycle that exists has also now returned full circle. In the 1980's and 90's we saw the move from mainframe to Client/Server architectures. Now we are seeing this application workload returning to the mainframe once more. Improvements to System z, in both hardware and software, since the 1980's have increased the amount of the Client/Server workload it can manage. For example services such as TCP/IP, Email and web server applications are now supported.

Over the same period massive increases in the workloads Client/Server solutions are supporting have revealed the true long term TCO for this type of architecture. This has allowed the mainframe to compete for more of a business's computing needs, not just the traditional number crunching workloads, based on the TCO **and** what kind of 'service' that actually buys you.

Another technology emerging from the x86-world is Cloud computing. Initially driven by key Internet players such as Amazon and Google, it offers Users access to web based computing resources using a pay-for-use or utility computing business model. From the users perspective they have a user focused interface that shields them from most of the technical aspects of the service. They also get an SLA that, from their perspective, covers the complicated issues of availability and recoverability. And finally they can add or remove resources at will (depending on any contractual requirements).

The provider of any cloud computing service has to take these needs and turn them into a physical reality. To do this they must create a secure, affordable, environmentally friendly system that is capable of meeting all of the user's requirements, including service level agreements, while remaining within budget.

This is probably starting to sound rather familiar - yes I'm afraid the Emperor **is** naked! In essence cloud computing is the same concept as was popular when I first started in mainframes, only we called it facilities management back then. The technologies are more complex now and the turn round times much shorter but the same core challenges remain.

External, outsourced service provider companies have proved that the mainframe is perfect for sharing hardware between customers, with varied and often complex workloads, while still providing a secure system that meets SLAs. Bringing the mainframe, and all that it offers, to the cloud computing environment is just the next logical step.

With its ability to provide scalable virtualization System z servers could make an obvious choice for cloud computing. They are able to provide thousands of virtual Linux instances all securely sharing resources and components; all backed by a robust system management structure. This can reduce the total operating cost, while significantly increasing the processing and storage capacity.

3.3 The Parallel Sysplex

A parallel sysplex is a group of LPARs on one or more mainframes configured to act together as a single system. It combines data sharing and parallel processing allowing systems to share the workload, leading to better availability and increased performance.

Parallel sysplex centers on a sysplex timer, to ensure that events are synchronized between systems. It also contains a Cross system Coupling Facility (XCF) to allow the systems to communicate and Global Resource Serialization to allow concurrent access to resources by all systems, only reducing to exclusive access where necessary.

3.3.1 The need for more Processing Power

The drive from customers to introduce additional functionality to the mainframe and products that run on the platform is constant. This pushes the need to have additional processing power and resources available so that this new functionality will not have an impact on overall system performance.

Database management system requirements have expanded exponentially in

recent years with the increase in the amount of data stored. Online accounts together with a perceived need to spread one's assets amongst a number of different financial institutions has only added to the amount of data that is stored, and therefore needs to be managed. The expansion of the mainframe platform into the online world has led to the need for web servers, XML and java support.

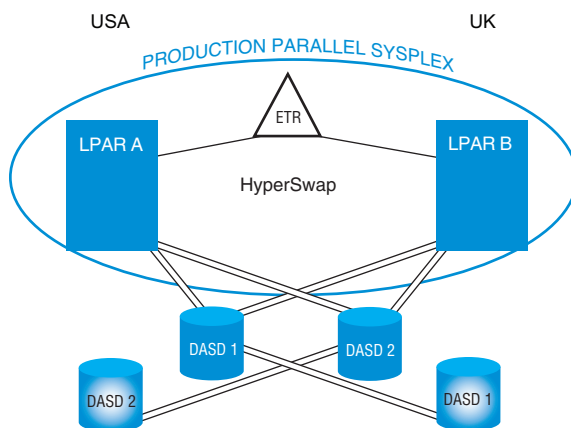
The Parallel Sysplex configuration method of linking different systems, and farming out work to the system that has the spare capacity to perform the task without impacting online performance allows for a much more flexible method of workload balancing.

3.3.2 Massive Parallelism

There has been much discussion from various industry commentators speaking about "virtualization" and "massive-parallelism". Well, I hate to brag (who am I kidding?) but we've been doing both for decades in the mainframe world.

Wikipedia has the following to say:

A massively parallel computer is a distributed memory computer system which consists of many individual nodes, each of which is essentially an independent computer in itself, and in turn consists of at least one processor, its own memory, and a link to the network that connects all the nodes together.



System z was actually designed to accommodate efficient implementation of parallel computing. Parallel Sysplex was an expansion on the capabilities of the hardware platform that was needed to cope with the massive amounts of processing and data storage going on in the environment. Further support for massively parallel implementation is being released by IBM all the time. One of the most significant announcements in recent years was GDPS.

Described by IBM as "The ultimate Disaster Recovery and Continuous Availability solution for a System z multi-site enterprise", the Geographically Dispersed Parallel Sysplex is an extended Parallel Sysplex where the mainframes can be located in different buildings or cities. Automation, as well as specialized hardware implementation, plays a large role in the recoverability aspects of GDPS.

In synchronous data mirroring (GDPS/PPRC) mode there is a limit of 120 miles between sites. In a 2 site configuration this allows for switchover in the event of a failure, recovery can take place with little or no loss of data, potentially with no disruption to Users. This offers the ultimate in resilient, "hot swap" IT environments but can be expensive to implement.

The asynchronous remote copy facility has no limit to the distance between sites, meaning that the mainframes can be in different countries or even on different continents! Asynchronous implementation, or "warm swap", is much less expensive to operate. However there is less resilience and it is more likely that there will be an interruption to the users in the case of major systems failure.

3.4 Operating System Selection

System z offers a unique perspective on operating systems. There are currently 5 different, generally available and supported operating systems which can be run on the zEnterprise hardware (6 if you include the ones in the optional zBX frame - AIX and POWER7 with a statement of direction to include Linux in the

future). They range from the open source z/Linux through to the highly specialized business engine that is z/OS. The list looks set to grow in the near future as System z continues to evolve alongside business requirements.

3.4.1 What is it?

At a most fundamental level, the operating system is the specialized set of programs that allows your applications programs to communicate with your computer hardware. Deciding which operating system to pick is often a decision that is made based on an arbitrary requirement of a new software package that the organization is investing in.

3.4.2 Advantages and Disadvantages

Being offered choice always brings both advantages and disadvantages. The situation is no different when it comes to operating systems. The advantage of having multiple operating systems on one hardware platform is seen as the customer having control over what and how their business will run. Smaller organizations can save money whilst larger companies have the option to make economies of scale.

The biggest disadvantage of the 6 operating systems which can be used to run your z hardware is that personal opinion can often end up influencing business decisions.

Another is the different skills that may be required to secure data stored under these operating systems. A highly skilled z/OS External Security Manager specialist may not have the required knowledge to configure security for Linux for System z. With the introduction of the zBX, presenting the customer with the opportunity to run AIX under the mainframe umbrella, sharing data and devices with the z196 could leave data unsecured. This draws attention back to the fact that Front Doors are the easiest way to gain access to data, where this access may not be recorded by the systems.

The logical solution then is to pick the platform/operating system combination which make most sense for each new application. With System z, the system support team (from the z/OS Systems Programmers through to the Security Analysts and beyond) can provide a stable, secure platform with limited additional start-up costs making it an obvious choice for new business projects.

3.4.3 z/Platform

At the time of publication of this book, IBM provides a choice of 6 different operating systems to run on the z/Platform. Each offers specific possibilities to the customer wanting to get value for money from their investment in hardware with the lowest mean time between failures available.

zBX

Whilst not technically an operating system this option (which is new to the zEnterprise hardware announced on July 22nd 2010 but will also be retro fitted to the z10) will affect operating system choices. It allows for POWER7 & System x Blades to be incorporated into the mainframe hardware cabinetry providing a highly resilient hardware infrastructure that supports the multiplatform environment of z196.

z/Linux

The darling of the new technologists that's busy invigorating the mainframe industry. Developed at IBM's labs in Böblingen, Germany, z/Linux exhibits the

close ties to z/VSE that one would expect from teams working at the same facility. Additional flavors of Linux that run under System z are available from Novell and RedHat.

z/OS

The industry standard, high powered, business engine of choice for large and very large organizations.

z/TPF

The airline control system used by all but the tiniest of operators. This truly is an example of world dominance in an industry sector. Non-airline related industries need not apply though.

z/VM

Whilst it is also a business-strength, operating system in its own right, z/VM has gained traction in recent years due to its enabling properties. z/VM allows an organization to run many thousands of “Guest” operating systems turning z10 into the world’s most power efficient, super server.

z/VSE

Seen by some “Old Timers” as the baby of the z operating systems because it was the last to be developed - before z/Linux that is. Still popular in Germany z/VSE makes a great, low-cost option for running CICS.

3.5 The External Security Manager

When first installed, IBM’s z/OS and z/VM operating systems do not include a fully secure-able system as might be expected from most commercially available Operating Systems. The mainframe approach is to use an external security manager to provide control over access to resources.

There are 3 commercially available external security managers for the z/OS operating system. These are IBM’s RACF, and the 2 offerings from CA Technologies, CA ACF2 and CA Top Secret, both of which come under the company’s eTrust group of products.

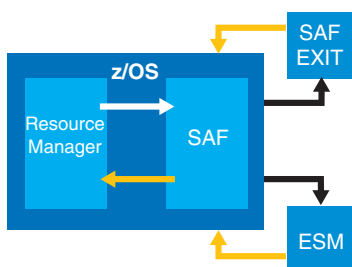
Each of these External security managers can provide the security required for the installation. The specific functions can include Identification, Authentication, Access control, Policy Administration and audit tracking. And the functions can be LPAR specific or Sysplex wide depending on configuration of the external security manager.

3.5.1 What is it?

z/OS provides a direct interface for External Security Managers (ESM) in the System Authorization Facility or SAF.

Applications (or application environments like CICS or ISPF) though are ultimately responsible for deciding who can gain access to specific resources. However they can pass a request for access to the ESM, usually using something called RACROUTE. This is sent to SAF which passes the information to the ESM. The ESM then searches its data banks to see if the user is permitted to use the resource and sends a return code to SAF - e.g. RC0 means that the user is defined and access has been found to be allowed. SAF then communicates this to the originator of the request. The application decides what to do in light of the return code. This process is invisible to the user unless their access request is rejected.

You can look at IBM’s RACF, CA’s ACF2 and CA’s Top Secret as being the applications that enable the use of resource level security within a z/OS environment. This level of security must, of course, be audited. In fact it is all that is currently



audited and is often representative of just a subset of the entire z/OS security footprint that should be examined as part of the audit process. What I hope to show with this book is that we need to be looking at a much wider picture than we do at the moment.

3.5.2 Advantages and Disadvantages

An external security manager package should **ALWAYS** be used. Without one in place there can be no control over a user's level of access to resources.

An external security manager can also stop any attempt to log on to your system by unauthorized personnel. This provides protection for even the most basic of information held on your system ensuring it can only be accessed by authorized users.

When a user is authenticated on your system they are identified to the security manager dependent on the UserID information provided. This allows you to track each user's access of resources, including logon times and locations together with which datasets, applications, etc. the individual UserID has used during their current authenticated session. This can provide your organization with an easily track-able and auditable record of access requests to resources on your system. A word of caution - it's not always possible to definitively tie the UserID to a human being unless the original ACEE (**A**ccessor **C**ontrol **E**nvironment **E**lement - the control block which contains the majority of the user information) was built using some form of secure, external, biometric validation process.

You can see the disadvantages of not using an ESM: No trace-able path of who used your system, what resources they gained access to or what they did with these resources once they had access to them; Changes could have been made, copies taken and sent to competitors or other interested parties. All of these would not only be a breach of most organizations internal security policies, but also a serious breach of many countries legal requirements regarding the protection of data.

A badly configured (and/or administered) ESM is also a dangerous thing. Incorrectly specified security settings are the easiest way to leave a Front Door (or Back Door!) open to sensitive data. The first thing a potential hacker will do is try to access the data directly, without any tricks involved. The security team needs to know what and who needs access to data and to what level. Setting a dataset's protection so that everyone has READ access does not stop copies being taken. But stopping the people who need access getting it through official routes is a guaranteed way to force them into "creative" practices!

While some Developers or Systems Programmers may count some aspects of using an external security manager as a disadvantage, this is not the case and most attempts to prove otherwise would more rightly be described as an inconvenience to the individual rather than a genuine disadvantage.

3.6 Application Focus

System z was born out of an era where technology was implemented for the sake of that technology alone. Things have moved on and, in the 21st century, it's all about the Business.

With the entire infrastructure being much more business orientated, systems and security staff have come under pressure to deliver working and secured application systems to tighter and tighter schedules. Administration and the securing of these systems are also performed by smaller and smaller teams.



Simple documentation of configuration information may not always be completed in the rush to deliver the system to production, but this can lead to a position where there is no easy to read record of which systems are connected to which devices. What may appear outwardly to be configured as a secure system may have been mistakenly set up allowing unsecured access to data from an undocumented system.

Processes and procedures surrounding System z may not have adapted at the same pace. Re-designs aligning applications processes to business requirements are becoming de rigueur. It's time to start thinking the same way about the systems hardware and a little used feature of the IODF (**I**nput/**O**utput **D**efinitions **F**ile) may be just the thing to address the issue - more on this later.

3.6.1 Business Objects

Every business has an Organization Chart. It doesn't matter if you are Microsoft or a "Mom & Pop" operation (and for the British readers - "One Man and his Dog" company) you understand the overall structure of your organization.

An organizational chart (I love the term - **organigram**) of a company usually shows the managers and workers who make up an organization. It also shows the hierarchy, or relationships of directors: e.g. managing director to chief executive officer to various departments. In many large companies the organization chart can be incredibly complicated and is therefore sometimes dissected into smaller charts for each individual department within the organization.

The organizational chart then can be used to show the discrete parts of your business, regardless of takeovers, mergers and the like. You can think of these as the "Lines of Business" and these are the elements to which compliancy regulations apply.

Audits and controls are applied to these lines of business and it's your job to find out which parts of your IT infrastructure are related to them.

3.6.2 Business Controls

In accounting and auditing, **internal control** is defined as a process affected by an organization's structure, work and authority flows, people and management information systems. They are designed to help the organization accomplish specific goals or objectives.

It's a means by which an organization's resources are directed, monitored, and measured and plays an important role in preventing and detecting fraud and protecting the organization's resources.

Internal control procedures reduce process variation, leading to more predictable outcomes. Internal control is a key element of the Foreign Corrupt Practices Act (FCPA) of 1977 and the Sarbanes Oxley Act of 2002, which required improvements in internal control within United States public corporations.

The increasing level of applicable compliancy regulations may, in part, be driving the current trend to re-map IT functions to the business. And the more complex your environment becomes, the more critical it will be to make sure that this process is undertaken.

3.6.3 Disaster Recovery

Disaster recovery (or Business Continuity as it is now frequently known - less negative connotations) introduces additional challenges to the business controls placed on the IT systems. The primary purpose of DR is to allow the business to

carry on despite the occurrence of a situation which would normally cease trading.

Normal business controls (e.g. retention of audit records generated during the DR window which are vital to tracking activity after the event) are often bypassed in favor of simply getting the system back up.

Whilst this may currently be deemed acceptable, it is avoidable. Not that disaster will occasionally strike (no one can stop earthquakes occurring on the San Andreas fault line or a volcanic ash cloud drifting across Europe for example) but by performing regular testing you can ensure that normal business controls do not **need** to be bypassed in these circumstances.

One of the reasons that business controls are often abandoned during real (rather than simulation/testing) DR processing is that it takes place off-site. That is all the hardware, from the mainframe out to the network, is completely different from those elements in use for day-to-day operations.

The problem is that even something as simple as being on a different TCP/IP subnet can be catastrophic to tasks expecting specific values - e.g. a Firewall only having entries for the normal network might leave back doors open that could be exploited by an opportune hacker. And this pales into insignificance as an example when you know what damage can be done with an open Front Door to the mainframe.

Regular testing of the procedures is an absolute must. It needs to be frequent enough that all of the staff needed view it as a normal part of their job. The majority of testing can be carried out on-site but there needs to be one or 2 tests each year which are full simulations of a disaster. These will need to be done off-site. And they must be treated as opportunities to make sure the systems can be restored without exposing the organization to any additional risk. Let's face it - things are tough enough already if you've had to invoke DR without having to worry about hackers exploiting your situation!

3.6.4 Asset management

Asset management is another one of those disciplines which have expanded and matured over the last decade or so. IT Asset Management or ITAM business practices are process driven and matured through iterative and focused improvements.

Most successful ITAM programs are invasive to the organization, involving everyone at some level. This is not an exhaustive list but such personnel as end users (educating on compliance), budget managers (redeployment as a choice), IT service departments (providing information on warranties), and finance (invoice reconciliation, updates for fixed asset inventories) will all be involved.

This involvement is often seen as a nuisance to day to day operations by the IT departments affected but Asset Management is critical in ensuring that all resources are identified to the business. Without it an audit becomes a much more high impact process than even the Asset Management!

IT asset management generally uses automation to manage the discovery of assets, so inventory can be compared to ownership information. Full business management of IT assets requires a repository of multiple types of information about the asset, as well as integration with other systems such as supply chain, help desk, procurement and HR systems.

This information can then be used to apply controls to who has access to the various resources. And, in turn, this leads to the ability to be able to account for



use of those assets. This, at the end of a long chain, enables all sorts of things from ease of audit to charge-back on resource usage.

3.7 Using the IODF as the z/Audit Control Anchor Point

Let's go back to an idea I introduced earlier - "Re-designs aligning applications processes to business requirements are becoming de rigueur." A great example is the newly introduced zEnterprise and its Unified Resource Manager (URM). Using URM business object and control object workload can be mapped directly into a collection of diverse zEnterprise resources to create a resource ensemble. It's time to start thinking the same way about the systems hardware and a little used feature of the IODF may be just the thing to address the issue.

OK, so you may be thinking this is an idea from left field but the IODF should be viewed as **THE** central point of control for the whole of a System z installation. It represents a map for the way that the whole thing hangs together - both currently **active** and **potential** environments. The IODF can be seen as the blueprint for your System z environment. And, as we see much larger numbers of LPARs able to run in one processor complex, effective use of the IODF is critical.

Historically not much attention has been paid to the IODF contents. Most organizations will make sure that the IODF dataset(s) have adequate protection through the external security manager. Many will make sure that there is some physical separation of the HMC (**H**ardware **M**anagement **C**onsole) but very few are using the mapping within the IODF to define a picture of their complex. In fact, very little in the way of "Best Practices" documentation exists for the IODF contents at all.

This is one area of System z that few people are entirely comfortable with. So expect a little resistance when the subject is raised. Partly this is because the interface used to update the IODF is so darned complicated but also getting it wrong could be disastrous for the installation!



Misunderstandings exist about the content of the IODF - I know I was taught that all of the DESCRIPTION fields should contain the SMFID of the system. That's not the case and it is this DESCRIPTION field that can be exploited to successfully map the System z resources to Business Objects.

Getting on top of the content of your IODF can significantly increase your capability to be audited successfully. Let's take just one example: in order to fully accept responsibility for mainframe security requires that you identify any shared CHPIDs (Channel Path IDentifiers - at its most basic, the function by which peripheral hardware is connected to the mainframe in the IODF) you have defined on your system. There are 2 ways to approach the exercise to find out if you do have them. The first option is to compare your SMF data for a full year to find activity to shared CHPIDs. On an average sized System z this can take anything up to a week per SMF stream - making the minimum elapsed time to extract the comparison information this way around 2 weeks! The second option is to use the contents of the IODF to find any shared CHPIDs. With a well-defined IODF this can take just minutes.

And this is starting to go global! Denmark has recently implemented legislation that places a requirement on all Facilities Management organizations to prove that there can be no crossover of access to customer data (something which **could** be achieved with shared CHPIDs). If this hasn't hit you yet, you can bet the farm that it will soon. The time to start treating your IODF as the major control point in maintaining the accuracy and integrity of z/OS and its associated hardware has arrived.

Details on how the issue can be exploited as well as what to do to mitigate will be given in later chapters. For now I want to focus on why the IODF is so important as well as the strengths of using z element description fields to map your business to System z.

3.7.1 Why the IODF is Critical to System Integrity

The Input/Output Definition File (IODF) is the set of logical configuration statements that are used to define a network of hardware resources. These resources are generally available to both the z/OS operating system (OSCP) and the z/OS platform hardware (IOCP) as well as related ESCON/FICON Directors (SWCP), if any.

I've said something similar a number of times already, so what do I mean by it? The IODF is the mechanism by which you define the shape of your z environment. It not only maps those systems which are active on your z/Platform but also all of those systems which can be activated at any time in the future - including things like which operating system configurations can be used, what DASD can be connected and how TCP/IP traffic is routed.

What's the problem with this then? Well, partly it's the complexity of setting up new systems - which often leads to existing ones simply being cut and pasted leading to all sorts of duplicates and device sharing which can, in turn, lead to system integrity problems. But also it's a problem which has been amplified by IBM's practice of sending the System z hardware out pre-configured for all possible 60 LPARs - which potentially opens up all sorts of doors for the wily hacker.

In the pre-z10 days a customer had to pay for using the additional storage (HSA) that was needed to have additional LPAR definitions in the IODF. This meant that people kept a much closer eye on the contents. Now there is hardly any technical incentive to do so. Audit must take the lead on resolving this problem.

The IODF has not had much change since the early days of its inception so remains one of the more difficult areas of the z/OS functions to support. Where once most Systems Programmers understood the IOCP macros, today there are few that are trusted to delve into HCD and HCM. Changes in hardware are essential but getting the view of the physical and logical connections is near impossible!

The balance, or in reality the lack of balance, found in most IODFs has a major effect on the integrity of the z/OS Sysplex and its images. This includes for instance the correct device pathing and connections as well as the efficient use of virtual storage. Lack of integrity within the IODF can lead to IPL failures, loss of access to subsystems and the inability to run critical business applications on the System z platform.

The IODF is not just a configuration file it's the **PRIMARY CONTROL FILE** for System z. It defines the placement of hardware as well as the relationship between the various resources. It controls how specific devices can be accessed and which logical partitions will have that access. But it comes with no tools (apart from HCM - more of which later) to understand where conflicts may occur. We must rely almost entirely on the contents of an individual, to borrow briefly from Douglas Adams's *The Hitchhiker's Guide to the Galaxy*, carbon-based life-form's brain!

This same individual is responsible for making sure, manually, that there are no conflicts which might affect critical load balancing. The impact of a badly balanced IODF can be disastrous to the overall performance of the entire z/Platform!

They have to do all of the planning for new resources working out if there is room



to add additional devices or whether a new bank of DASD may force the purchase of new switches too. Again, all this happens without tools making it a little like playing 3D chess blindfolded!

And that's all before we start talking about just how much trouble these guys (I use the term generically to also include those few gals who work in the field too) will get into if something should go wrong. An example would be the time I accidentally connected an entire bank of Production DASD to a new Test LPAR when I forgot to make a change after cloning an existing LPAR definition!

Given the criticality of the contents of the IODF it is somewhat alarming to think that there is currently no change management software which can be used to control it! Partly as a result of this lack of tools, the IODF is often excluded from the normal business controls which affect the rest of the z/Platform.

For example, most disciplines on System z are serviced by the same Capacity Planning function. This is generally a department which predicts when more resources are going to be needed as a result of natural business growth as well as those peaks caused by special projects. IODF capacity management is normally an entirely manual process which is generally only undertaken when it is discovered that a lack of a particular resource may cause a project to fail to be implemented.

For a much more detailed technical view of the IODF I can strongly recommend reading this recent Red Book: www.redbooks.ibm.com/redbooks/pdfs/sg247804.pdf

What I want to show you with this book though is that it is vital to start making the practice of IODF management into a main-stream process. Starting with Chapter 4 there will be detailed ideas on IODF Best Practices which can be introduced to your organization.

3.7.1.1 Hardware Configuration Definition (HCD)

HCD is an interactive, end-user, ISPF application used to define, update or view I/O configurations for both hardware and software. Your Hardware Specialist uses HCD to create an IODF.

It is vital that access to this ISPF application and the accompanying IODF datasets is restricted to only those carbon-based life-forms who are tasked with maintenance of the System z environment - typically either z/OS Systems Programmers or a specialist Hardware Management team.

3.7.1.2 Hardware Configuration Manager (HCM)

The z/OS and z/VM Hardware Configuration Manager (HCM) is an optional, PC based, client/server interface to HCD that combines the logical and physical aspects of hardware configuration management into a single application interface.

The IODF contains information on logical objects and how they are defined in relation to each other. The HCM interface shows you the physical objects and their connections using a configuration diagram format, with the logical information associated with each of the physical objects presented in dialog form. For example, a device such as a DASD drive may be logically defined to a processor; however at the physical level the device may reside in a unit, in a string, which connects to a controller daisy-chained to another controller, leading to a series of crossbar switches which finally connect through a CHPID to the processor.

For everything to work the processor only needs to know the logical routing that reaches the desired device (the logical system definition). Both the Hardware administrators and Auditors need to know how the processor and each device are connected at the physical and logical level.

Any updates to your configuration made using HCM are performed via the HCM client/server link to HCD, where all the changes to the logical I/O configuration are written into the IODF fully validated and checked for syntactical accuracy and completeness by HCD, thus minimizing unplanned system outages due to incorrect definitions. As such access to the HCM application, if installed, must be restricted to the same groups as the HCD ISPF application.

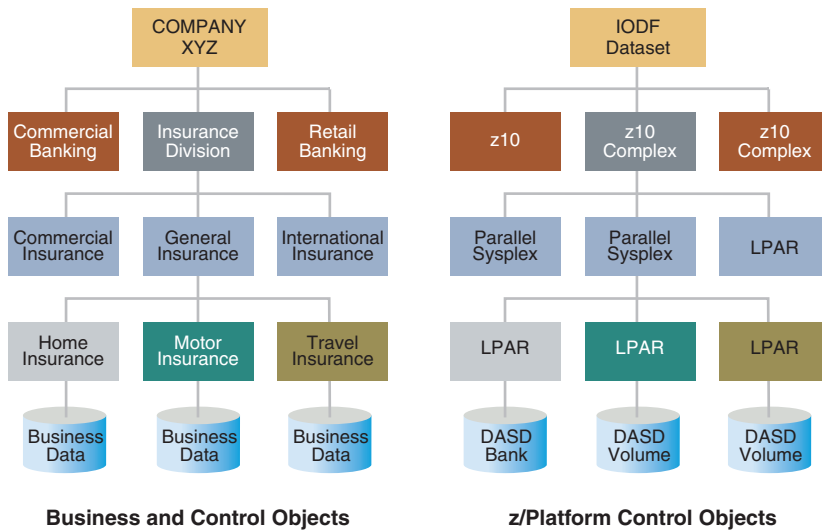
3.7.2 Mapping Business Control Objects to System z

A bit of time invested in updating the contents of your IODF now could save massively going into the future. The Business is the most critical part of any operation and it's time the fundamentals of the IT infrastructure used to back the Business became as important to the Systems Programmers and Security Analysts as it is to any other department.

Make no mistake what I am proposing will not be immediately popular amongst the folks who have run System z for decades. As a group we are somewhat uniquely opposed to change when the purpose is not clear. So my job here is to make sure that clarity of purpose can be defined and shared with the souls who will have to be involved in the changes and whose buy in is so critical.

Remember that Technical staff don't use the same language as Business staff (Sysprogs - use this section as a way to understand what your auditors are really asking you!). You'll need to make sure that your requirement is properly understood so let me give you a quick run through the basics.

The simple example of business structure that I was first taught oh so many years ago really doesn't apply to the majority of mainframe users in the 21st century. All sorts of factors have combined to make today the most complex business environment most of us have ever seen!



Just think about the changes in organizational structures that the recent credit crunch has imposed. For example the number of independent banks and financial institutions has radically contracted world-wide. Not for the most part because the business has collapsed but primarily due to Government imposed takeovers ordered while trying to stabilize a country's economy.

It's no longer good enough to just think about the business IT infrastructure as a whole for audit purposes, we must start to focus on Line of Business separation.

Let's take the diagram as an anonymous example of the new, more complex environments. Each of the Lines of Business still exists as a separate entity but is now under the overall control of The Company. Each Line of Business still has its own, complex IT infrastructure. Wouldn't you expect an audit to happen at the Line of Business level in this picture? Well that's not happening today.

What we see is an environment that is still being audited the same way as main-frame systems from 30 years ago. The Image (LPAR - equivalent to the Business Process level) is the focus of current audits. This completely ignores the entire surrounding structure and doesn't acknowledge any risk to the environment posed by exploiting back doors at the hardware/configuration level. And this is the crux of the problem I am trying to address.

You can see that Company XYZ's core business has expanded over the years to include a number of different Lines of Business (Commercial Banking, Insurance and Retail Banking). The Insurance Division has further specialty streams from take-overs in that sector. Actually, all of the Divisions have probably seen similar expansion but showing it all would make the diagram completely unreadable!

The Business Data for each sub division remains separate, of course. This is probably the main reason that methods of auditing z haven't changed much in the last few decades.



But do you know just who else can really get to your production Business Data?

The z/Environment today is significantly different to the one that we learnt how to audit all those decades ago. Whilst things like APF list and LINKLST are still important, connections between IO devices (disk, tape, networks, etc.) becoming way more prolific changes our view of what needs to be assessed. The hardware set up is managed by a small group who operate outside the confines of the ESM and who are not used to communicating with the Business. It's very easy to get it all slightly wrong without being technically wrong.

These inaccuracies are normally caused by human error but the process is the same for people trying to do deliberate harm to your organization. And there's no way to spot the difference until you've been hacked!

```

View Processor Definition
-
Processor ID . . . . . : CP2
Support level:
  XMP, 2008-E10 support

Processor type . . . . . : 2096
Processor model . . . . . : E10
Configuration mode . . . . . : LPAR

Serial number . . . . . : 0A31C42093
Description . . . . . : Used by Insurance Division
Number of Channel Subsystems : 2

Processor token . . . . . :
SNA address: Network name . . . : IBM390PS
                  CPC name . . . . : CP2

Local system name . . . . . : CP2

ENTER to continue.
    
```

An expanding number of audit bodies are now insisting on not sharing CHPIDs in the z/Environment. Identifying whether such definitions exist is going to become part of our normal processes. You can make it hard on yourself (comparing multiple SMF streams with a minimum elapsed processing time measured in weeks) or "easy" (use the IODF definitions themselves with an elapsed time measured in minutes - with the right tools).

The mapping solution that I am suggesting involves changing no more than 200 of these 32 character Description fields even in a very large installation. There can be incredible pay-off for such a small investment in terms of auditability and stability of your systems.

So, with a quick "Don't Panic", come with me on a Magical Demystifying Tour of the IODF with special emphasis on what you as an auditor need to be most interested in...

4 Processors and Partitions

Wikipedia has the following to say on Processors:

Processor may refer to:

- **Central processing unit** (CPU), an electronic circuit that can execute computer programs
- **Microprocessor**, a CPU on one chip as part of a microcomputer
- **Graphics processing unit** (GPU / VPU), a dedicated graphics rendering device for a personal computer or game console
- **Physics processing unit** (PPU), a dedicated microprocessor designed to handle the calculations of physics
- **Digital signal processor**, a specialized microprocessor designed specifically for digital signal processing
- **Network processor**, a microprocessor specifically targeted at the networking application domain
- **Front end processor**, a helper processor for communication between a host computer and other devices
- **Coprocessor**
e.g. **Floating point unit** attached to a CPU for processing floating point mathematical calculations at higher speeds than the CPU can attain.
- **Data processor**, a system that translates or converts between different data formats
- **Word processor**, a computer application used for the production of printable material
- **Audio processor**, used in studios and radio stations

As you can see, the word “processor” can be assigned many different meanings. What am I talking about here then? For the purposes of this discussion I am using the term to mean the physical hardware that IBM supply to run the System z operating systems.

System z offers a full range of connectivity options for attaching peripheral or internal devices for input and output to the server. At the other end of these connections are a variety of devices for data storage, printing, terminal I/O, and network routing, etc.

This combination of connectivity and hardware offer System z customers solutions to meet most connectivity requirements but, to make use of these features, the System z server must be properly configured.

The Input/Output (I/O) configuration is the definition of the hardware and software of the computer system, including how it is all connected. Traditionally I/O configuration has been done by Systems Programmers. Often the individual responsible is actually determined by the size and organization of the installation. A large installation may have dedicated teams to handle the I/O configuration while a smaller installation may have overlapping duties (a Systems Programmer who handles hardware configuration and storage, etc.). So, in summary the I/O configuration is usually done by:

- Systems Programmers
- Hardware Planners

System z also provides the ability to sub-divide these resources into Logical Channel Subsystems (LCSS) and then each LCSS into a Logical Partition (LPAR). When resources are used by more than one LPAR in an LCSS, the resource is said to be SHARED. If the resource is used by one or more LPARs in one or more LCSSs, the resource is said to be SPANNED. Resources that are used exclusively by a single LPAR regardless of LCSS placement are said to be DEDICATED resources. Shared and Spanned resources present special security challenges that warrant investigation and documentation. Ever mindful of these concerns, hardware designers have available ever increasing, but complex, methods of configuration management; for example with the introduction of InfiniBand CHPIDS (CIB), channel paths can be dedicated exclusively using the LSYSTEM/CSYSTEM configuration options.

Increasingly though, customers are being asked to provide for complete separation, not only within the duties of the staff but also, in the allocation and use of IT resources. System z can allow for multiple LPARs to run without the ability to affect or interact with each other in any way.

Brief technical jargon alert...

The relevance of LSYSTEM (Logical/Local System Name) became apparent with the introduction of the z10 and the new InfiniBand Channel types (CIB). A new Keyword CSYSTEM (Connected System) was introduced when coding a CIB type channel. This allows you to optionally specify which CIB Channels can be directly connected to which processors. Without using CSYSTEM any CIB Channel can be connected to any processor!

The field normally used to identify the ID Name of the partition in the IODF is the PROCID and this remains a useable field. The value of LSYSTEM can be the same as or different from PROCID. While PROCID is an 8 character field used to identify the processor which this IODF can be loaded on. When defining an XMP processor, the channel subsystem ID (CSSID) of the target processor can be appended to the processor ID. This allows for additional, more granular separation of resources by selecting what value to include in the CSYSTEM field.

Since the new CSYSTEM parameter introduces this new level of hardware control/integrity I expect that it will be more fully exploited in future releases.

4.1 Logical Channel Subsystems

Over the last decade IBM correctly predicted an enormous increase in the processing power needed by large organizations. They realized that this would mean that there would be much more data being moved between devices. This meant they had to re-think the way that these devices could be attached to the newly renamed (again) System z platform. This was IBM planning for the future and exercising the need for continuous improvement to keep pace with the business world's needs.

The amount of data held and moved around by companies, driven by explosions in online and financial markets, is now at an historic level. Multinational companies need to be able to link systems so that divisions can collate and compare market data from the entire organization. A globally distributed company needs an IT infrastructure that can cope with its needs. The following diagram illustrates the growth of mainframes in terms of historical trading models:



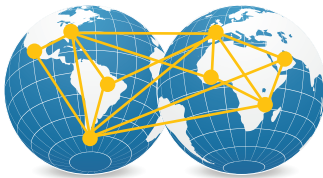
International (mid-19th to early 20th century)
Most operations are centered in the home country, with overseas sales and distribution.

15 mainframes and growing



Multinational (mid-20th century)
Creates smaller versions of itself in countries around the world and makes heavy local investments.

1000 mainframes and still growing



Globally Integrated Enterprise (21st century)
Locates operations and functions anywhere in the world based on the right cost, skills and business environment.

In excess of 10,000 mainframes and still growing

This revelation (along with the fact that almost all of IBM's large customers had pushed the mainframe hardware boundaries beyond their limits) led to the development and introduction of Logical Channel SubSystems, altering the way the channels and paths were defined to the mainframe and its operating systems. The numbers available using logical channel subsystems compared to the old methods are staggering.

Each of the (up to) 4 LCSSs can be connected to up to 15 of the available 60 LPARs in the currently available System z hardware, and each LCSS has 256 channel identifiers available. Together with the ability to share devices between LPARs this gives a level of connectivity that matches the needs of even the largest organizations. In addition each of the channel paths connected to a device can be further sub-divided by up to 3 Subchannel sets. An additional subchannel set is planned to be introduced per channel by IBM, which will allow over 260,000 unique DASD devices to be directly addressable.

The zEnterprise introduces yet another way to complicate things! One now has the ability to split each of the LCSSs into 3 logical channel subgroups enabling even more quantity and complexity in the connections to IO devices.

As the I/O structure is now configured differently, the CHPID number no longer directly corresponds to a specific hardware channel. The CHPID's can now be numbered as required. Hardware channels are now identified by their **Physical CHannel IDentifier** (PCHID - no it's not a typo, but IBM do like to confuse us with their acronyms).

4.1.1 Logical Partitions

With the changes made to the way that Logical Channel Subsystems are defined IBM, mainframes are now able to connect up to 15 Logical Partitions to each available LCSS. And each LCSS can be further split into logical channel subgroups. This represents an astounding step forward in device availability across multiple logical partitions.

Allowing the connection of 15 LPARs to each of the Channel Subsystems has expanded the flexibility of the Enterprise Class System z hardware by enabling up to 60 distinct LPARs, with each LPAR running any of the available operating

systems. Business Class System z hardware has a limit of 2 LCSSs and so a maximum of 30 LPARs.



As with any type of flexibility, this also introduces a degree of risk. Sharing devices and Channel Subsystems between LPARs opens up the possibility of access to I/O devices. Ensuring that these devices are only configured for access by LPARs that actually need them ensures that no holes are left in the security setup of the enterprise. If a device is configured to an LPAR which does not require it then access to information held on it may be unsecured, and unrecorded. This is another example of leaving a Front Door open to an organization's data.



Each of the potential 15 LPARs attached to an individual LCSS can access any of the 256 PCHIDs available to it allowing up to a massive 3,840 possible channel configurations per individual Logical Channel Subsystem.

Taking into account that there are 4 LCSSs per (EC) mainframe installation you can see how it could be possible to lose track of what devices are connected to which partitions and operating systems!

Bring z/VM into the equation and suddenly this level of complexity can expand exponentially. It is now possible to have hundreds, or even thousands of virtual machines running on a single hardware installation, with data and connected devices shared across massively multiple instances of operating systems.

4.1.1.1 Types and Descriptions

The modern mainframe is capable of being split into multiple logical partitions, which can either be linked, or run as completely autonomous systems. The method of partitioning the hardware to allow it to perform a variety of different tasks has been around for almost as long as the hardware itself, and is used to great effect by organizations around the world. I'll deal with them in detail later in the book but briefly there are 2 types of potential LPAR:

Coupling Facility LPARs – a specialized LPAR that is used to link multiple systems to each other, allowing access to each other's resources.

Operating System LPARs – used for actual data processing. This is the type that the end user will attach to. They can run any of the commercially available operating systems designed and produced for System z hardware.

There is a little used, free-format, 32-character description field available in the IODF that I have mentioned already. It can be updated using HCD and the HCM client and can be used to document channel assignments and device usage. My theory is that this could make the auditing of your System z environment a much easier task.



By using this field to its full potential one can describe which business units and which of their LPARs have access to the devices on a particular channel. This would allow the audit team to have an already documented structure to reference when performing an audit.

Many audit organizations are moving towards exclusive (i.e. non-shared) use of channels. A pre-emptive move of documenting these within the IODF would be seen as a major step forward in assisting with an audit. And more to the point, you'll be able to demonstrate exactly why (if) you do need shared channels in your parallel sysplex but also that you are on top of it - **and can prove that as per various audit standards.**

4.1.1.1.1 Coupling Facility LPARs – Risk

Much confusion has arisen from the use of loose terminology like "CF LPAR" to

distinguish a CF on a server from a standalone CF when actually all IBM CF micro-code runs within a PR/SM™ Logical Partition. This is true regardless of whether the CF processors are dedicated or shared, or whether the servers also run z/OS or are “standalone” CFs. All CFs always run within a dedicated PR/SM LPAR.

So when positioning certain CF configurations as fit for data sharing production (like a standalone model), and others for test/migration (e.g. ICF or a logical partition on a server without System-managed CF Structure Duplexing), it would be wrong to assume that this positioning is based on fundamental functional differences.

Running z/OS workloads side-by-side on the same physical server with the CF in the same Parallel Sysplex cluster introduces the potential for single points of failure for certain environments. IBM advice is against running a whole production plex on one piece of hardware. A logically stand-alone CF is acceptable for this as long as it runs on a processor which is not part of the sysplex for which it is the CF.

A coupling facility is used to allow multiple processors to connect together as a sysplex. IBM has recently announced that the Internal Coupling Facility (ICF) will be dropped at a later (unannounced) date leaving only external CFs. The statements below hold as true for an external CF as they do for a CF LPAR.

There are no specifically defined I/O devices on a Coupling Facility LPAR and it runs no application software so you would think that such a system could be defined as risk free.

However there are still issues that need to be guarded against on the Coupling Facility LPAR. As with any part of the System z infrastructure, access to the coupling facility should be secured in the same way as the rest of your enterprise, using your External Security Manager. This must include access to the structures held within it.

For those who haven't come across the system logger before, it is able to store LOGSTREAMS in 2 ways. The first is by writing the information to a DASD-only stream. And this should be secured at the dataset level by the ESM.

The other method of data storage for the system logger is to keep the information for an LPAR, an entire SYSPLEX or even a MULTIPLEX in the Coupling Facility structure. This is the area where security requirements are often overlooked.

The list of potential exploiters of system logger now includes CICS, OPERLOG and SMF data logging. The protocol used is much more efficient than the previous (write to datasets) methods, which have led to lost audit data during very busy processing times for some users.

The External Security Manager requirements to protect these log streams are completely new and if not activated there is no security applied at all. Worse this also means that the integrity of the data held in these streams has been lost.

One example I'd like to share involved a customer's Systems Programmers starting to use the additional functionality available with the System Logger for SMF. Let me state up front that this is a great idea - almost no down sides - as long as the external security team is involved.

The Systems Programmers implemented SMF Log Streams during the upgrade to z/OS 1.11. Unfortunately, the RACF team had not activated the LOGSTRM class and so there was no security on the audit data. Worse still, the housekeeping suite was not updated to reflect the new location of the SMF data. The situation



was spotted by an eagle eyed RACF consultant some days after it happened when the audit reports started “looking a bit odd”.

My advice (to my customer and to you gentle reader) is to **immediately** activate the security processing for log streams. For example in RACF the new class for logstream profiles is LOGSTRM. This needs to be activated and a catch-all profile defined with no access for anyone.

This will force the Systems Programming team to contact security to enable the new functionality they are trying to implement. All of which gives the Security Team the opportunity to manage new functionality proactively instead of running to stay still. And that also gives us some chance of making sure audit data integrity, in both its content and scope, is maintained!

4.1.1.1.2 Operating System LPARs – Risk

An operating system LPAR is one which has been configured to run one of the eligible System z operating systems, including all devices and resources required to run.

With this in mind most of the risks involved with the use of these LPARs is directly linked to how the operating system and external security manager are configured. A badly configured operating system is likely to be both unreliable and insecure, leaving the systems open to abuse.

Access to all resources available to these LPARs should be controlled, ensuring that no access that steps outside of the level required for the requestor to successfully execute their role is granted.



There is however an additional risk for these operating system LPARs and that is ensuring that the correct devices are connected to them via the IODF. Without securing access to the device definitions for the LPAR prior to IPL, there is nothing stopping the system and it's applications being loaded from an invalid disk, or accessing incorrect data.

Connection of devices in this manner could open Front Door access to the systems, allowing undetected intrusions in to the system, and the potential for denial of service.

4.1.1.1.3 LPARs

Logical partitions are available for running operating systems, or internal coupling facilities. System z can use LPARs for multiple levels of virtualization.

In the first level a single machine can be divided into as many as 60 logical partitions (LPARs), with each LPAR capable of running as an individual machine running a separate operating system.

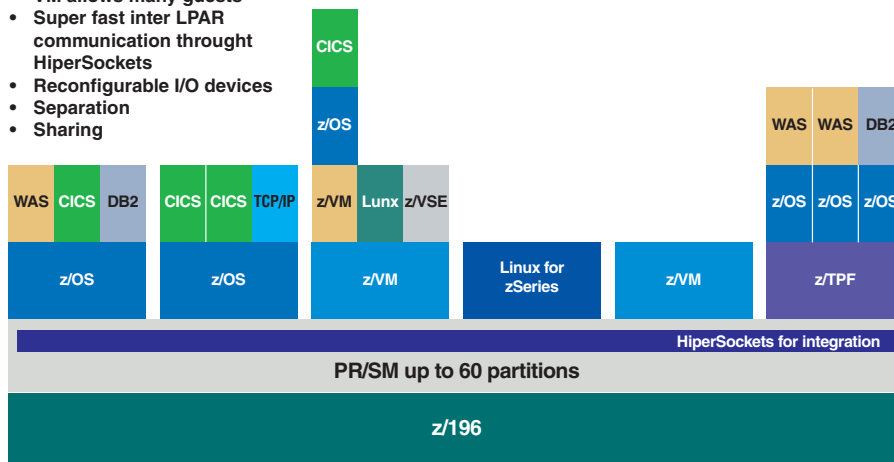
LPARs use a feature called PR/SM which can alter available resources dependent on the business requirements in real time.

This means, the resources accessible by individual operating systems (including virtualized ones) can be altered on the fly.

Access to these resources should be secured correctly. Failure to do so can lead to the exposure, whether accidental or otherwise, of potentially sensitive resources and data.

Virtualization across a single z196:

- Up to 60 basic LPARs
- VM allows many guests
- Super fast inter LPAR communication through HiperSockets
- Reconfigurable I/O devices
- Separation
- Sharing

**4.1.1.1.3.1 z/OS LPARs**

Still the most common use for the mainframe architecture, a correctly configured and secured z/OS installation is one of the most reliable (and secure) methodologies for data storage and processing. z/OS and its predecessors have been number crunching the world's data since their inception and many large financial institutions and governments rely on this ability to ensure the smooth running of their organizations.

When it comes to securing resources under z/OS the default settings will not be enough. It is also worth mentioning that default security settings are dependent on your choice of External Security Manager. RACF as standard (without PROTECTALL) has a default setting of allowing access to data, whereas with CA's offerings no access is allowed straight out of the box.

z/OS can be a complicated beast even in the best and most secured of environments. Some of these systems have been around for nearly 50 years, and many of the old bad habits of configuration to make things easy to run can still be found. The problem is that z/OS applications and systems can be asked to do much the same tasks well beyond the hardware life cycle, so the base level of some operations have not been changed, or even been looked at in a very long time!

4.1.1.1.3.2 UNIX Systems Services focused LPARs

UNIX Systems Services aka USS comes bundled with z/OS. It is a UNIX implementation that has been tailored for running on a mainframe and was the first UNIX code that could not fully trace its roots back to the original AT&T source.

Running under z/OS it introduced methods of data access that integrate it with z/OS allowing access to data through TSO and ISPF and batch functions. There is also a UNIX shell (OMVS) for data access so it is imperative that all data access methods are sufficiently secured.

Under USS, applications from other UNIX platforms are allowed to run on the IBM mainframe with very few changes required. Often the source only has to be recompiled for it to function at a base UNIX level, but some code work may be required for the z/OS integration.

Remember that access between z/OS services and USS functionality is a two-way-street so z/OS data may also be accessible from the UNIX shell, or to applications running under USS. Not to mention any UNIX savvy hacker who gets that far!





What would be considered as a Back Door on one operating system, for example z/OS, could represent a correctly configured and secured Front Door under USS. Access to the HFS/zFS file structure must be secured on all systems that have access to data stored under USS.

2 file systems are available to USS, the older being HFS (Hierarchical File System), and the now preferred zFS (zSeries File System), both of which provide full support for long filenames. It can be secured using the standard UNIX methods which some mainframe security professionals may not be familiar with, as well as via the ESM for z/OS access. zFS is the z/OS UNIX strategic file system and IBM recommends that, after z/OS 1.7, customers should migrate all HFS to zFS.

USS, together with WebSphere and CICS Transaction Gateway (to mention just 2), has helped to open the mainframe world up to the option of online services. The inclusion of Java support with UNIX Systems Services provides the scripting option for these services. The z/OS JVM (Java Virtual Machine) regularly wins “fastest” awards because of its incredible optimization for running on System z and use of the specialty zIIP engines.

Really the only downside is the additional work that must be done by the external security manager admin team to protect Java scripting from misuse. Many organizations have not yet adapted to this brave new world and that is where the real risk lies.

Within the IBM Health Checker for z/OS functionality, there is a USS health checker to aid in USS integrity and security. This extra function (USS_PARMLIB) can be used to monitor current UNIX use and compare that against the configuration definitions, reporting and dynamic changes that may have occurred post-initialization.

4.1.1.1.3.3 Non z/OS LPARs

The mainframe is not just limited to running z/OS (and hasn't been since they first appeared). The choices available are increasing as IBM angles itself towards the greener, more efficient datacenter.

Linux for System z is not emulated on a mainframe. It runs as a complete native operating system, in its own LPAR. It runs using mainframe processor instructions at full operating speed. Should your installation wish, you could run a single LPAR on a zEnterprise just running Linux for System z. This would be insanity though not to mention totally uneconomical. IBM System z servers can run mixed workloads, including numerous other operating systems, through the use of virtualization.

Most Linux on System z users run their systems under z/VM, which virtualizes the processors and memory, together with disk storage, networking and other resources. z/VM runs in its own LPAR as an operating system. Linux can be run under z/VM virtually in 2 ways; the first is based on hardware, the second on a hypervisor with hardware assistance. z/VM can be run in a nested configuration; however this introduces an extra overhead, so running z/VM as a guest is primarily used for testing purposes. Securing z/VM in a nested configuration can lead to confusion over exactly what level of the configuration is being secured by what.

Mantissa Corporation in the USA has recently announced that they hope to bring to market an x86 instruction emulation package to run under z/VM.

This could lead to the ability to run and virtualize the traditional, desk top and file server based technologies such as Windows and Linux for x86 platforms, hosting many hundreds of systems, and able to roll out new systems with incredible ease.

But that is for the future. z/VSE is still alive and kicking. And this is an OS that

can trace its roots back to 1965 - it is still affectionately referred to as the baby of z operating systems. Any operating system that is now in its fifth decade is likely to have dragged some baggage with it along the way.

4.2 Workloads

System z offers a unique opportunity in the many different types of workloads it can process. So fundamental is this to the platform now that the method of calculating throughput on z has changed from MIPS (**M**illions of **I**nstructions **P**er **S**econd - a pure hardware metric) to LSPR (**L**arge **S**ystem **P**erformance **R**eference - a weighted calculation dependent on the type of work being performed).

But this statement shows just how divorced from the business most IT issues appear to be. The business simply doesn't care how many millions of instructions can be processed. It's just not how they measure success.

Most businesses measure their success in financial terms. The Profit and Loss account is generally king. And revenue is measured in units (that are specific to each individual organization). This revenue generation metric is often hidden from IT (who are seen as a cost of doing business rather than a revenue generating object).

I worked for a large Japanese automobile manufacturer in the early 90s and they took an unusual approach to making sure every employee understood what was actually important to the business.

Each new employee had to spend 2 weeks in "Gemba" training whatever position they were taking in the organization. Gemba is a Japanese word which I understand means "in place" (but I always think of as "total immersion!") and saw every single employee doing 2 weeks on the automobile production line before moving to the post they applied for.

The unit of revenue was seen as a single automobile rolled off of the production line. And no employee was ever in doubt of what was the most important part of the business. If the production lines weren't running there was a very simple way to calculate lost potential revenue.

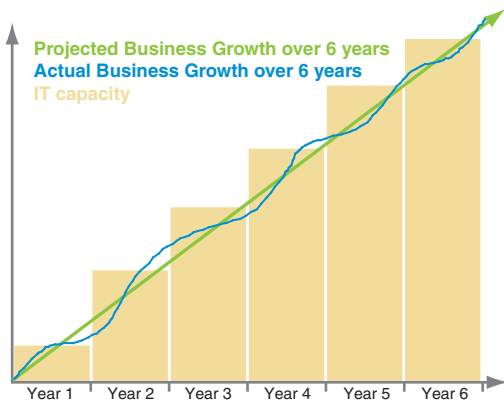
In this environment every single employee was always looking out for ways to improve efficiency to increase the number of units produced daily. Every department had a real-time digital display showing how many units had been produced so far that day and what the target was. It served as a constant reminder of why we were there - IT and all the other departments were only necessary to support the production of new automobiles!

We all saw it as part of our job to be alert to any potential bottlenecks whether they were a lack of capacity in a particular area of IT or a faster/more accurate assembly line robot being available. And whilst it may not appear so clear cut in your organization, rest assured that somewhere there is a Profit and Loss account and people who care about lost potential. Anticipating bottlenecks to revenue potential is a big deal.

The introduction of the Unified Resource Manager and its workload management functions allows z/OS, USS and zBX server workloads to be managed together as a single unit. These ensembles of platforms can then be managed based on performance metrics that are in turn based on business requirements with regard to system performance.

4.2.1 Revenue Generating Objects

This type of workload represents the core business of an organization i.e. the way that you actually make money; as opposed to non-revenue generating workload that must be viewed as the acceptable overhead of running any IT system. For example taking backups of data makes no one any money but it is vital to the continued running of the systems. Some kind of “charge back” processing for resources consumed is normally performed against these workloads.



As such it is vital that requirements for increased capacity are anticipated by IT. The business will have its own ideas about how it is set to grow and IT must provide sufficient resources not to be seen as a bottleneck to actual revenue.

Most large organizations have a department devoted to Capacity Planning. These are the unsung guys and gals who calculate what fixes are going to be needed to mitigate future IT bottlenecks. And these folks are the closest to the business that IT tends to get.

The IODF can be looked on as the blueprint of System z because it defines the hardware capacity that is installed and how much of it is in use. But the usage statistics rarely make it out to the Capacity Planning team. Most organizations are currently missing a big chunk of data

when it comes to planning for the future capacity of the System z environment.

4.2.1.1 Business Control Objects

These are the elements which relate directly to specific business controls. For example everything that needs to be done to maintain compliance with SoX or other audit regulations. You can look at this type of business control object as being where IT helps to enforce the policies of the organization.

This type of workload is not necessarily revenue generating (as in the example above) but it can be representative of core business. Another example would be the implementation of the organization’s password policies. Yet another is the frequency with which internal audits are to be conducted.

Implementation of so-called “Detective Controls” (i.e. those controls that detect activity on the system) can help to mitigate the identified risks.

4.2.1.2 Shared Objects

Some elements that are vital to the system’s integrity cannot be split into either the Revenue Generating or Business Control categories. These objects however can be just as important to the business as any of the others.

One example of a resource that cannot be categorized in this way is something that is one of the most important functions performed on the system. Every day backups are taken of system and business data, and it makes more sense to do these backups at the same time, when the data is known to be valid.

But how do you specify if these functions are for revenue generating or business control purposes; you can’t. However without backups any loss of data could lead to the inability to generate revenue, so that makes them revenue saving. SoX also states that proper backups must be taken for audit reasons so they are also a Business Control Object. This leads to the need for some objects to be shared between both categories.

4.2.1.3 Legacy vs Contemporary

Whether you view System z from a business perspective or a pure IT infrastructure one, legacy really isn’t a good term to describe the main core of System z workload



processing. It implies an old technology that people need to move on up from to cope with today's business requirements. System z embraces the older style workloads (like TSO and batch) and **also** incorporates such contemporary elements as web apps and Java.

A list of currently available LSPR workloads is as shown below and you can see that it's a pretty much even split between old and new types:

- OLTP-T - Traditional On-line Workload
- OLTP-W -Web-enabled On-line Work
- WASDB - WebSphere Application Server and Data Base
- CB-L - Commercial Batch Long Job Steps
- CB-S - Commercial Batch Short Job Steps
- ODE-B - On Demand Environment - Batch
- CB-J - JavaBatch
- CICS/DB2 - On-line Workload
- CICS - On-line Workload
- DB2 - On-line Workload
- TSO - Online Workload
- FPC1 - Engineering/Scientific Batch Workload
- WASDB/LVm - many z/Linux guests under z/VM running WebSphere Application Server and Data Base
- CMS1 - CMS Workload used for z/VM
- PD - CMS Program Development On-line Workload
- HT - CMS High Transaction On-line Workload
- CICS - On-line Workload running as a VM/ESA V=R Guest

4.3 An LPAR's PHYSICAL Properties

The task of the Logical Partition is a simple one - to create the hardware environment that the operating system then uses. This environment is created from the devices defined to the LPAR within the IODF, such as channels and DASD units, combined with some of the physical mainframe hardware, such as storage and CPUs. Each of these hardware elements can be dedicated to a single LPAR or used by several depending on the type of element and the needs of the site.

It is important to understand that the sharing of a physical resource does not always create the potential for data exposure. Consider the mainframe CPU, in order to maximize the utilization of these expensive hardware elements it is standard practice to share them between LPARs but this is done without creating the potential for cross contamination. Whereas a Coupling Facility LPAR, be it internal or external, is designed to facilitate the controlled sharing of information between multiple LPARs when they are correctly defined to a channel's Access and Candidate list, whether that be for inclusion or exclusion. Changes to the logical mapping of these physical resources can have a significant impact on all the services that use the hardware in question.

The potential impacts can range in the levels of impact to the system

- Visible, such as loss of access to a DASD device that result in a loss of service.
- Subtle, such as adding a DASD device to an LPAR that is already in use by another unrelated LPAR that offers the potential for unmonitored access to data.



To understand the physical elements of the logical mapping that creates the LPAR you must also consider the entire logical map and its physical manifestation. Unsurprisingly this detailed view quickly becomes exponentially more complex when dealing with a multiple mainframe environment.

The development of the mainframe and the multiple operating systems that have grown alongside it over the last 40 years and the various virtualization techniques they utilize has created the potential for additional layers of virtualized hardware environments. For example z/VM can be used to host a z/VM LPAR which in turn could be hosting a Linux for system z LPAR.

This may seem a bizarre choice to make but the environment as it stands now may have been developed over many years and may represent the optimum for that site. Sadly, unnecessary virtualization layers may be the result of activities at the corporate level, such as takeovers and mergers. Many of the recent rash of mergers and acquisitions will result in the merger of IT systems that might have a combined age of 80 years or more.

4.3.1 Input/Output Control Program (IOCP)

The I/O subsystem controls channel operations, and needs to be configured correctly to ensure that data flows in the expected fashion between the CP and its storage devices. In order to create the I/O configuration information you need to run the input/output configuration program (IOCP). The IOCP reads the information from a file which has been created using either the HCD panels, or on the HMC. The IOCP program is invoked when HCD needs to write to the new IOCDS.

The IOCP runs as a batch process under the control of z/OS, and its use can be controlled in the same way as access to any other resource, using the external security manager FACILITY class. If one does not exist, somebody must respond to a console message to authorize the write process. It would be unwise to have automation respond to the ICP050D message unless it is part of a controlled service. Any user with authority to access one of the many console interfaces and to issue the response command can authorize the write process. The level of protection a site places around both the IOCP process and the IODF data will determine the integrity of this core system element.

Before the write process to the IODF occurs the IOCP validates the settings that you are attempting to write. If any problems are found it issues error messages and aborts the update. E.g. if your I/O configuration contains an unsupported function IOCP will state that it has completed with warning messages for the processor id.

IOCP can be run as a stand-alone batch process, and will complete dependent on the requirements that you specify in the PARM statement. In order to write the IODF to the support element the parameter to use is WRTCDS. There are a number of situations where IOCP will not attempt a write and these include:

- You have specified WRTCDS=NO
- Errors were encountered during processing
- The operator replies NO to message ICP050D
- The IOCDS is write protected, access to the HMC is required to reset this default protection.

The IOCP REPORT parameter can be used to produce reports on the IOCDS, including reports on the complete IOCDS, or on a partition by partition basis. IOCP will also produce reports on specific PCHIDs, channels and I/O to specific devices. These can include information such as the attached channels, Channel

Sub-Systems to which the device is connected and the device type and model.

Access to the IOCP program, especially use of the WRTCD5 parameter is something that should be limited to those responsible for making changes to hardware configuration, i.e. Systems Programmers and Hardware Planners. While any changes are validated for syntactical correctness and completeness before the IODF containing the IOCP configuration is written, no checks can be made that the alterations meet system, security or audit requirements and this could lead to problems the next time an IPL is performed. Sadly this also means sometimes what you specified in the IODF didn't mean what you thought!

If you rely on Operators to respond to messages, they must be made be aware when any changes are being made to the hardware configuration. Particularly if an ESM is not being used to protect access and ensure the correct response is given to the console message. Remember, the level of protection a site places around both the IOCP process and the IODF data will determine the integrity of this core system element.

4.3.2 Unit Control Word (UCW)

The Unit Control Word is the hardware (IOCP) side of your installation's device definition, and works hand in hand with the z/OS operating system's Unit Control Blocks on the operating system (OSCP) side to match the hardware to the operating system's defined devices. An operating system's UCB must have a corresponding Unit Control Word on the hardware side. If there is a defined UCW on the hardware side, and no UCB to match the device, no I/O can be performed as there will be no logical path defined to the device, and the device will be disabled.

Without corresponding UCW/UCB definitions, the device cannot be used by z/OS. This device mapping process, linking corresponding hardware and operating system definitions for I/O devices is one of the first tasks performed during an IPL.

The UCW represents the first level of control over which devices are available to an LPAR, and you are able to specify whether the device is shared by many LPARs or exclusively available to one. This represents a level of control over device availability that is in force before the operating system, and any ESM product is active.

UCWs can be added at any time without the need for a Power-On-Reset or IPL, and the number defined can represent the actual capacity available to deal with any potential demand, rather than the number of devices that are defined and in use. This means that you can plan for expansion of your organization, allowing additional devices, and paths to these devices to be enabled as your business grows. Where a site supports dynamic devices, additional UCB definitions can be created without the need for an IPL. This process is performed by HCD as part of the dynamic installation of a new IODF.

A physical device may be attached to a number of control units, which may in turn be defined to several channels. When the z/OS operating system issues a request for an I/O operation to a connected device, the z/OS Input/Output Supervisor (IOS) uses the IODF definitions to decide which channel and control unit to point the task at, where there is a defined UCW to match the device that is required.

When dealing with virtual, logical or physical hardware, Asset Management should be considered as an important part of not only hardware planning, but of your security and audit management. Errant entries in the IODF can lead to a shortage of paths and device addresses. These same entries offer the opportunity to attach



hardware that may not be part of the requirement for the LPAR or LCSS in question immediately opening the Front Door.

In the case of the LCSS this means that the device entries that remain in your IODF could be allocated on any of the LPARs that are using that LCSS. The need to stay on top of your IODF is becoming more important as the audit community moves closer towards not sharing devices between LPARs and business units. A well-defined and maintained IODF should be seen as the foundation on which the rest of your system is built.

4.3.3 HCD/HCM/HMC

The HCD and HCM, together with the Support Element and Hardware Management Console, are used to store and configure information about resources that are available to the processor and to each of its LPARs, together with the information required to IPL an LPAR with its guest operating system.

HCD is a z/OS base element which is accessible through ISPF. The HCD panels are used to define and configure the hardware and channel settings for a processor and its LPARs including details such as external switches, control units and even devices. This information can all be stored in a single IODF. In fact a single IODF can store the details of several mainframes. And this is what IBM advises.

HCM is a workstation based application with an intuitive GUI interface that has a client/server relationship with HCD. It offers all of the functionality of HCD, combining the logical and physical side of hardware configuration. When changes are made using HCD they are validated to ensure that they are correct, and HCM performs the same check when changes are made there. It is still possible to get it wrong, but using HCD/HCM makes it much less likely.

HCM is an optional GUI that can be used to manipulate the IODF data. The GUI can be enabled to run remotely i.e. through a web browser which is extremely useful for recovering from out-of-hours issues. Of course that introduces further security implications but that's not a subject for Part 1 of z/Enterprise Auditing Essentials.

The HMC and Support Element are external devices attached to the System z hardware, running a stripped down Linux kernel (replacing the earlier HMCs that ran OS/2). There is an application on the HMC that enables a systems administrator to configure and operate the partitions on its connected mainframe system, as well as monitor the status of the LPARs. Another major use of the HMC is to start the load process in order to IPL an LPAR, with the ability to alter load addresses and load parameters. The System Element is used more for the hardware management functions and as the repository for any of the up to 4 IODFs that can be used to POR the processor.

The HMC can also monitor the state of the physical hardware in the System z box, reporting on any changes in the state of the hardware. It also has the ability to perform Power on Resets and to power down the mainframe. Access to the functions on the HMC is usually granted to Systems Programmers for alteration of configuration information and operations staff for the IPL functionality.

In addition to physical access the HMC has an API that enables remote access to the HMC. This API has been added to permit Automated Operations software to further reduce regular direct human contact with the HMC. Whilst its use simplifies the physical access issue great care must be taken to ensure use of the API is audited, secured and managed safely.

Access to HCD and HCM should always be severely restricted, as any of the changes made can have a major impact. Treat it the same way as you would physical access to the computer room (and the HMC). I could do as much damage to a System z environment (if ordered to do so!) with HCD or HCM as I could with a chainsaw in the computer room! The very complexity of the IODF makes it unlikely anyone would notice until it was too late. So much easier to hide than hitting a mainframe with noisy power tools!



4.3.4 The Support Element (SE)

The Support Element is a notebook computer. Each System z mainframe actually comes with 2 support elements for redundancy that physically sits inside the cabinet. It stores the IOCDS and PR/SM profiles used to define LPARs on the mainframe. The hardware management consoles connect to the support element to access data required to perform an IPL of an LPAR.

It is the most direct way to access the mainframe, bypassing the HMC and any OS level security. Changes made on the support element can have dramatic effects on LPAR performance and device connections. Any changes made on the Support Element can be immediately applied to an LPAR as the support element has all of the available functionality that exists on the Hardware Management Console.

The support element has a base level of control over the definitions relating to how your System z environment can be shaped. You will notice that I have said “CAN” be shaped? Changes can be made on the support element or the HMC which may lay dormant on your system until a Power on Reset is performed. This can then affect the next IPL of an LPAR on your system, where configuration details may have been altered on the support element that affect that LPAR and it may not function as expected. This is a classic method used to build a Front Door into System z.



Strict controls over access to the support element are not only recommended by IBM, they should be part of the basic physical security of your environment due to the potential that any changes made here have to disrupt service or affect system and data availability, both authorized and not.

4.3.5 Power on Reset (POR)

Switching your mainframe off and then back on, causes a whole series of additional events that don't take place just for the IPL of an LPAR. A power on reset enables any changes made to the IODF and IOCDS to be used by the relevant LPAR. This can include the addition of devices as well as any channel definitions that are required to connect these devices.

If any dynamic changes have been made to device and connection details your Systems Programmer should also make these changes to the LOADxx member to ensure that it points to the correct IODF data. If this is not done the dynamically defined devices and channels will be lost (and have to be added again).

With the increase in use of dynamic changes to implement new hardware there has been a reduction in the need to carry out a POR on a mainframe. This in turn can increase the risk of ‘sleeper’ changes being introduced via either the HMC or SE devices although robust change procedures can help mitigate some of this risk.

4.4 Channel Path Id (CHPID)

The **CH**annel **P**ath **ID**entifier used to be the link between the actual I/O hardware installed in the IBM mainframe cabinet and any operating systems you defined. Since the mid-1990s this physical mapping is actually done with the **P**hysical

processors and partitions


CHannel IDentifier.

What this means is that changes to CHPID-PCHID mapping can affect real world resources such as in-house documentation and cable labels.

The z10 was the first IBM mainframe to have no default CHPID to physical address assignments. It is now the customer's responsibility to perform these assignments by using HCD/IOCP definitions. This is done either manually or using IBM's CHPID Mapping Tool (CMT). Changes made using the CHPID mapping tool are then imported back into one of the available IODF slots on the Support Element, and made available to the processor when a Power-On-Reset is performed.

Manual assignment is an extremely cumbersome process for larger installations. If you choose to do the assignment of CHPIDs to PCHIDs (using HCD or IOCP) manually, IBM state that it is your responsibility to distribute CHPIDs among the physical channel card ports (PCHIDs) for availability and/or performance.

```
View Device Candidate List
Command ==> _____ Row 1 of 2
                               Scroll ==> PAGE
The following partitions are allowed to have access to the device.
Device number . . . : 0000      Device type . . . : 3390A
Processor ID . . . : CP3        IBM z10 in VH
Channel Subsystem ID : 0
ENTER to continue.
Partition Name  Description
CP301          CP301
CP304          CP304
***** Bottom of data *****
```



CMT does still have limitations but it can ease the process dramatically. The greatest limitation is that it doesn't consider logical partitions, switch configurations or control unit availability characteristics. For these reasons the tool results may not be acceptable for all users. While it can be extremely helpful for many installations, it can't replace a real Systems Programmer.

The first part of the equation is for the Systems Programmer to define the CHPID-PCHID mapping. After this they must define which of the logical partitions can use the channel path (or the devices at the end of it). This is achieved by using HCD to add the CHPIDs to 2 lists - the Access List and the Candidate List.

From the IOCP point of view, the channel path candidate list includes the channel path access list. But from the HCD point of view, the channel path candidate list does not include the channel path access list. Partitions already in the access list do not appear in the candidate list.

So if you want to be able to configure a reconfigurable or shared channel path online to a logical partition, the Systems Programmer must place that logical partition in the channel path's candidate list.

A logical partition that is on a channel path's Access List can access the channel path when the logical partition is initially activated at POR. When a channel path is dedicated you specify one logical partition on the channel path access list. When a channel path is shared, you can specify more than one. And when the channel path is reconfigurable there is no need for an access list.

A logical partition that is on a channel path's Candidate List can eventually access the channel path. This occurs when the channel path is configured online to that logical partition.

Brief technical jargon alert...

The operation mode of the channel (dedicated, reconfigurable, shared or spanned) determines the need for a candidate list. Dedicated channel paths are not reconfigurable so they do not need a candidate list. If no logical partitions are specified on an access list for a shared or spanned channel path, then the systems programmer must specify logical partitions on the candidate list. Also for a shared or spanned channel path, if all the logical partitions are specified on the access

list, they are all sharing the channel path initially so a candidate list is not necessary. However if not all the logical partitions are specified on the access list, a specific partition may be able to access the channel eventually if it is in the channel path candidate list. Reconfigurable channel paths do not require an access list, just a candidate list.

When the Systems Programmer defines a device, HCD and IOCP also allow them to control logical partition access on a device level. A device might be attached to a control unit that uses a shared channel path. They can specify that only certain of the logical partitions sharing the channel path have access to the device. To limit logical partition access, the Systems Programmer should specify that they want an explicit device candidate list when they define a device. On the device candidate list, they indicate the logical partitions that share the channel path(s) that can access the device.

When your Systems Programmer does not specify a device candidate list, all the logical partitions that share the channel path to the device can access the device. For a logical partition to use the device, the logical partition would have to be in the access list or candidate list of a channel path going to the device.

Logical partitions can also be set up so that they are specifically denied access to a particular CHPID.

Sharing channel paths provides flexibility and reduced cost for an installation when defining an I/O configuration. An installation selects which I/O devices to access through shared channel paths, and can define and dynamically change which partitions have access to these channel paths and to the devices connected to them.

An integral part of the Intelligent Resource Director is dynamic channel path management. Instead of defining a fixed number of channel paths to control units, this function lets Workload Management (WLM) move channel paths through the FICON Switch from one control unit to another, in response to changes in the workload requirements. By defining a number of channel paths as "managed" they become eligible for this dynamic assignment.

Managed channels must be associated with a specific sysplex, and only images that are members of that sysplex can use them. The channel path access list and channel path candidate list cannot be specified for managed channels.

Dynamic channel path management will attempt to balance the responsiveness of all the DASD subsystems, moving dynamic channels as needed.

The Intelligent Resource Director allows an installation to group logical partitions that are resident on the same physical server, and in the same sysplex, into an "LPAR cluster". This gives WLM the ability to manage resources, both processor and DASD I/O, not just in one single image but across the entire cluster of logical partitions.

Managed channels and IRD represent vital parts of the functionality of parallel sysplex.

4.5 Switches - FICON/ESCON/InfiniBand

Fiber **CON**nectivity (FICON) is a protocol that IBM uses for I/O device connectivity over fiber optic cable. Like many advances in the System z world it grew from its predecessor when businesses needs for increased I/O could no longer be met, replacing the older, slower ESCON (**E**nterprise **S**ystems **CON**nection). The use of fiber optics rather than the old Bus/Tag and copper wire to connect devices

sped up the ability to perform I/O processes to meet the needs of modern businesses.

Whilst ESCON connections are still provided on the new zEnterprise, the ESCON Director has not been manufactured since 2004. Most businesses have moved (or are moving to) faster channel attachment protocols.

The FICON switch allows the sharing of I/O devices such as DASD and tape drives between different processors. These need to be defined in the IODF on each of these processors. The zEnterprise offers an optional, "Plug and Play" like facility for FICON attached devices.

The FICON switch sits between the FICON channel within the mainframe and the control units. It is used to manage the directing of information to the correct control units. The CUs then deal with the onward/return journey to the subsequent end device. The switch and associated Switch Control Program mean that there is effectively no delay waiting for a response from the devices before the next I/O packet can be sent.



The sharing of devices between processors, while an integral part of a global business, is an issue that the audit community is starting to flag as a potential risk. The securing of any stage in the link between the CPCs and the end devices is important. As distances between the processors and the end user increase, each step that business data passes through could be seen as a potential risk point, both from an audit and hardware point of view.

InfiniBand as a communications protocol emerged from the merging of 2 competing projects, Future I/O (developed by IBM, Compaq and HP) and NGIO (next generation IO by Sun, Intel and Microsoft). It can be used on a large variety of architectures and offers the same 'switched fabric' communications link that FICON offers, but at a higher transfer rate with a much simpler configuration model. It offers a much more reliable and efficient way of directing I/O.

Each host or target channel adapter has one or more ports, each of which maintains 2 separate queues (one for send instructions and the other for receive) allowing each port to support multiple send and receive operations at the same time. These queues contain information regarding target/host device. When a request for an I/O operation to take place is submitted it is placed in the appropriate queue, the channel adapter executes the requests in the order that they are received.

The cables connected to each port and used for the actual data movement are organized into what are called lanes, formed by either 2 fiber bi-directional connection, or a 4 fiber copper connection. You could think of these as lanes on a freeway, where the data-packets are only released when there is space for them to travel straight to their target destination, avoiding any congestion.

As with FICON, this frees up the processors and storage devices to tackle other tasks while it waits for a response regarding the read or write request, enabling great increases in overall processing efficiency.

Configuration and status information regarding the state and health of your defined channel adapters are available on the HMC, but access should be limited to Systems Programmers and Hardware Planners. The CHPID mapping tool gives the ability to assign CHPIDs across control units, and you should ensure that there is no single point of failure when defining the mapping.

4.5.1 Switch Control Program (SWCP)

Switches perform a very important task, directing data to and from devices. Ensuring

that data and responses are returned to the requesting system requires a process that has overall control over data routing through the relevant switches and control units.

The Switch Control Program performs this vital function by constantly monitoring the data traffic through the FICON switches. It is able to route data down one particular route, while allowing the responses to follow a completely different route through the switches from the device control unit to the requesting LPAR.

The importance of this task is suddenly blindingly obvious, and any failure of the Switch Control Program can have disastrous effects on data availability and response times. Such problems with the Switch Control Program could lead to a failure to be able to access data, whether it be application data, user data or that vital to the running of the operating system.

4.5.2 PORTS – Internal vs External Connections

PORTS represent the actual interface between the System z hardware and the peripheral devices needed for day-to-day operations. Making changes in this area relies on the Systems Programmers having meaningful discussions with the Hardware Installation team. Physical literally meets logical.

Some ports are used internally for connecting elements within the mainframe itself e.g. linking the processor books to the I/O processor. As long as your computer room is physically guarded against unauthorized entry then the internal type of connection doesn't really pose any risk.

The remaining ports are all used for connections to external devices e.g. DASD, tape, consoles, network, etc. Disgruntled Hardware Installations engineers could pose as much of a danger as disaffected Systems Programmers in terms of being able to build Front Doors to the system. Switching a cable can do as much damage as changing the IODF contents.

From the I/O processor cage, information is passed through further ports, using the associated PCHIDs to the device control units. These connections all require definition to both the hardware (IOCDS as well as physical implementation of kit) and the associated operating systems (OSCP).

The OSA-Express3 cards fitted to the latest zEnterprise processors can be either dual port 10 Gigabit Ethernet or 4 port 1000BASE-T Ethernet cards. This has resulted in an increase in potential LAN slots to 96 whilst also offering a more flexible choice of network interfaces.

Processing data through the internal ports (prior to passing the information to and from the control units and devices) is also handled in a much more efficient way. All of the I/O for all of the possible 60 LPARs running on a zEnterprise has to be handled in a timely manner.

As an example let's look at a typical banking environment. Numbers of transactions per day is measured in the millions. Each one of these transactions makes demands on the I/O subsystem and it's not a one-to-one relationship. The total number of I/O requests in a typical day at a large financial institution can be into the billions!

The ability to manage the IODF port mapping that is required to run System z is one of the most critical areas of configuration. This is another case where skilled carbon-based life-forms will always be required. Tools just aren't enough.

That said IBM's CHPID mapping tool does provide a relatively intuitive method of linking the internal ports within the mainframe hardware, with the ports defined to the operating systems running on your LPARs. No large site should try to run



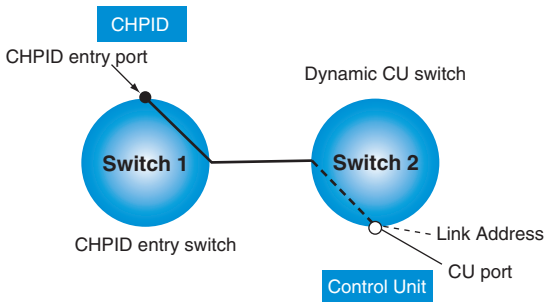
without this tool!

4.5.3 Chaining and Cascading

Early mainframes supported a relatively small number of channels, in no small part due to the physical size of the channel cable pairs and their connectors. To overcome this bottleneck the parallel channel architecture, introduced in the 1960's and potentially still in use today, allowed users to connect multiple devices in series to a single channel. This practice was known as daisy-chaining.

The ESCON channel interface, the replacement for parallel channels, is a point-to-point architecture and does not support this practice. Instead an ESCON director (aka a switch) was required, installed between the mainframe and the device. The ESCON director supports dedicated or dynamic switched connections and a maximum of 2 directors may be used in series in a process called chaining.

Configuration with 2 switches which are chained & the CU switch has a dedicated connection



Unlike daisy-chaining where it was possible to physically trace a channel path from mainframe to device, the ESCON director is a black box with lots of cables going into it. The IODF holds the details about the switch, such as entry and exit port numbers, that are needed to validate a channel path in the physical world.

FICON, the next evolution in mainframe channel architecture adds the practice of cascading to this already complicated connectivity map. The path that an ESCON channel takes through the ESCON switch network is a fixed point-to-point connection. However, switched FICON channel uses a switched fabric network topology. This means that although the FICON switch is defined in the IODF in a similar manner to an ESCON switch the physical path is not fixed. Instead the path or paths taken through the fabric are dependent on the configuration of the fabric.

As with the IODF the configuration of any fabric (hardware - typically cabling combined with intelligent switches aka network) should be protected from unauthorized access or alteration.

It is very important to note that neither ESCON nor FICON switches, nor their networks, are visible to the operating system. The IODF contains the only references to them.

4.6 I/O Devices

I/O devices are the most important part of the IT Infrastructure (alongside the network which is also a series of I/O devices) as far as the business is concerned. They allow humans to tell the computer what they need it to do. They provide stable environments for storing data as well as. They provide network connections to the outside world. Without I/O devices, even the latest zEnterprise would be just an expensive paper weight!

Ranging from consoles to DASD, and tape drives through to printers, the variety of I/O devices which can be connected to System z is almost limitless. Each individual mainframe is delivered with the drivers (aka UIMs) for hardware devices that have been ordered at the same time pre-installed. If the driver isn't installed in the z firmware then a new device type will not work. This makes the firmware for every z implementation unique!

Whilst the device type is critical to the mainframe and requires much planning it is typical for the impact not to be felt by the end users. New DASD tends to be included in Systems Managed Storage (SMS) automatic class selection (ACS)

routines and the average carbon-based life-form user of System z doesn't care in the slightest. They've been able to allocate their file or run their transaction. That's sufficient for most purposes.

The IODF allows your Systems Programmer to logically group device types with similar characteristics and assign the group a generic name. This is known as a generic device type and is stored in the Eligible Device Table (EDT).

For example z/OS groups the 3330-1, 3330-2, and 3333-1 into a supplied generic device type named 3330. Any time a program allocates a 3330, z/OS interprets it to mean any of the devices in that generic device type. This allows the end user to gain space on any eligible 3330 type device without having to deal with specifics, but in order for a device to be available to an LPAR it must be on device CANDIDATE LIST, and when the LPAR has the device assigned, on the ACCESS LIST.

An esoteric device group can include devices of more than one generic device type. When z/OS attempts to allocate a device from an esoteric device group, it turns to the device preference table that defines the order in which to attempt the allocation. The system uses the table to serialize its selection of those devices.

A poorly maintained IODF can lead to device allocation errors, where devices defined as Generic or Esoteric are unable to find a matching hardware Unit Control Word. This is sometimes known as 'Configuration Drift'.

HCD provides a predefined device preference table that defines the generic device types and lists them in order of performance. (The first device type on the list is the fastest. The last device type on the list is the slowest.) Systems Programmers can use the default device preference table values for a generic **or** specify a different value to change the allocation order.

This grouping and sharing is absolutely perfect for day-to-day use in our most complex environments. However it can lead to difficulties identifying ownership of data in respect of audit compliance. More on this last point a little later..

4.6.1 Control Unit Sets

A control Unit (CU) is typically the hardware device which sits between the System z platform and the end device. For example, DASD is generally supplied in banks with built in, or external, controllers. The controller translates the request passed through it to something that the specific device at the end will understand.

Control Units sit immediately below CHPIDs or PORTs when connected to a SWITCH in the IODF and are defined by your Systems Programmer using HCD and/or HCM. HCD also ensures that only the correct devices types can be defined below the control unit.

Systems Programmers will also use HCD and/or HCM to define by Processor (PROCID), by Logical Channel SubSystem (LCSS) and possibly by SWITCH how a defined control unit is attached. This is the point (and above) at which I would advise using the Description field to annotate the business object associated with the CU. There is little point taking the Descriptions down to the individual device level from the auditor's perspective.

4.6.2 Allow LPAR Access

I've already talked briefly about "Candidate Lists" and "Access Lists" in the IODF and now I'd like to expand on the subject a little.

When a channel path is attached to a processor in LPAR mode, you can use HCD



to specify which LPARs have access to the channel path. You specify access by including an LPAR on the candidate list, access list or both.

An LPAR that is on a CHPID's access list can access the channel path when the LPAR is initially activated at POR. An LPAR included on a CHPID's candidate list will eventually be able to access the channel path, but it will require some intervention post POR e.g. VARYing paths online to the LPAR.

Devices can be defined as SHARED where although a device might only be assigned to one (or more) partitions, it may be available to a large number of others. The partitions that a device CAN be assigned to are listed on the devices candidate list.

HCD automatically considers a logical partition in an access list to be in the candidate list so you do not need to add a logical partition in the access list into the candidate list.

The operation mode of the channel (dedicated, reconfigurable, shared or spanned) determines the need for a candidate list. Dedicated channel paths are not reconfigurable so they do not need a candidate list. If no LPARs are specified on an access list for a reconfigurable, shared or spanned channel path, then the Systems Programmer must specify LPARs on the candidate list.

Also for a shared or spanned channel path, if all the LPARs are specified on the access list, they are all sharing the channel path initially so a candidate list is not necessary. However if not all the LPARs are specified on the access list, an LPAR may be able to access the channel eventually if the LPAR is on the channel path candidate list.

When the Systems Programmer defines a device, HCD and IOCP also allow them to control LPAR access at a device level. A device might be attached to a control unit that uses a shared channel path. Systems Programmers can specify that only certain of the LPARs sharing the channel path can have access to the device.

To limit LPAR access, the Systems Programmer must specify that they want an explicit device candidate list when they define a device. On the device candidate list, they indicate the LPARs that share channel path(s) that can access the device. When a device candidate list is not specified, all the LPARs that share the channel path to the device can access the device. For a LPAR to use the device, it would have to be in the access list or candidate list of a channel path going to the device.

Sharing channel paths provides flexibility and reduced cost for any installation when defining an I/O configuration. You select which I/O devices to access through shared channel paths, and can define and dynamically change which LPARs have access to these channel paths and to the devices connected to them.

4.6.3 An LPAR's LOGICAL Properties

The advantages of splitting a high power, high capacity IT system into multiple logical views are undisputed (think "virtualization"). I've talked a lot about Logical Partitions because that's the way that System z is typically implemented (one could still run a zEnterprise as a single logical entity - i.e. not using PR/SM to split it down into multiple LPARs - but it would be an incredible waste of resources) however this process is invisible to the end user.

Much time and energy is devoted to making sure that users can reach all of the resources that they need. It's our job as Security Professionals to make sure that they can't get to more than they need.

The logical properties of an LPAR consist of things like which operating systems

can be IPLed (OSCP definitions), which consoles can be used to control the operating system both during the IPL process and during general running (NIP Consoles) and where to find all of the elements required during the IPL process (LOADxx). But they also include the elements which interface from the operating system (UCB) to the hardware (UCW) and the processes used to maintain the IODF (HCD/HCM).

Most importantly for the end user are the definitions which allow simplified access to the massive quantities (up to 260,000 individual) I/O devices which can be attached to the System z hardware (EDT).

All of these elements can pose potential risk in the System z environment and require additional consideration above simply defining profiles to protect the data under whichever External Security Manager your organization uses. In all cases the logical properties of an LPAR can be affected outside of the normal running of the operating system. In some cases the only additional security which can be applied is pointing a CCTV camera at the element in question!



4.6.3.1 Operating System Control Program (OSCP)

Ask Systems Programmers who have been in this business for a **long** time and they'll tell you that the OSCP (used to be known as the MVSCP) is the way we used to define the things needed by the operating system at IPL time (to build all the necessary control blocks, etc.). Well, it still exists even though the process to create it is now part and parcel of generating an IODF, not a separate entity.

An IODF can contain more than one OSCP and your Systems Programmer instructs the LPAR which to use at IPL time. HCD can provide a list of defined OSCPS within a particular IODF showing which systems could potentially be IPLed.

IBM recommends that the Operating System Configuration is the first thing that your Systems Programmer defines in the IODF. It doesn't take long because there are only 3 parameters initially and one of those is the Description field. The remaining 2 are the OS configuration id and the Operating System Type variables.

The Nucleus Initialization Procedure or, NIP, is used to "boot" z/OS. During the process, control of the system resides in a special class of Console called a NIP Console. Your Systems Programmer defines the NIP Consoles to z/OS using the OSCP component of the IODF.

It's a simple(ish) update with only 3 parameters which sit beneath the OS configuration id. The first parameter defines the search order that the operating system will use to try to find a NIP console at IPL time (1 = first, etc.). The second parameter is the device address of the console to be used so this can't be done until after console devices have been defined to the IODF. And finally there's the device type identifier which must match with the device type in the IODF definition for the device.

4.6.3.2 NIP Consoles – Security Considerations

Since the External Security Manager is not up until after z/OS is fully initialized, little can be done to monitor and/or prevent unwanted activities from taking place on these consoles. Operator training and strong physical controls are the primary options.

Never define a NIP console that is housed in a public area!

4.6.3.3 Esoteric Devices – Allocation Concerns

There can be hundreds of thousands of individually defined I/O devices in any IODF. It would be impossible for the humble user to keep track of every single



processors and partitions

byte of data and so methods of easing this predicament have evolved. Nor do your users want to have to resubmit their work simply because the single tape drive they specified is currently in use by another user.

The first thing that was done by IBM was to allow logical grouping of like I/O devices. A user will submit a job that specifies an esoteric device group for a file they are going to allocate. That job will request z/OS to allocate a device from that group.

All of the devices that your Systems Programmer assigns to an esoteric device group must be of the same device class with one exception - you can define esoteric device groups that contain devices from both DASD and tape device classes but this is not recommended by IBM.

Devices belong to one of the following classes:

- Channel-to-channel adapters
- Communication devices
- Direct access devices
- Display devices
- Character readers
- Tape devices
- Unit record devices



The functionality that is provided by the use of esoteric devices is critical to the successful implementation of a parallel sysplex of any kind. However, strict control should be maintained on who can change the contents of the eligible device table. Accidents do happen but problems in this area are more likely to be the result of deliberate sabotage.

Imagine what could happen if someone changed the selection order or added new devices to the front of the EDT in the communication devices list. You could be sending classified information out to a completely wrong destination.

Even an accidental change to a direct access device selection order could result in production data being sent to the wrong disks. The resultant mess can become very hard to audit. This means that it is vitally important to monitor any changes that are made to Esoteric Device definitions, as discovery of where the change has been made after the fact would be very time consuming.

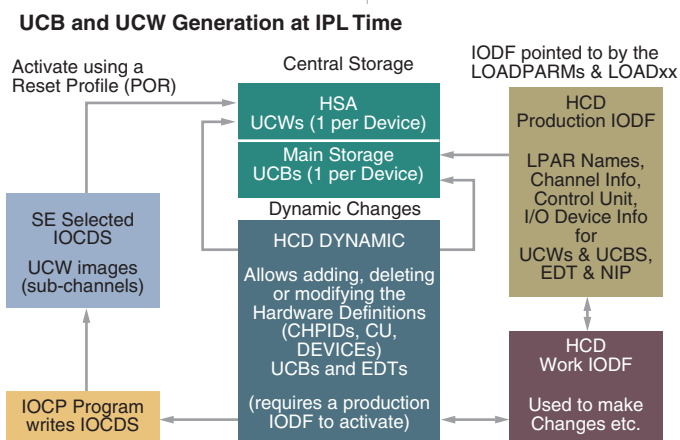
4.6.3.4 Unit Control Block (UCB)

A unit control block is a section of memory that describes a device used for I/O

to the operating system. They were introduced in the 1960s with the OS/360 operating system as a way of saving expensive memory space, flagging a device as busy when I/O is actually being performed.

UCBs are built during the IPL process by the Nucleus Initialization Program using information from the IODF. Data stored in the UCB can include such things as the device type, its address, CHPID, sub-channel identifier and volume serial number. Much of the information stored is dependent on the particular device type.

Planning your mainframe hardware installation and its associated channel definitions would be almost



impossible without understanding the link between operating system resources (UCB) and actual devices (UCW). It is one of the primary reasons that System z specialists will always be needed whatever tools are available!

The UCB is used as part of the method of controlling access to the device, and there used to be a limit of a single I/O to one UCB at a time. Nowadays Parallel Access Volumes allow the allocation of more than one UCB to a logical device using an alias address for the device. This allows more than one I/O to a device at a time, increasing response times and improving data availability.

Writing to the same contiguous part of a file is still serialized, but access to other data areas, whether for read or write processing, can continue to be processed without having to wait for the write to complete.

IBM first introduced the ability to move alias UCBs from one device to another with Work Load Manager. Design flaws meant that the move was often not processed quickly enough to improve response times. The advent of Hyper Parallel Access Volumes solved this problem by assigning a pooled UCB only for the duration of the I/O operation, releasing it immediately when processing had completed, avoiding the delay with Work Load Manager. This means that a smaller number of UCBs are required to service the workload for a device.

When defining a device's UCB you can decide whether it is online immediately or not, and whether there are alternative channels available for communication with it. A lot of these settings are dependent on the specific device type being defined.

UCBs are no longer fixed to a specific device, but will contain details for the device they are currently being used to address. And a UCB **must** have a corresponding UCW so that the operating system can match its device definitions to the hardware.

4.6.3.5 HCD/HCM

HCD and the GUI counterpart HCM are used in conjunction with one another to map the physical hardware with the logical definitions contained in the IODF. While some of the functionality overlaps there are differences. HCD owns the data held in the IODF (i.e. it is the only process allowed to update the IODF) which contains all of the logical definitions. HCM also displays the physical objects and their connections.

Both the HCD and HCM can access and update the IODF, however only one IODF dataset at a time is available to both HCD/HCM and the IPL process. From a hardware perspective, one or more IODFs can be loaded into the one of the 4 available system element slots. One of these slots will be detected when a Power on Reset is performed and this will be used for the processors LCSS, LPAR, CHPID and device UCW configuration information.

An LPAR only needs to know the logical path required in order to access the end device. However the physical path required for access to this device also has to be defined on the HMC. This will include all switches and control units that the data must pass through to get to the end device.

As HCM will usually configure the logical and physical routes at the same time, (with the crossover in functionality where HCM can also be used to build an IODF) the logical and physical definitions will usually correspond with one another. If there is a mismatch a warning message will be displayed.

The HCM configuration diagram is created during the process of mapping and this shows the connection between physical devices and their logical definitions.



The IODF and configuration file need to be exported from HCD for input into the CHPID mapping tool. This allows the logical definitions for channels and paths to be matched up with related switches and their ports. Once these have been mapped the IOCP is loaded back to the host and the CHPID to PCHID logical assignments are refreshed.

Access to the HCD and HCM should be restricted to Systems Programmers and your hardware team as any changes to the logical mapping of your configuration can have an effect on the security and availability of systems.

4.6.3.6 LOADxx Configuration Member

Considering how vital the LOAD member is to getting the system enabled and running, its contents are actually surprisingly small. However the importance of these members and the impact of attempting to IPL an LPAR using the incorrect member can be enormous.

The ability to alter the physical location (which DASD device the dataset containing the LOAD member is on) and which LOADxx member entry contains the correct parameters to IPL the LPAR as required is as simple as altering a character or 2 when performing the IPL.

Which LOADxx member is to be used (and which disk the library that contains it resides on) is passed as load parameters. A full explanation of these can be found in the next chapter.

The load unit address specifies the physical address of the device on which the library containing the LOAD member is stored. By default the system will search for the LOAD member in the following locations and order on the IODF volume:

1. SYSn.IPLPARM (where n = 0 through 9)
2. SYS1.PARMLIB
3. SYS1.PARMLIB on the SYSRES volume

If the relevant LOADxx member is not located in any of the above locations the LPAR will enter a wait state.

By specifying the physical unit address to look for the LOAD member, the dataset does not need to be cataloged, so it may not necessarily be the same member that you think you are looking at when investigating any issues occurring during an IPL.

4.6.3.7 Initial Program Load (IPL)

The IPL process is phased with each particular step gathering and using information to perform its tasks before moving on to the next phase. Each of these phases of the process requires information to be passed between the processor and attached devices, whether they be a DASD device where all of the system parameters are held, or the actual passing of the initial instruction to perform a load as it is passed from the HMC or Support Element to the processor.

Each time a stage is completed the next set of parameters is gathered from system libraries in order to bring the LPAR up to the next step of readiness, and each of these transition stages offers an opportunity for incorrect parameters to be used. This could lead to an unstable, unsecured system being available to employees, customers or other members of the public.

Remember that all of these settings that are passed to IPL an LPAR are based on just 12 parameters entered on the HMC. How often have you seen, or heard of, a production LPAR being IPLed with incorrect parms? It happens. Most of us will have known it to happen at some point (some of us may even have caused it®).

Mistakes happen.

During an IPL the NIP (**N**ucleus **I**nitialization **P**rogram) pulls all of the necessary information from the IODF required to begin the process of loading the operating system onto an LPAR. This will include all of the hardware definitions, together with paths and channels required to access devices.

The first step of the IPL process sets the device ID where the IPL information will be gathered from. When this information is pulled from DASD it is gathered from the physical location on the disk Cyl 0 Trk 0 Record 1.

The actual load parameters are read from the LOADxx member found in either SYSx.IPLPARM or SYSx.PARMLIB, where xx is the load parameter passed when the IPL instruction was issued on the HMC or Support Element.

The LOADxx member will also point to the datasets containing other system parameters, the location of the master catalog and other device definitions required for the system to be loaded. It should however be noted that the operator can override many of the pre-defined settings by altering the Initial message suppression load parameter when triggering the IPL.

This can include the locations of parameters regarding system datasets and settings, together with security settings regarding access to resources. These system settings can be altered so that applications and tools do not use external security.

The IPL process for z/OS, in other words, the software which can affect hardware configuration, is covered in detail in the next chapter.



5 IPLing a z/OS Logical Partition

IPLing (aka **I**nitial **P**rogram **L**oad) is the first step in making z/OS available to end users. It is akin to the PC “boot” process and is the part of the equation that finally links hardware resources with the logical routes through to them. The result of a successful IPL is that the operating system will sit waiting for input. This is the point where the operators (or preferably automation) start the actual business applications running.

From a technical stand-point the whole process is separated into different steps - none of which is any use on its own. But each step builds upon the last eventually providing the processing environment expected of z/OS. Let me take you on a simplified tour of what happens when you IPL a z/OS LPAR...

First of the steps is the hardware IPL. Everything starts with a single mouse click from the Hardware Management Console (HMC). The process followed is defined by the System z hardware and controlled by that same hardware. There is no security inherent in this part of the process. The HMC should be monitored by CCTV.

A hardware system reset is issued before the IPL process can begin so anything that had been running in that LPAR will die catastrophically! Unsurprisingly it is not considered to be good practice to ‘IPL over the top’ of a running system.

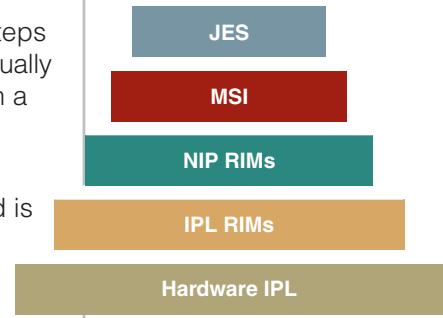
The second step is the **R**esource **I**nitialization **M**odules (RIM). Originally this process just loaded the Nucleus and set up a Master Address Space environment. Now-a-days it is much more complex because of the requirements of dynamic I/O. The process is still single threaded though and this means that nothing else can take place on the LPAR during the IPL.

Next up is the **N**ucleus **I**nitialization **P**rogram (NIP) which initializes the basic systems resources. Just very basic z/OS system services are available as NIP begins but more are enabled as the process progresses. This is where the control blocks are created for the various I/O processes e.g. UCW/UCB and DASD pathing. The NIP Console defined in the IODF is in use now. Both SYS1.NUCLEUS and LNKST are opened at this time and WTO messages are starting to be written to SYSLOG. There is still no external security manager environment so physical controls are needed to mitigate risks.

NIP terminals/consolas are defined in the OSCP element of the IODF. Like all devices they must have corresponding (and matching) UCB/UCW entries before they can become functional. It is often the case that as changes are made to the hardware configuration (IOCP), for example to remove a device that is defined with NIP status, the corresponding entry in the OSCP is overlooked and not removed. When the IODF update is completed it will validate and no error or warning will be displayed.

The device in question may have been removed from the operations area and all is thought to be well, but this may not be the case. The device address in question may well be re-used at a later date in an area where physical security is not considered as important, for example in a public area. The device will appear to function as normal for staff and public in that area UNTIL THE NEXT IPL. There will then be a fully functional NIP console available that is outside of the control of operations staff (and possibly without their knowledge), leaving changes to system configuration settings open to alteration. Not a good thing!

The z/OS element in control at this time is the **B**ase **C**ontrol **P**rogram (BCP). BCP



provides the essential operating system services of z/OS. It includes the I/O Control Program, the z/OS UNIX System Services Kernel and (from z/OS 1.8) the z/OS XML System Services.

The penultimate step is **Master Scheduler Initialization (MSI)**. MSI completes the initialization of all of the remaining system resources and passes information to the **Job Entry Subsystem (JES2 or JES3)**. All of the system consoles are now available and the external security environment is ready to go by the end of the process. It's the last step of the actual IPL but only a limited subset of systems applications, such as automation, are available until JES is up.

JES initialization is considered the last or top step of the IPL process because it makes running actual work on the system possible. In reality, this flows seamlessly from the MSI step.

Information about the required shape of the JES subsystem (e.g. whether JES2 or JES3 is in use, which procedure libraries may be used and which initiators should be able to run what type of work) is gathered from various locations.



You must remember that this is a **very** simplified overview. Some of the steps can contain hundreds of individual elements! The important point to take away from this though is that there is absolutely no external security manager available for the majority of the IPL process. For the first 3 steps there isn't even any recording of what has taken place. It is absolutely critical that physical controls are in place to prevent misuse.

5.1 Required System Elements

So we've established that there's no external security manager available for most of the IPL process. What may not yet be obvious is that the required system elements used during the IPL process are the same ones that allow changes to the system configuration to be made. These activities must legitimately take place on System z but what controls are required and more importantly how can you mitigate the problem without ham-stringing the Systems Programmers tasked with keeping the environment running?



Whilst protection of the z/OS configuration envelope is critical, over stringent external security implementation can be as bad for a site as lack of definition. Amongst other things, it's bad for the morale of those un-trustable but vital carbon-based life-forms who **have** to deal with it to do their jobs. What is important is being able to figure out when it changed, who changed it and why it was changed in a provable manner, preferably before the change bites.



One example that leaves no end of Front Doors open for abuse is the Authorized Program Facility (APF) Table, a component of system memory. It is populated with the fully qualified names of the APF datasets during NIP processing. At early points of the IPL the master catalog has not yet been loaded, let alone the ESM. This means that any entries specified in the prevailing PROGxx PARMLIB members will be loaded into the table without the NIP process having any security checking as to the validity of a dataset - does it exist, does it reside on the volume specified, is it SMF managed and does it have, for example, a UACC(READ) like ESM profile?

None of these questions can be answered during NIP processing. All of which results in the APF Authorization being subject to question and abuse at the end of the IPL. The best, real world, example of what happens next is that someone comes along and allocates a dataset that did not exist during NIP but was within the defined set of datasets found during the PROGxx process cycle, that dataset is now fully APF authorized. Not cool, someone is in trouble!

The following sections will deal with what damage can be done from specific elements and, where possible, what to do in mitigation.

5.1.1 Hardware Management Console (HMC)

The hardware management console (known by the hardware folks as the Support Element - a Think Pad running Linux which is stored inside the System z cage) can be used to start the IPL process for an LPAR. The HMC provides access to details for all of the LPARs that are either active or defined in the IODF.

It is vital to know that anyone with access to the HMC can change the LOAD address and load parameters used to perform an IPL on a system (it does not **need** to be followed by an immediate IPL). This forms part of normal, authorized system maintenance to bring any changes into the definition for an LPAR. This level of control is available to all of the LPARs that are defined to the system. It doesn't matter whether they are production, test or coupling facility LPARs.

All that is required to perform an IPL (including a catastrophic load on a system that is already up and running!) is to select the LPAR in question and click "LOAD". The load address and load parameters can be changed at this point too.

Often new LPARs are created by cloning an existing environment in the Plex. This can mean that the IODF and PARMLIB from your existing production environment could be used elsewhere simply by altering the parameters used on the HMC to perform the LOAD. If an OSCP for one of the alternative System z operating systems has been defined in the IODF, any one of these could be loaded on the LPAR!

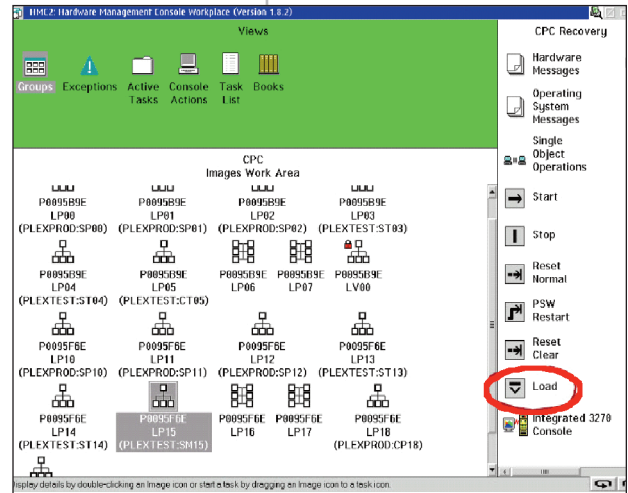
IBM's practice is to deliver the z10 with all of its LPARs configured and available. This means that a new LPAR could be activated and loaded by someone with the knowledge and skills to gain access to your system, allowing another potential Front Door. Load parameters used to IPL this LPAR could be from un-cataloged and un-audited configuration datasets. The possibilities could be catastrophic, ranging from full denial of service to exposure of confidential customer and company information with no way of tracking it!

The IPL unit address is required by the operators if there is a failure of an LPAR. Often it will be written on a white board in the computer room. By altering this address to an entry that is not an IPLable volume the load will fail. And the operators won't know why!

Use the correct volume address, but alter the LOAD parameter to allow IPL messages to be given a response and the whole shape of the system can be changed. Equally damaging to an organization, an LPAR may appear to load correctly but sensitive company data could be made available to unauthorized persons! Or for an organization that doesn't have automation correctly deployed, an LPAR could be active with connections to data even though it is under the control of a hacker.

The list of potential problems that can be caused by altering details on the HMC, and the levels of impact that these can have is wide and varied. It is vital that access to the Hardware Management console should be strictly limited to authorized personnel, and that any changes are fully documented.

The damage that can be done to an organization by simply altering the parameters



used to load an LPAR is severe. Due to the resilience and reliability of z/OS and System z as a whole, changes made might not be noticed until the next IPL (or even POR) is attempted. A regular check should be made to determine whether unauthorized changes have been made to the IPL parameters for all of the LPARs.

5.1.2 IPL Unit Address

Specified on the HMC this value points the IPL process to the disk containing the necessary information needed to load an operating system into an active LPAR (IPL Text, IODF, SYSn.IPLPARM, etc.). It is normal operating practice for Systems Programmers to create a duplicate IPL volume in order to apply maintenance to z/OS. At the next IPL the new volume will be used bringing all of the required changes in at once.



However, this would also be one of the ways to build a Front Door into System z. Often the IPL Unit Address is written on a white-board in the computer room. It is a required parameter if there is a catastrophic failure of an LPAR in order to facilitate restart after all. The white board is used to check the values specified match what the Systems Programmers are expecting.

Were I an unscrupulous individual looking to do some damage to an organization's reputation (and had spent some time on site) I could create one of these duplicate IPL volumes. A minor distraction in the computer room could give me the opportunity to change the IPL Unit Address on the HMC and alter the white board to make sure that my duplicate volume was picked up at the next IPL!

The only way to secure what is being typed into the HMC, and by whom, is to point a CCTV camera at the device. But there is more that you can do to provide an audit trail. For example anyone using the utility that is needed to initialize a new disk (ICKDSF) should have their activity closely monitored. In fact it is not inappropriate to have the program specifically controlled and all access to it (authorized or not) audited.

The aim here is not to prevent the Systems Programmers from performing perfectly valid maintenance tasks. It is to offer some opportunity to find out what happened if you are unlucky enough to face a situation like the one I have described above.

There is one last security consideration surrounding the IPL Unit Address and that is exploiters of the HMC API (for example automation). The API can be used to control IPLs remotely. Whilst this can be a legitimate requirement the process should be subjected to stringent controls.

5.1.3 LOAD Parm Decoded

The Load parameter is entered along with the IPL unit address when the IPL process is started on the system. It provides the information (or paths to it) needed to perform an IPL on a Logical Partition. The IPL process can be started from the HMC, the remote feature of the HCM, or the Support Element, with each of these systems being able to alter these parameters. These changes can be made any time up until the load process is initiated.

The LOAD parameter will define how the system is loaded, together with providing a path to other information required to bring the LPAR to a position where users can log on. They are passed as an 8 character parameter the first 4 of which specify the physical device where the IODF is stored. The remaining 4 characters provide further information required to perform the IPL.

These additional parameters, such as the Nucleus ID to be used to "boot" z/OS and which messages will be displayed on the NIP consoles are also passed.

Which LOADxx member is to be used to load the system completes the LOAD parameters, and this contains additional system settings, specifying the location of further parameters required to bring the operating system up to a point where it is ready for use.

This section explains what effect the LOAD parameter has on the shape of the z/OS partition, detailing how the IPL process uses entries to configure and start all parts of the system. This will include details on how system parameters are selected for use, together with which messages and prompts are to be displayed on the console. Some of the risks involved with altering LOAD parameters, together with their potential effects will also be explained.

ADDRXXMN

ADDR	the device number containing the IODF
XX	Identifying characters for the LOADxx member to be used
M	Initial message suppression setting to suppress most information messages and use LOADxx system parameters.
N	Nucleus ID to be used

5.1.3.1 LOADxx Member

The LOADxx member is one of those z/OS “moveable feasts”. There is no fixed place for it to reside. There are rules on placement of the member but you can pretty much guarantee that 2 separate installations won’t have them in the same place. This member provides a really good example of why it is critical to foster the relationship between Audit and the Systems Programmers. This factor is examined in a bit more detail in 5.1.3.2.

The LOADxx member is selected through the use of the LOAD parameter on the system control (SYSCTL) frame of the system console.

The information held in LOADxx starts with details about the expected I/O configuration as defined in a specified IODF Dataset. It moves on through lots of data about the nucleus for the LPAR including: alternative nucleus ID, architecture level of the nucleus, and modifications to be made to the nucleus at IPL time via the NUCLSTxx member that is controlled by your Systems Programmers.

Data about both the master catalog and the PARMLIB concatenation is also held in LOADxx. Along with the name of the sysplex which this LPAR is part of and pointers to various other configuration members (e.g. IEASYMxx and IEASYSxx), the LOADxx member literally shapes the local environment.

For quite some time it has been possible for your Systems Programmers to define a single LOADxx member to work with multiple systems. This is achieved using filtering keywords like HWNAME, LPARNAME and VMUSERID, more of which later...

Updates to the LOADxx member are made by Systems Programmers on an active z/OS system. This means that it should be subjected to the installation’s standard external security manager processing. Although, unless you use an external security manager which allows for the securing of members within a library, the influence you can exert extends only to the container library not the member itself.

Changes to LOADxx are perfectly normal for z/OS operation and the Systems Programmers do require access to do so. Implementing close scrutiny of who is making those changes can help to secure against unexpected changes but there are no guarantees.





Let's go back to thinking of me as an unscrupulous individual for a moment. If I needed to build a Front Door into a z/OS system, update access to the various PARMLIBs would make my life very much easier. I could change what code would be running allowing me to implement programming to make copies of data, bypass security under certain circumstances or provoke a denial of service attack!

I've been in this business for the better part of 30 years and I have never yet come across a z/OS system that I couldn't hack into. Let's all start to take this more seriously and you will be able to feel confident that if you come across a "Black Hat" hacker (i.e. one of the bad guys - I'm a "White Hat" aka one of the good guys), he can't get as far as he was hoping in your z environment.

5.1.3.2 Location

What follows is a detailed list of the search order z/OS uses when trying to find out what LOADxx member to use. The search ends when the first LOADxx member is found so it is vital to keep track of how many LOADxx members there are in how many different places.

1. The IODF volume (specified for the LPAR at LOAD time from the system console) is searched for a SYSn.IPLPARM PDS. N is a numerical value 0 through 9.
2. If no SYSn.IPLPARM dataset is found, SYS1.PARMLIB is searched for on the IODF volume.
3. If no valid dataset has been found on the IODF volume, the System Resident (SYSRES) volume is searched for the SYS1.PARMLIB library. IBM does not recommend that the IODF is specified on the SYSRES volume because
 - a. Replication of the SYSRES volume is difficult if the IODF is present as it is a VSAM dataset.
 - b. If the SYSRES volume is shared across multiple logical partitions, when the systems are IPLed there is an increased chance of I/O contention if the IODF is present.

If a SYSn.IPLPARM or SYS1.PARMLIB has been found by z/OS, but the LOADxx member specified in the LOAD parameter is not found, the partition will fail to load, entering a wait state, i.e. no processing can be performed on the LPAR.

If the LOADxx member is stored in SYSn.IPLPARM the Systems Programmer must specify the high-level qualifier in the IODF statement. If the high-level qualifier is not specified in the IODF statement in the LOADxx member, z/OS will use the high-level qualifier of the SYSn.IPLPARM PDS on the IODF volume as the default.

The practice of "hiding" various parts of the IPL chain from inquiring minds is quite common. For example it is not unusual to have the SYSn.IPLPARM dataset uncataloged. This can be used to obscure the structure slightly but cannot be used to replace "proper" security.



In such cases it will be vital to liaise with the Systems Programmers to understand what has been implemented. This is particularly true in the case of complex, compound symbols where multiple symbolic values are concatenated together. In fact, it is always better to ask the Systems Programmers because you might find yourself trying to audit the z/OS environment whilst work is being undertaken to change it. Only the Systems Programmers will be able to tell you this.

5.1.3.3 Example and Keyword Syntax

Some pretty standard "MVS rules" apply to the syntax of LOADxx parameters. I call them "MVS rules" because they date back such a long way - e.g. each record in the LOADxx member consists of records which are 80 columns (Parameters begin in column 1. Data begins in column 10. Columns 73 thru 80 are ignored).


This comes from formatting requirements of punched cards!

The fields are all column-dependent and columns which do not contain data must contain blanks. In ISPF the member should have a profile of UNNUM - which automatically blanks out columns 73 to 80. All data must be left-justified within the column range for that parameter.

Any lines that begin with an asterisk in column 1 are comments. And finally, blank lines are ignored completely. Both can be used to make the LOADxx member more readable.

The following example shows the type of thing to expect in a typical LOADxx member:

IODF	00	SYS1	OSSYS1PR 00 Y
NUCLEUS	1		
SYSCAT	SYSV001	3	PLEX1.MASTER.CATALOG
SYSPARM	01		
SYSPLEX	PRODPLEX		
NUCLST	00		
IEASYM	00		
PARMLIB	SYS1.SHARED.PARMLIB		
PARMLIB	SYS1.PLEX1.PARMLIB		



IODF identifies the I/O definition file that contains the I/O configuration defined to your system through the hardware configuration definition (HCD). Columns 10 and 11 contain the suffix that is appended to form the name of the IODF data set and 13 thru 20 contain the prefix. In the case of the example above it produces a dataset name of **SYS1.IODF00**.

Columns 22-29 contain the Operating System identifier as defined on the HCD, in the case OSSYS1PR. The final 2 parameters are an eligible device table identifier (in this case 00) and an indicator as to whether the system should load all the support modules for devices defined in the IODF (Y).

NUCLEUS identifies the IEANUC0x member of SYS1.NUCLEUS that your system is to use. Column 10 is a single digit field and the only parameter available on this keyword. It provides the suffix to the nucleus member of SYS1.NUCLEUS to use. In the example above it resolves to SYS1.NUCLEUS(IEANUC0**1**).

SYSCAT represents the information needed to process the master catalog correctly. There are a number of positional parameters here. Columns 10 thru 15 contain the volume serial number of the disk to use. Column 16 can contain either 1 or 2 and it says whether or not SYS% will be converted to SYS1. Column 17 is another single digit field used to represent the number of levels that can be specified in an ALIAS. Columns 18 and 19 contain the Catalog Address Space (CAS) service task lower limit. Columns 20 thru 63 represent that actual name of the master catalog. So this example shows dataset **PLEX1.MASTER.CATALOG** on volume **SYSV00** will **convert SYS% to SYS1** whilst also allowing only a **single level alias**. The **default** service task lower limit is applied.

SYSPARM identifies which IEASYSxx member to use for IPL. In this case it would be member IEASYS**01**.

SYSPLEX specifies the name of the sysplex in which the system participates. The example above sets the sysplex name to **PRODPLEX**.

NUCLST identifies the NUCLSTxx parmlib member that your system is to use. The NUCLSTxx member must reside in the same data set as the LOADxx member. In this example NUCLST**00** would be selected.

There is another parameter not listed here (see 5.1.3.3.8) which allows your Systems Programmer to specify that they would like the system to hang (load a wait state) if any of the INCLUDE statements in the NUCLSTxx member specify a member that cannot be found in SYS1.NUCLEUS - this is a great, but under-used, security feature.

IEASYM specifies which member to use to define system symbolics that will be available on the IPLed z/OS LPAR. In this example IEASYM00 will be used.

Each PARMLIB keyword specifies a data set that will be included in the logical parmlib concatenation. The parmlib concatenation consists of up to 16 PARMLIB data sets and SYS1.PARMLIB.

When there is more than one PARMLIB statement, the statements are concatenated and SYS1.PARMLIB is added at the end. This example shows the use of cataloged datasets but there is a volume serial number field (columns 55 thru 60) which can enable the use of uncataloged resources.



There are a lot of parameters available in the LOADxx member not shown here. If you require full details please refer to the IBM documentation for the release of z/OS that you are running. If you are running an audit then you should, most definitely, be consulting with the Systems Programmers.

5.1.3.3.1 IODF Decoded – ConfigID as link to OSCP

Systems Programmers used to have to define the operating system to be loaded on an LPAR using the separate MVSCP function until HCD was introduced in MVS/ESA v5. When the MVSCP process was complete they would have to define the IODF!

These steps have been consolidated in HCD. When defining an available operating system (OSCP) using HCD your Systems Programmer need only define 3 parameters. These are the ConfigID, the Operating System Type, and the Description field.

This information about defined operating systems and available hardware is all stored in the IODF when you complete the multi-step HCD process. Provided no syntactical problems were found (any invalid entries will cause the write process to fail, issuing error messages) it is now available to perform an IPL by specifying the IODF name in the LOADxx member.

The first parameter in LOADxx specifies the IODF to use when IPLing the LPAR. In the example in the previous section SYS1.IODF00 would be selected. There are other parameters available on the IODF keyword statement in LOADxx that have not been shown in the example.

One of these additional parameters (ConfigID) links the entry in the LOADxx member to what has been defined in the OSCP on HCD. This is specified in columns 22-29 and contains the Operating System Configuration Identifier field, which is the ConfigID specified in HCD when defining the operating systems available to be loaded.

This ConfigID details the Operating System Types available to be loaded on LPARs defined on the systems, and configured to use the IODF in question. Specifying this parameter in the LOAD member allows an organization to limit the access of the OS of the LPAR in question to the specific devices defined.

The ConfigID must be picked from one of the entries available in the IODF used to IPL. This is not an issue if only one operating system has been defined to the IODF. The field can be left blank, and this one OS will be loaded. If there is more

than one operating system defined to the IODF it must be specified, or the LPAR will enter a wait state.

5.1.3.3.2 IEASYM – Path to System Symbols

System symbolics have been around for a while now. They are extremely valuable when setting up parallel sysplex. The ability to substitute values into symbolics at IPL and started task run time means less maintenance of separate members for separate LPARs.

Linking off from the LOADxx member keyword IEASYM is the IEASYMxx member which contains statements that both define static system symbols and specify the IEASYMxx member(s) that z/OS is to use.

Symbolics should **NOT** be used in LOADxx. The system must complete processing of LOADxx to define substitution texts to the system symbols. Therefore, the system might not substitute text for system symbols that you use in LOADxx.

Systems Programmers can use the HWNAME, LPARNAME, and VMUSERID parameters to limit the scope of statements in IEASYMxx. When you specify one or more of those parameters, the parameters on that statement apply only to the system that HWNAME, LPARNAME, or VMUSERID identify. More on this capability in 5.1.3.3.9...

Some 800 fixed system symbolics can be assigned alongside those supplied by z/OS. One last comment on defining symbolics - the length of the resolved substitution text cannot exceed the length of &symbol, including the ampersand on &symbol and excluding the single quotation marks on 'sub-text'.

Symbolics can make administering a sysplex a greatly simplified task but they can be complicated to interpret visually. This is particularly true if you are trying to audit multiple LPARs that are all shaped by the same IEASYMxx member.

Use of symbolics in PARMLIB members and/or Started Tasks is indicated by the use of the name found in the SYMDEF keyword prefixed by ampersand (&). If you see this during an audit you may need to ask for assistance to make sure that you are interpreting the contents correctly.

Risks that could arise from the abuse of symbolics would, primarily, be based around changing the resolved values. The damage that could be done would depend on what symbolics were defined.

There is no way to specifically secure the contents of symbolics. The approach to take is to make sure that there is adequate external security manager protection on the library(s) containing the IEASYMxx member. Auditing all activity (authorized and non-authorized) against this resource ensures that you can at least find out who made changes.

5.1.3.3.3 SYSCAT – Path to System Catalog

Not to be confused with the DB2 SYSCAT, the master catalog effectively provides an index to the datasets which are cataloged on the system. It provides end users with the ability to locate a data set by name, without knowing where the data set resides. In much the same way that you type a URL to get to your favorite web sites rather than having to remember the IP addresses.

By cataloging data sets, end users will need to know less about your storage setup. This means that data can be moved from one device to another, without requiring any change in JCL DD statements which refer to an existing data set.

In a GDPS environment, where processors can be in different buildings, and indeed different cities the need for a master catalog to locate both configuration



and application datasets is amplified. This data may not be in the same Data Center as the processor that is using it, and indeed data may be shared between processors and systems in different locations.

Cataloging data sets also simplifies backup and recovery procedures. Catalogs are the central information point for VSAM data sets; all VSAM data sets must be cataloged. The same is true of all SMS managed data sets, with the majority of data stored on system z being SMS managed.

It is normal operating practice to define a new master catalog when undertaking major system upgrades. This new master catalog is then deliberately brought in during an IPL to change the datasets that will be used.

This is done by setting up a new LOADxx member in SYSn.IPLPARM with the SYSCAT keyword definition changed. This will affect everything that is picked up by its catalog entry. Entries which have a specific volume serial number defined will not be altered by changing the master catalog.



Unexpected alteration of the master catalog specified in the SYSCAT keyword parameter of LOADxx can lead to anything from security policies which no longer apply due to changed high level qualifiers through to complete denial of services attacks!

The security of this element is entirely dependent on the security applied to the LOADxx member(s).

Whilst it is necessary for Systems Programmers to be able to change the contents of LOADxx, it is not a daily task. It may be appropriate to use a break-glass id for access even though this significantly reduces the accuracy of any audit trail.

5.1.3.3.4 SYSPARM – Path to IEASYSxx

As with many of the z/OS system parameters, values can be supplied either by directly coding them into the specified IEASYSxx member in PARMLIB or by entering them as operator commands during the IPL process.

Current advice suggests that the majority of installation default system parameters should be kept in IEASYS00. These are the parameters that don't change on a regular basis, such as between normal IPLs.

Alternate IEASYSxx members should contain the settings and parameters that are subject to change as part of your organization's routine operations. Use of these alternative system parameters reduces the need for operator intervention during the IPL.

Which suffixes are in use, specifying the IEASYSxx member(s) to be used for the IPL process are listed in the SYSPARM parameter. This parameter can be located in either/or (or both) LOADxx and IEASYMxx. The members pointed to in the LOADxx member are processed before those specified in IEASYSxx.

For example if the LOADxx member contains the statement SYSPARM(01,03,L) and the IEASYSxx member, SYSPARM(02,05,L) the IEASYSxx members would be processed in the following order:

- IEASYS00 – the installation defaults, these are automatically concatenated first
- IEASYS01 – 1st member specified in LOADxx
- IEASYS03 – 2nd member specified in LOADxx
- IEASYS02 – 1st member specified in IEASYSxx
- IEASYS05 – 2nd member specified in IEASYSxx

If any of the members specified in the SYSPARM statements are not valid (i.e. do not exist) then a console message is issued asking you to re-specify the system parameters. This offers the opportunity for the Initial Message Suppression settings to be overridden during the IPL process which could lead to unauthorized system changes being applied.

5.1.3.3.5 PARMLIB – Path to PARMLIB Concatenation

The SYSx.PARMLIB libraries contain members detailing parameters for z/OS and many of its subsystems. They contain vital operating system parameters important to the reliable running of your z/OS operating system LPAR such as the LOADxx members, clock settings and concatenation data for LNKLST libraries. The Master Scheduler JCL, which controls system initialization and processing, is also found in the PARMLIB dataset.

Up to 17 datasets (16 plus SYS1.PARMLIB) can be concatenated to PARMLIB.

When customized the PARMLIB concatenation order is defined by using PARMLIB statements in the LOADxx members. If no PARMLIB statement is present in the LOADxx member only SYS1.PARMLIB is used.

The PARMLIB datasets are concatenated in the order specified in the LOADxx member.

The option to specify the dataset name AND volume offers the opportunity to use non-cataloged datasets within the PARMLIB concatenation, even using an alternate SYS1.PARMLIB by entering a specific volume. If no volume is specified, IPL processing will first search the master catalog for an entry, and then the system residence (SYSRES) volume.

A search of the master catalog for the dataset can be enforced by specifying '*MCAT*' in the volume field. Alternatively, '*****' or '&SYSR1' will specify that the dataset should be located on the SYSRES.

Ensuring that the order in which PARMLIB datasets are specified for use is correct is especially important. The first valid entry for a member that is found in the order that the datasets are specified is the one that will be used when initializing it.

5.1.3.3.6 SYSPLEX – The Name of the Parent Sysplex

I've talked a lot about the strengths of System z when configured in a sysplex. The SYSPLEX keyword parameter in LOADxx defines which sysplex an LPAR is a member of. It is also the substitution text for the &SYSPLEX system symbol.

The sysplex name must match the name specified in both the SYSPLEX parameter of the XCF couple data set format utility and the SYSPLEX parameter in the COUPLExx PARMLIB member.

LOADxx defines the substitution text for &SYSPLEX early in system initialization so that other PARMLIB members can use it. So if you plan to use the &SYSPLEX system symbol in PARMLIB your Systems Programmer should specify the sysplex name in LOADxx. To ensure that the name in COUPLExx matches the one in LOADxx, specify the &SYSPLEX system symbol on the SYSPLEX parameter in COUPLExx.

If you do not specify a SYSPLEX statement in LOADxx the system uses the value LOCAL until after it processes the COUPLExx PARMLIB member. Then it uses the name specified in COUPLExx. Doing this can leave a brief window of opportunity for a hacker when system resources are not protected under their usual names.





5.1.3.3.7 NUCLEUS – Path to Platform Architecture

The NUCLEUS represents the core of z/OS. It encompasses things like the I/O supervisor, various resource managers (ASM, RSM, SRM, etc.), interrupt handlers, and the most essential service routines. In other words it is the element of the operating system which negotiates the relationship between the hardware (Platform Architecture) and the software (both operating system and applications).

Some applications environments also need part of their operating code loaded into Nucleus - e.g. CICS. But by allowing update to SYS1.NUCLEUS an organization opens up a potential Front Door to the system. If I can include code in nucleus then I can alter the way that z/OS will behave.

It is therefore critical that there is control over who can update the SYS1.NUCLEUS library. This is not something that happens every day so you could consider using a break-glass id. However, in an attempt to make things more auditable rather than just more awkward, it would be better to audit all access to SYS1.NUCLEUS successful or otherwise and maintain a normal access list to the resource.

Best Practice is to create a baseline copy of SYS1.NUCLEUS. This can then be used to compare the contents at any time subsequent to the copy being taken. It's really the only way to be 100% certain nothing has changed.

The NUCLEUS statement in LOADxx identifies the IEANUC0x member of SYS1.NUCLEUS that your system is to use at IPL time.

5.1.3.3.8 NUCLST – Path to NUCLSTxx

The NUCLST statement in LOADxx identifies the NUCLSTxx PARMLIB member that your system is to use at IPL time. It also specifies whether the system is to enter a wait state if any of the INCLUDE statements in the NUCLSTxx member specify a member that cannot be found in SYS1.NUCLEUS.

If changes are required to the nucleus then the safest way to address them is to make use of the PARMLIB member NUCLSTxx. This member allows your Systems Programmer to load (or delete) modules into the nucleus at IPL time without having to re-link-edit the actual nucleus code. This maintains the integrity of the IBM supplied nucleus module whilst allowing the installation of specialist system code such as SVCs and is considered to be Best Practice.

The example earlier in this chapter does not show one of the simplest security measures that can be implemented for nucleus. That is the parameter which controls what to do if a specific problem is found. If your Systems Programmers set column 13 to upper case Y in the NUCLST keyword parameter of LOADxx, it tells the system to hang (i.e. wait state) should it discover an entry in a NUCLSTxx member which does not map to an existing member in SYS1.NUCLEUS.

5.1.3.3.9 Member Filtering – Flexibility, Efficiency

IBM has been consistently improving usability of the System z environment. Quite some time ago they introduced the ability to maintain a single LOADxx member for an entire installation. This ties in quite nicely with the suggestion that a single IODF should contain all information for your entire installation.

Conceptually it can reduce the complexity of maintaining a multiple LPAR environment. Oftentimes though filtering is not used to best effect because it can look confusing at first. This has been allowed to continue because - if the Systems Programmers think it is confusing how on earth are we going to audit it?! In actual fact, implementation of this incredibly flexible and efficient filtering practice can simplify the process required to audit the environment.

Firstly, a single LOADxx member can describe the entire System z installation.

No more hunting for the starting point of the audit. No more battling with obscuring methods of protection of the system configuration data.

Secondly, correct implementation of filtering in LOADxx can provide for increased security for the period in an IPL where there is no external security! Using the filter keywords (HWNAME, LPARNAME, and VMUSERID) can help to tie an IPL to a specific LPAR.

I'd advise using a large helping of common sense when applying the filtering though. The last thing you want to end up with is a parallel sysplex where specific operating systems can only be brought up in specific hardware partitions. This goes against the flexibility and recoverability that parallel sysplex stands for!

There will be more detail on filtering in LOADxx in section 5.1.3.6...

5.1.3.4 Risk – Orphan Members

Over the many years (in most cases decades) that System z has been an integral part of doing business a whole lot of changes have occurred at the hands of many Systems Programmers. Most of these changes are planned but some will have taken place under emergency conditions.

One of the first things that a Systems Programmer is taught is - before you make any changes... take a backup. I know that every time I have ever changed a LOADxx member I've created a duplicate member with a new name in the same PARMLIB. I will also admit that I haven't always cleaned up after myself by deleting the old member after the change has been successful.

Often the lack of cleanup is because your Systems Programmer has had to move on to the next "fire". But it's actually pretty unusual for a Systems Programming team to delete backups of any kind - just in case.

Orphan Members occur when the LOADxx and IODF get out of sync. In other words there is no OSCP to match the LOADxx definition. It's not something that is easy to monitor.

The risk is that someone could add an OSCP for a LOADxx member allowing them to IPL an LPAR outside of the normal operating environment. For example I could bring up a system that is a duplicate of a production LPAR - with all the associated hardware connections. Unless automation is watching for unexpected LPARs there would be every chance I could do this undetected. Thus leaving myself with the perfect, secret environment from which to take copies of your customer data to sell to the highest bidder!

The risk can be mitigated by keeping a clean list of LOADxx members. Consolidating all of a sysplex's LOADxx requirements into one member would significantly reduce the number of LOADxx members required legitimately. Doing this would lead to a much easier environment to monitor for audit purposes.

Sadly there is no method of policing orphan LOADxx members supplied by IBM.

5.1.3.5 Nucleus Initialization Process (NIP)

The Nucleus Initialization Process is one of the earliest parts of the IPL process. Communication with resources like the NIP console that will be used to monitor the IPL and reply to system prompts as and when they are required.

When the IPL process is started by the load parameter being entered on the HMC, NIP processing reads it to find out how to proceed. UCW entries are also mapped to their corresponding UCB, testing the availability of DASD devices and initializing non-DASD.



Parameters and the location of additional settings required for NIP processing are detailed in the LOADxx member. 4 of the keyword parameters in LOADxx have a direct effect on how NIP processing is completed. These are the IODF statement, NUCLEUS, SYSPARM and NUCLST. The effects of altering one of these from the expected value can range from minor inconvenience, to an LPAR entering a wait state at the next IPL.

IODF is the first keyword parameter in the LOADxx member. It details the IODF that will be used to IPL the LPAR. Details of the I/O devices that are defined to the LPAR, including the all-important NIP console used to control and monitor the IPL process are defined to the IODF.

The NUCLEUS parameter specifies the member of SYS1.NUCLEUS that is to be used. The value specified is appended to IEANUC0x to give the member name. This entry can be overridden by the operator by entering an alternative identifier at the start of the IPL process.

While the default for this setting is 1, if an alternate nucleus id is selected, a corresponding architectural level extension of this nucleus must exist. This can be set using the LOADxx ARCHLVL keyword statement, and defines the mode that the operating system will run in. The default for ARCHLVL is 2, which specifies z/Architecture mode. If an invalid ARCHLV setting is specified the system will revert to the default.

SYSPARM affects the order in which NIP processing uses parameters from the IEASYSxx members storing system settings. During NIP processing this can affect settings such as whether to perform a CLPA, the operator prompt to initialize the clock and which installation SVCs NIP processing places in the SVC table. The LAST entry found when searching in the specified order is the one that will be used.

Finally there is NUCLST. The 2 characters specified to this parameter are the suffix which identifies the particular NUCLSTxx parmlib member that will be used. This NUCLST member must be in the same dataset as the LOADxx member that references it.

This NUCLST parameter has an additional field that can ensure that the partition will enter a wait state if any of the required INCLUDE statements in the NUCLSTxx member cannot be found. The default is not to wait state. My preference would be to enable this feature. If the circumstance has arisen it is likely a deliberate sabotage event - this is such a critical element that the Systems Programmer making changes legitimately will **rarely** make an error.

NIP processing has one final task which is to start the Master Scheduler. This system element initializes the required sub-systems. This is when JES, VTAM, TSO, etc. are started. After this processing has been completed the system is available to start work.

Alteration of any of these parameters can affect the IPL process. It is possible to change everything from the devices attached to the LPAR to which sub-tasks are started by the Master Scheduler at the end of the IPL process. It can be an extremely powerful tool for a hacker to gain access to.

5.1.3.6 **LOADxx Filtering**

Consolidation of LOADxx using filtering can simplify maintenance of the sheer quantity of members that tend to accumulate over time. It can also help keep control over the number of Orphan Members and generally improve security on the platform when it most needs it, i.e. IPL time.

Brief technical jargon alert...

This section looks in detail at the filtering capabilities. Let's start with the technical basics. The LOADxx filter keywords HWNAME, LPARNAME, and VMUSERID allow your Systems Programmer to use a single LOADxx member to define IPL parameters for multiple systems. At IPL, the initial values of the keywords are set to match the actual values of the system being IPLed.

The LOADxx filter keywords are hierarchical in their implementation. HWNAME is at the top and VMUSERID on the bottom. The diagram below shows this hierarchical relationship.

The HWNAME parameter is used to specify the Control Processing Complex (CPC) name. HWNAME also sets both the LPARNAME and the VMUSERID to their default values.

The LPARNAME parameter, the next level in the hierarchy, is used to set the Logical Partition name. LPARNAME also sets VMUSERID to the default value. The value of HWNAME is unchanged.

The lowest level of the hierarchy, the VMUSERID parameter, specifies the userid of a z/VM system under which a z/OS image is running as a guest.

Due to the hierarchical relationship of these keywords the following rules apply to the specifications in the LOADxx member:

1. Keyword HWNAME is in effect from its occurrence to the next HWNAME statement in the member or to the end of the LOADxx member.
2. Keyword LPARNAME is in effect from its occurrence to the next HWNAME or LPARNAME statement or to the end of the LOADxx member.
3. Keyword VMUSERID is in effect from its occurrence to the next HWNAME, LPARNAME, or VMUSERID statement or to the end of the LOADxx member.

These statements hold true only in a partitioned environment. If the system being IPLed is not running as LPAR mode or on VM as a guest then LPARNAME and VMUSERID won't be meaningful.

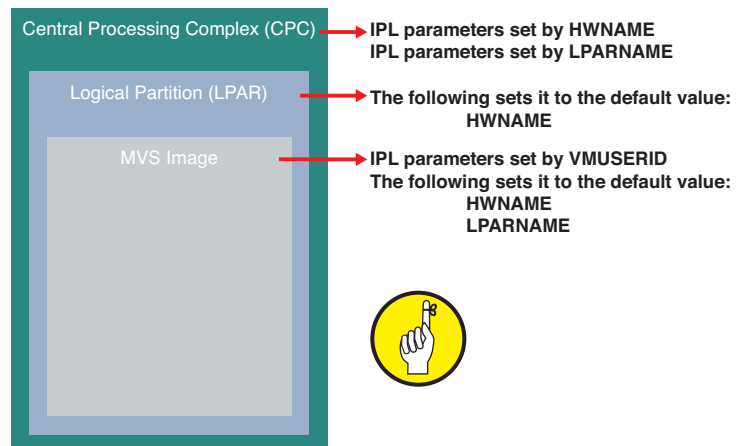
When another LOADxx statement (such as SYSCAT or IODF) is specified, the current HWNAME, LPARNAME, and VMUSERID filters are compared to the actual system values. If they all match then that LOADxx statement is accepted as applicable to the system being IPLed. If they do not match then the IPL will abort (wait state).

Honestly, it's easier to do than it is to explain! But you will need the cooperation of the Systems Programmers to get LOADxx filtering implemented.

5.1.3.7 IMSI - Initial Message Suppression

During an IPL, messages are routed to the NIP console. These show the progress of the IPL, together with reply messages that would allow the parameters to be altered. In an environment which has tightly controlled automation you may have to go hunting for these.

The final character of the LOAD parameter is used to control which display messages are sent to the console, together with which system parameters the



IPLing a z/OS logical partition

operator will be able to override by replying to prompts.

The Initial Message Suppression character (IMSI) has a number of settings which handle the process from 'A' for full operator intervention to a blank or period for fully automated, hands off.



The table below details all of the available settings and the effect each character will have on the displayed messages and reply prompts delivered to the console. The value used for this parameter can enable or prevent a carbon-based life-form from changing shape of your z/OS environment during the IPL process.

IMSI Character	Display info messages	Prompt for Master Catalog response	Prompt for System parameters
A	Y	Y	Y
C	N	Y	N
D	Y	Y	N
M	Y	N	N
P	N	Y	Y
S	N	N	Y
T	Y	N	Y
. or blank	N	N	N

Suppressing messages does not stop them being written to the console log. But it can help to prevent overflow of messages on the console at what is a very busy time.

Suppression of prompts for replies regarding system parameters can be used to ensure that system settings cannot be altered during the IPL process. With proper planning this provides the re-assurance that settings have been fully investigated prior to them being set for the next IPL.

In some cases, however, if the required settings cannot be found during the IPL process a prompt will be sent to the console asking for the correct parameter to be entered regardless of the IMSI setting. This can allow settings that do not meet internal or audit requirements to be used, so all settings should be verified prior to the start of the IPL process.

5.1.4 SYS1.SVCLIB

SYS1.SVCLIB is the z/OS library which contains **SuperV**isor **C**all (SVC) programs. These SVCs alter normal z/OS operation for specific ends. For example the Type 3 SVC which enables Multi Region Operation (MRO) in CICS.

Typically updates are made to this library only during installation and/or maintenance of systems software. All access to the library should be audited whether it is successful or not.

It's impossible to understand what each SVC specifically does. There are published lists of which software products typically use which SVCs though. As an auditor the best one can hope to do is match up the installed SVCs with the installed software. If there is a mismatch there may be a problem.

Best practice is to keep SYS1.SVCLIB as the exclusive container for IBM supplied

SVCs. Some third party products also require SVCs. These should be kept in a separate z/OS library. It is important that the library must be cataloged in the Master catalog (i.e. it must not have an ALIAS).

What can be helpful in keeping track of changes to SYS1.SVCLIB is baselining the library. That allows you to run a compare between the copy and the current library to make sure there are no differences. It is important to establish that the contents of the members have not changed as well as assessing any new members which have arrived since the last baseline.

5.1.5 SYS1.NUCLEUS

SYS1.NUCLEUS is the z/OS library which contains all of the programs required to initialize the nucleus (or core z/OS functions). It is very unusual that the contents of this library need updating. Examples of exceptions are: if a new version of the operating system; or an application environment like CICS or WAS, is being installed.

Under normal circumstances no one should require update authority to the library. Either a break-glass id should be used for software installs or (preferably) a temporary connection to allow access for the duration of the installation project. However external security is implemented it is vital that all activity against this library (both successful and unsuccessful) should be audited.

Contents of SYS1.NUCLEUS are notoriously difficult to audit. They are executable modules with no source usually supplied by an external vendor - generally IBM.

It can also be valuable (like with SYS1.SVCLIB) to take regular copies of the library and make sure that nothing changes within it - at least not without a change record!

5.1.6 IPL Text

In order to start an IPL from a specific volume it has to have been initialized as an IPLable pack using the IBM supplied program ICKDSF. This will format the DASD device as an IPL pack by building the bootstrap records allowing it to be a bootable disk.

When this task is run, the IEAIPLO0 (also known as the IPL text) is supplied as input for the job completing the initialization process. This is written to the 4th record on cyl0, trk0 of the IPLable volume, the previous 3 records being the bootstrap data and the volume label. This information should only be used for "showing off" as it is of no practical purpose during an audit!

This IPL text is only created from the IEAIPLO0 member when the device is initialized as an IPL pack. This reduces the opportunity for any of the settings to be altered. It does not however remove them, and access to, and attempts to run the ICKDSF program should be limited only to those that require it for their job. This should only really be z/OS Systems Programmers and their use of the program should be monitored and audited.

When the ICKDSF task is run to prepare an IPL volume, it is imperative that the file members used to create the IPLTEXT meet all of the internal requirements for preparing an LPAR. Ask the Systems Programmers for copies of the job used to prepare the volume and check with them that the file that is used for input in the **IPLTEXT** DD statement meets all of these standards. In the example files from the IBM supplied SYS1.SAMPLIB have been used.



IPLing a z/OS logical partition

```
//IPLTXT EXEC PGM=ICKDSF,COND=(0,NE)
//SYSPRINT DD SYSOUT=*
//IPLTEXT DD DISP=SHR,DSN=SYS1.SAMPLIB(IPLRECS)
// DD DISP=SHR,DSN=SYS1.SAMPLIB(IEAIPL00)
//IPLVOL DD DISP=SHR,VOL=SER=SYSRES,UNIT=3390
//SYSIN DD *
REFORMAT DDNAME(IPLVOL) IPLDD(IPLTEXT) NOVERIFY BOOTSTRAP
/*
```

When the IPL process is started the bootstrap is loaded. This is what effectively boots the z/OS system to begin the load process. The bootstrap then reads the IPL text from the disk and uses this information to continue the IPL process.

The IPL text contains the information required to prepare the LPAR for the starting of the programs and modules that make up the operating system. Its tasks include the clearing of the Central Storage Area (CSA), and the definition of storage in preparation for the Master Scheduler, together with locating the SYS1.NUCLEUS on the SYSRES pack and loading the IPL resource initialization modules.

This information required for an IPL is held on the first 4 records of the very first area of useable space on the disk. The IPL process knows to load the required information from this area, and will not load from any disk if it has not been initialized as an IPL volume.

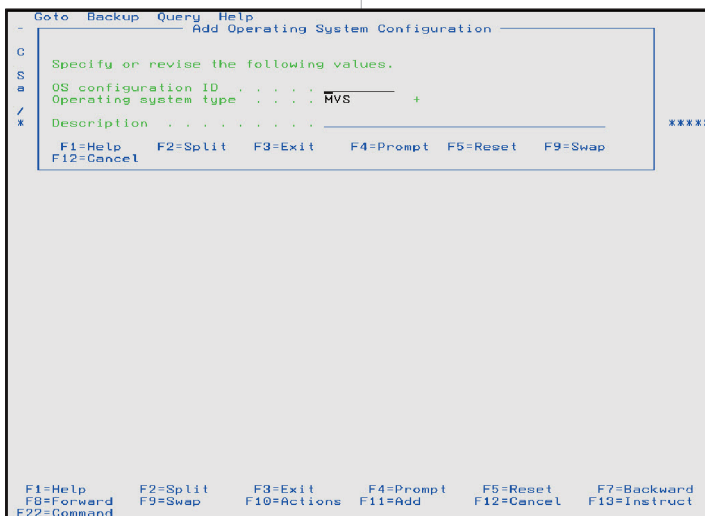
5.1.7 The IODF Dataset – OSCP

When Systems Programmers make changes to the IODF dataset, one of the functions available is the definition of which operating systems can be loaded. This uses the OSCP (Operating System Control Program) functionality that now forms part of the HCD functions. It is surprisingly simple to use!

The production OSCP contains all the I/O configuration information that is currently defined to the OS in use by a specific LPAR defined to the zEnterprise by the IOCP.

Any alterations are not made here, but into a separate work IODF. This is to make sure that there is a fallback option available.

The work IODF must then be built as a production file and activated.



While there can be multiple production OSCP's the only one in use by the LPAR is the one specified by the LOADxx member is during the IPL of the LPAR.

Any of the commercially available operating systems can be configured to run on the z platform, but when the definitions are specified to the IODF the options for OS type are limited to either MVS or VM. Once the operating system has been defined to the IODF, and the system datasets are in place, it can be loaded on to one of the defined LPARs. A single IODF can contain the configuration details for multiple z machines.

These operating system definitions are written to the IODF when the configuration process is complete. LPAR's that have been defined to the system can then be configured to load and run these operating

systems. Only one definition per operating system is required for each sysplex.

If an operating system is loaded on an LPAR, there must be network devices,

DASD, NIP consoles and other devices defined in order for it to be able to run. The sharing of data between different operating systems can lead to inconsistencies with regard to how access to this data is controlled.

5.1.7.1 **HARDWARE**

It is possible for multiple z machines to share a single IODF. In fact this is what IBM recommends as Best Practice. But there must be a way to separate the definitions in the IODF for each separate CPC.

Each processor is assigned its own specific ProcessorID, and when a new processor is defined to the IODF this processor id must be specified, together with its type, model, configuration mode, number of available Channel subsystems and serial number. There is also the optional description field, and if the processor is to be part of a microprocessor cluster, SNA address.

If an SNA address is defined the CPC name in this definition should match the ProcessorID as this is the name that is used by the Hardware Management Console.

The ProcessorID is recorded in the IODF and is used when defining the I/O configuration for any of the LPAR's defined to it. With there being up to 60 LPARs configurable on a z10 EC, and the fact that a single IODF can be shared between processors, this ensures the separation of device and channel definitions between different CPCs.

Once the ProcessorID has been defined it can be used in the HWNAME keyword parameter in the LOADxx and IEASYMxx members created to IPL an operating system. This enables the filtering of defined values to those for the specified CPC. When HWNAME is specified in the LOAD member LPARNAME and VMUSER are reset to their default values.

First the processor must be defined to the work IODF. Then the available channels, control units, and all other devices required for I/O processing can be defined. All of which must happen before assigning these devices to the defined LPARs on the system.

5.1.7.2 **LPARNAME**

The LPARNAME in LOADxx equates to the Image profile name that was assigned to the LPAR when it was defined in the IODF.

When an LPAR is defined it is given a name that will differentiate it from other LPARs within the same processor defined in the IODF. On HCD this is known as the Partition Name and when defining a partition you also assign it to a predefined Partition Number. At this point the system can also be told whether the LPAR will be used for running an operating system (OS), as a coupling facility (CF) or both.

The LPAR Name is used to separate the definitions in the IODF. In other words, when, for example, a CHPID is defined to the IODF the PARTITION sub-parameter will specify the particular LPAR(s) that it is being defined to. In the example below a channel has been defined to the IODF, running under **LCSS0**, although CHPIDs can be assigned to multiple LCSSs.

If CHPIDs are assigned to more than 1 LCSS they are referred to as being defined as SPANNED. This can leave potential exposures as LPARs that are on different channel sub-systems will share access to CHPIDs (and possibly have access to their associated devices) with the possibility of no recording or auditing of access being recorded on 1 or more of the LPARs.

The IODF shows that this channel is defined for use by the **PRODN** LPAR and has been allocated the **Physical CHannel IDentifier 101**.

IPLing a z/OS logical partition

```
CHPID PATH=(CSS(0),16),SHARED,PARTITION=(PRODN),(=),  
PCHID=101,TYPE=OSN
```

This partition keyword is used throughout the IODF when specifying devices to differentiate which partition the devices are being defined to. Ensuring that the correct logical partition name is used when defining I/O devices to an LPAR is very important, and the ability to alter these definitions should be controlled. Access to the HCD and HCM functions should be limited to Systems Programmers and Hardware Planners.

The PARTITION name corresponds to the LPARNAME keyword parameter in the LOADxx member. This can be used to identify a segment of the LOADxx and IEASYMxx members that relate to the specific LPAR running z/OS on the particular CPC identified with the HWNAME keyword.

One of the keywords in the above example, SHARED in the example below, allows the CHPID in question to be available to multiple LPARs. This opens up a possible exposure where this CHPID and its associated devices could be made available to a partition where the device is not required. In this example data held on these devices may not be secured by the ESM, leaving the potential for abuse.

The LPARNAME keyword and the equivalent definitions in the IODF ensure the separation of definitions within both of the files, ensuring that the correct devices are defined to LPARs, and that the correct information is used during the IPL process to bring these devices online for the partition.

Potential access to devices is stored in the CANDIDATE LIST, while devices defined to be used by an LPAR from IPL are listed in the ACCESS LIST.

It is possible to define a device as not available to an LPAR using the NOTPART keyword, where the Channel Subsystem and Partitions are explicitly defined as not having access to a device (see the example below).

```
CHPID PATH=(CSS(0,1),FC),SHARED,*  
PARTITION=(CSS(0),(BDEV,SS05,IBOC,SS15,SDEV,TDEV),(=))*  
,NOTPART=(CSS(1),(CFF1,CFF2),(=)),DESC='HYPERSOCKETS',*  
TYPE=IQD
```

5.1.7.3 VMUSERID

When running the z/OS operating system as a guest under VM, the system parameter files will contain sections linked to it using the userid of the z/VM system using the VMUSERID keyword.

When the VMUSERID keyword is being used, the HWNAME and LPARNAME keywords must also be specified, and they must represent the settings for the system that the IPL is being performed on.

```
SYSDEF HWNAME(processor-name)  
LPARNAME(lpar-name)  
VMUSERID(vm-userid)  
[SYSPARM(aa[bb...][L])]  
[SYSNAME(system-name)]  
[SYSCONE(system-clone)]  
[SYMDEF(&symbol='sub-text')]
```

This filter value links the sections of the LOADxx and IEASYSxx members to the particular VM systems, if the HWNAME matches the ProcessorID and the specified LPARNAME also matches the actual name of the Logical partition in which z/OS is running. This userid is defined to z/VM and a blank entry indicates that the image is not running under z/VM.

As with LPARNAME and HWNAME this filtering value can be used in the LOADxx and IEASYMxx members when performing an IPL, however the VMUSERID keyword is only valid until the next occurrence of HWNAME or LPARNAME. When either of these keywords is used the VMUSERID is reset to the default until the next use of the keyword is found. It should be checked that

the required values for each of these keywords is not reset to the default prior to the end of the settings and parameters for the system in question.

The use of these values for filtering and accessing sections of the relevant systems parameter files makes it is easier to ensure that the correct settings are being used when loading a system. A person must have the knowledge and ability to know the name of the CPC and the name of the logical partition as defined to the IODF, together with the userid allocated to the particular z/VM system.

All of the values allocated to these keywords are system and site specific, and when the LOADxx and IEASYMxx members are using them for filtering, they will affect the ability to load the system. Parameters specified in these members for the systems can be used to affect all aspects of the system, and therefore should only be altered by qualified personnel.

These values are not normally common knowledge, and any changes made to these values could cause the LPAR to fail to IPL. The filter values are contained in members in the PARMLIB libraries, and any access to these should be restricted. Any changes made to these values should be audited to ensure that any and all alterations have been approved and meet audit and internal standards.

5.1.8 Discovering parameters

It is always important in any audit to be able to provide evidence of compliance. Because of the dynamically changeable nature of z/OS it is impossible to get this information from the configuration files read at IPL time. I'm always worried when I see audit checklists which state "Check the contents of SYS1.PARMLIB". The site being audited may legitimately not even use SYS1.PARMLIB!

This means that there are a lot of z/OS parameters that must be interrogated in "real time". There are many ways to do this so as auditors of System z it will be very difficult to know how to gather the information in each environment. Most of the methods should be tied down by the external security manager. By far the best approach for us as auditors to get the information is to ask the Systems Programmers. They need this information in their day-to-day job and will be able to help.

It's not even like there is a single set of default ways to do things. For example, z/OS provides a number of parameters which can be interrogated from program function calls. A site may have developed a REXX or CLIST program or a SAS program or even an Assembler program to produce this data.

There are MVS commands which can provide the information as well as third party products which can do similar things. For example there is an MVS command "D IPLINFO" to show a number of parameter settings from the IPL but if the site has CA Sysview installed there is an alternative "IPLINFO" command which provides similar information in a bit more detail somewhat better formatted!

In the following sections I am going to discuss ways to find the information using only tools/commands provided by IBM. There are often easier/better ways to work and examples of products which can ease the pain can be found in the Appendices of this book.

5.1.8.1 Find NUCLSTxx

Of course you could start the hunt for NUCLSTxx by looking in the LOADxx

LOADxx member used for IPL and it's location

z/OS release

```

D IPLINFO
IEE254I 11.17.19 IPLINFO DISPLAY 350
SYSTEM IPLD AT 01.13.27 ON 11/06/2010
RELEASE z/OS 01.10.00 LICENSE = z/OS
USED LOAD08 IN SYS0.IPLPARM ON ACB2
ARCHLVL = 2 MTLSHARE = N
IEASYM LIST = (X6,U6,0L,R8)
IEASYS LIST = (ST,IN) (OP)
IODF DEVICE ACB2
IPL DEVICE 3C2A VOLUME D83EL
    
```

List of IEASYMxx members

List of IEASYMxx members

IPL volume used



member used to IPL and following through the chain from there manually. Or you could take the advice offered in most audit checklists for System z and look in SYS1.PARMLIB for LOADxx members and just hope that the site does not use an alternative PARMLIB chain.

Or you might prefer a simpler life with accurate information... in which case, you will need a tool (that interrogates the IPA control block built during the IPL process). For example NewEra's IODF Explorer - see appendices for further options.

IBM does not provide any simple way to obtain nucleus information from z/OS. Currently, the only route through to NUCLSTxx is as described in the first paragraph in this section.

5.1.8.2 Find IEANUCxx

The same problem exists with IEANUCxx. There is no simple way to gather the information and no IBM supplied alternative to starting with the LOADxx member used to IPL and working from there out.

5.1.8.3 Determine the MVS Level

It used to be quite hard to uncover the MVS level in a running system - way back when. Now though it is one of the easiest pieces of information to find. I like to start data gathering for an audit with the MVS level purely because it makes me feel positive about being able to find everything I need before I really get started.

Everything you need is in the output of the D IPLINFO command! See the diagram in 5.1.8...

5.1.8.4 Find the Master Catalog

The Master Catalog represents an index to all other cataloged datasets on the z/OS LPAR. An unexpected change in Master Catalog can prove catastrophic for the LPAR!

For example by creating a copy and changing all the volume serial number information contained, I could create a system pointing to all of the versions of programs or data that I wanted. I could change the program library of a production system to pick up my version of a program which had been written to send \$5 to my account every time a transaction completed.

The simplest way to find out which Master Catalog is being used is to check the SYSCAT keyword parameter in the LOADxx member of SYSn.IPLPARM used to IPL this LPAR. The Master Catalog is unique to the LPAR. Other members of the sysplex (i.e. those LPARs defined to be part of the same sysplex) may not use the same one.

To gather evidentiary data you should run a job which invokes IDCAMS on each LPAR being audited. You should already have established a relationship with the Systems Programming team so asking them for a job to do what you need should be easy. The JCL requirements may differ in each site but in general terms all you need to specify in the job is:

```
//LISTCAT JOB you'll need to know the format for the site
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LISTCAT CATALOG(MCat name from LOADxx SYSCAT keyword parm)
/*
```

5.1.8.5 PARMLIB Concatenation

If you find yourself looking at an audit checklist which tells you to look in

SYS1.PARMLIB you had better find out if that is **really** where the information lives before you rely on anything you find.

Fortunately there is another MVS command (D PARMLIB) and the output provides a full list of all libraries concatenated to the useable PARMLIB path. You should be aware that this will provide the details for a single, targeted LPAR. The command will need to be issued on each member of the sysplex to achieve a full picture of the PARMLIB concatenation across the plex.

Also listed are: the volume serial number that the library resides on; and where the entry came from (LOADxx, defined in the MASTER JCL, the result of a dynamic change).

D PARMLIB			
ENTRY	FLAGS	VOLUME	DATA SET
1	S	D72665	SYS1.PARMLIB1
2	S	CTDSD1	PARMLIB
3	S	D72665	SYS1.PARMLIB2
4	D	CATALOG	PARMLIB

Entry number 2 shows an entry for SYS1.PARMLIB but it has had a volume serial number specified which does not match that found by going through normal catalog processing.

The cataloged SYS1.PARMLIB library (4), which may (or may not) be the same as the library SYS1.PARMLIB on volume CTDSD1 (2), is automatically added to the end of the PARMLIB concatenation by default (it is not necessary to explicitly include it in the concatenation).

This may have been done deliberately, but if I came across it during an audit I would want an explanation from the Systems Programmers.

5.1.8.6 IEASYMxx Member(s)

System symbols are extremely useful to the folks who have to administer a sysplex. It is unusual to find a site with none defined at all. It is also common to find symbols used throughout the PARMLIB chain from deciding which members to load to the contents of keys system elements such as the LLA and LINKLIST. And whilst symbolics do not have a direct effect on the security of the environment, it is important to consider what would happen if someone changed one (or more) of the defined values.

A list of the IEASYMxx PARMLIB members used to create the full list of symbolics available on this LPAR can be found in the output from the D IPLINFO command. See the diagram in 5.1.8...

To obtain a full list of the available symbolics along with their values there is another MVS command to use - D SYMBOLS. The output of which will look something like this:

IEA007I STATIC SYSTEM SYMBOL VALUES 350	
&SYSCONE.	= "AC"
&SYSNAME.	= "TR1A"
&SYSPLEX.	= "TESTPLEX"
&SYSR1.	= "TI0RES"

5.1.8.7 IEASYSxx Member

Otherwise known as the System Parameter List, IEASYSxx provides the blueprint for which initialization parameters are to be used during the IPL process. The Systems Programmer can specify system parameters using a combination of



IEASYSxx PARMLIB members and operator responses to the SPECIFY SYSTEM PARAMETERS message issued during IPL.

Best practice suggests that the best place to put installation defaults or parameters that will not change from IPL to IPL is in IEASYS00. The Systems Programmer can add to or modify parameters in IEASYS00. The alternate IEASYSxx members, in contrast, should contain parameters that are subject to change.

Another method of ensuring that the correct IEASYSxx parameter member is used is to code the OPI=NO parameter into the member. This prevents **OP**erator **I**ntervention during the IPL process specifying an alternate IEASYSxx member.

Use of the IEASYS00 or IEASYSxx members can minimize operator intervention at IPL. Because IEASYS00 is read automatically, the operator can respond to SPECIFY SYSTEM PARAMETERS with ENTER or “U” and need not enter parameters unless an error occurs and prompting ensues. In fact, if the Systems Programmers have defined the system correctly it is possible to have an IPL that requires no input from the operators at all. The less manual intervention required during an IPL, the less likely it is that there will be problems.

A list of the PARMLIB members used to create the merged IEASYSxx on this LPAR can be found in the output from the D IPLINFO command. See the diagram in 5.1.8...

5.1.8.7.1 Parameters

I want to take you back to those punched card oriented “MVS rules” for a moment. Because there are some very rigid rules when it comes to what entries can be placed in the IEASYSxx member.

Each record in IEASYSxx consists of 80 columns. In a slight alteration from the rules as they applied to the LOADxx member, data can only be held between column 1 and 71. Columns 72 thru 80 are ignored and do not **need** to be blank.

Blank lines can be used in comments but only after the last statement in the record. Otherwise, do not use blank lines within record specifications. It will lead to anything after the blank line being completely ignored!

Leading blanks in a record are acceptable in IEASYSxx which means parameters do not need to start in column 1.

Record continuation is indicated with a comma followed by at least one blank.

Commas can also be used to separate multiple, short parameters on one line, but do not leave blanks between commas and subsequent parameters.

The system considers the first line that does not end in a comma to be the end of the member and ignores subsequent lines. The remainder of the line, which contains the last parameter, can be used for comments, providing there is at least one blank between the last parameter and the comments. Additional lines can be used after the last parameter for comments. This is the only way that comments can be specified in IEASYSxx.

Enclose multiple sub-parameters in parentheses. The number of sub-parameters is not limited.

And finally in the exhaustive list of rules, the part of the system which reads IEASYSxx does not recognize lower case letters. The whole content must be in upper case.

Despite these incredibly strict rules, there is no validation method supplied by IBM. Problems can occur very easily. I have managed to get it wrong quite catastrophi-



cally myself. One that I particularly remember was when we were getting ready to implement parallel sysplex and were adding an additional character to the high level qualifier of all of our JES2 datasets - this was just part of a wider project.

All the prep work had been done so I made the change to IEASYS00. I didn't bother taking a backup first because it was just a change to the JES2 spool dataset. Sadly, the extra character in the high level qualifier meant that I overtyped the comma at the end of the line. When I IPLed everything past the JES2 spool was ignored. We ended up with a system that just wouldn't start up, i.e. an accidental Denial Of Service!

Some parameters in IEASYSxx (like CLOCK, CON, and MLPA) allow specification of the list option (L). If you specify the L option, and message suppression is not active, the system writes all of the statements read from the associated parmlib member to the operator's console during system initialization. If message suppression is active (the default), the system writes the list to the console log only.

This is a good thing to do as it allows the auditor to trace back what happened at the last IPL. It's pretty difficult to find some of this information so having it on the console log would be quite helpful. So to ensure that messages are suppressed and not deleted, get your Systems Programmer to specify an appropriate initialization message suppression indicator (IMSI character) on the LOAD parameter.

5.1.8.7.2 Directors

The PARMLIB concatenation contains both a basic general parameter list (IEASYS00) and possible alternate IEASYSxx members. It also contains other, specialized members, such as COMMNDxx and IEALPxx.

Any IEASYSxx member can contain both parameter values and "directors". The directors (e.g. MLPA=01) point or direct the system to one or more specialized members (in this example it would be IEALPA01).

Where multiple members are to be available for use, the suffix for these members can form a list in the parameter. One example of this is for the parameter that is used to ensure the correct LPALST members are used. In the IEASYSxx member the member to be used is specified using the LPA=(aa,bb) statement.

In this statement aa and bb can be ANY character, and these characters are appended to form the LPALSTxx member to be used. The order that they appear in specifies the concatenation order that the members are used.

Specifying a director in IEASYSxx works a little like including a copy book into a program. It allows an additional, specialized set of system initialization parameters to be incorporated into the general parameter list used for IPL.

One advantage of separating these specialist parameters out is that security on them can be handled differently. For example, some of the specialist members pointed to by directors in IEASYSxx deal with storage initialization parameters. By placing these members into a separate library in the PARMLIB concatenation, different security rules can be applied. This can ensure that only storage professionals will be able to alter storage related initialization parameters.

5.1.9 Prevailing PARMLIB Members

z/OS can cope with up to 17 libraries in the PARMLIB concatenation. They can come from: the PARMLIB keyword in LOADxx; in the Master JCL; within various IEASYSxx members; or from a SYSP parameter specification during IPL. Effectively the system sees PARMLIB as one big pool of members which is created by reading the list from the bottom up.

This works just fine when there are unique names for all the members in the concatenation. But there is nothing that can be done to stop a same name member being created in a library higher up in the chain. In fact, Systems Programmers use this quite legitimately to perform system maintenance. Under these circumstances the first member with duplicate names will be used and the others will be discarded.

The final part of the equation that can affect the PARMLIB concatenation is that a new one can be implemented at any time by issuing one of the available MVS Operator Commands that can be used to dynamically change the member in use by z/OS.

It is always vitally important to get current information about an LPAR's PARMLIB concatenation. Manually following the chain from LOADxx and onwards simply can't be guaranteed to leave you with the correct information. The MVS operator command "D PARMLIB" can be used to provide the up to the minute concatenation list - including the order.

The list of PARMLIB members that will pose risk is individual to each installation. There are Best Practice guidelines but the highly configurable nature of z/OS and the zEnterprise means that these are general at best. It is vital to perform an individual Risk Assessment for the PARMLIB concatenation contents for each sysplex.

5.1.9.1 High Risk

Bizarrely, one of the highest risks associated with managing PARMLIB can be posed by providing the Systems Programmers with **insufficient** access to be able to do their day-to-day work! Most audit programs suggest that Systems Programmers are akin to spawns of the devil and that they should be locked out at all costs. It is true that these guys could bring down the system that would be somewhat counterproductive to their continued employment in the industry. And most of them really enjoy their jobs!

The risk comes about when this valuable team has been locked out of the operating system to such an extent that they can no longer fix it when things go wrong - and they will go wrong!

It is far better to allow the Systems Programmers the ability to update the PARMLIB libraries for which they have responsibility (use separate libraries to control the separation of duties). Monitor everything! Monitor the Systems Programmers on an individual basis (Userid Audit). Monitor **every** access attempt to all of the PARMLIB libraries.

Don't let anyone persuade you that looking at failed access attempts is of any real value. All that will show you is the external security manager doing its job and rejecting unauthorized access attempts. The real value is in closely monitoring successful access to these resources. This can help to identify who changed something and when giving the Systems Programmers a fighting chance of being able to fix the problem caused.

Another of the high level risks comes to you courtesy of 5 decades of weird and whacky uses that customers have found for their mainframes. Back when the world was first introduced to the concept of large data crunching computers life was simpler. The computer was a big thing that had its own environmentally controlled room. There were no network issues - there were no networks! The term "Hacker" was used to refer to an excellent programmer who could make the mainframe do what was needed!



Security used to primarily be a matter of making sure that data couldn't be accidentally lost or damaged. This led to the majority of IBM mainframe installations defining security with READ to all users capability (in RACF it's referred to as UACC(READ)).

It is still important to make sure that data cannot be lost or damaged but it means that there is a prevailing attitude amongst z/Security Analysts that UACC(READ) is acceptable. The world we live in now doesn't agree. All of the latest research points to the majority of hacking in the 21st century being an "inside job". Grabbing copies of customer data files to sell to the highest bidder is not unusual anymore. It's sad, but true.

One of the side effects of the "Read is OK" philosophy is that if you can read data you can copy it. An id with update access to a PARMLIB library concatenated above the more secure one that the Systems Programmer wants to "mess with", would enable a modified copy of the real member to be inserted into the PARMLIB library they have update access to.

Another aspect of allowing users READ access to members is that this does not stop an altered task being submitted for processing. A member could be edited in ISPF, run through the system without needing to save the member. There is the possibility that this could be done without any evidence of wrong-doing being detected up by the ESM.

If the copy member were to be placed in the library that the id does have access to it would take priority over the **real** one. No security alerts would trigger because no one has tried to gain access that they weren't allowed. I could make any number of changes this way; e.g. turn off SMF recording for audit records!

The only way to be truly certain that such situations can be back traced is to audit **all** activity to **all** PARMLIB concatenated libraries. This provides the excellent benefit of making the system more auditable due to lessening of the use of break glass ids. Everyone will be happy.

Members of PARMLIB that have been identified by an installation should be kept in a separate library that can have tighter controls than most. And whilst there are installation specific variations, treating the following list as potential high risk level data is advisable:

BPXPRMxx	IEASVCxx	LOADxx
COMMNDxx	IEASYMxx	NUCLSTxx
CONSOLxx	IEASYSxx	PROGxx
COUPLExx	IEALPAxx	SCHEDxx
IEACMDxx	IEFSSNxx	SMFPRMxx

5.1.9.2 Low Risk

To be quite honest, there are no members of PARMLIB that can be considered as carrying a low risk level. All of them are used to define the shape of the z/OS operating system and various subsystems or functions.

I suppose that those which pose least risk are the members which are used to dynamically configure specific functions. The best example I can think of is IEASLPxx which controls the way that dumps are taken on the system. Typically the values only really matter when debugging a problem and they would be tailored specifically for each individual type of problem. It's quite unlikely that SVC dump processing changes could cause major problems on z/OS.

One exception to that would be if I were to be interested in causing a denial of service. If I changed IEASLPxx to put out more information to the dump than



usual, I could flood the I/O processing capacity by repeatedly running a program I knew would fail. But there are much easier ways to crash z/OS so I can't think of a situation where I might have to do so.

5.1.10 Dynamic Configuration Changes

z/OS and the zEnterprise platform is one of the most configurable hardware and operating systems combinations available. Any changes to the hardware configuration and allocated resources for an LPAR can be made dynamically, allowing the system to be tailored to react to immediate business needs.

Even after the operating system has been loaded and the system is in use, additional processing power can be dynamically moved from a less active Logical Partition. This means that should the load on a particular partition increase beyond a predefined threshold, response times need not be affected as additional power can be allocated to meet the need.

Think about the Victoria's Secret live catwalk web-event debacle a few years back! If they had been running the services on z/OS their reputation would not have been damaged by too many customers logging on causing an accidental denial of service!

Changes can be made to the IODF work datasets to add or remove devices from the I/O configuration. If this work IODF is then built as a production IODF it can be activated. This means that an IODF entry does not have to be in place prior to the partition being IPLed to add hardware devices. An alternate production IODF can be activated on an LPAR using the ACTIVATE IODF command.

z/OS also allows devices to be varied on and offline from the operating system configuration (provided of course that a corresponding IODF entry for the device on that LPAR is in place).

Under z/OS dynamic changes can range from increasing the priority of a task on the processor to allow it additional resources, to the actual varying OFFLINE of devices to the operating system. While an I/O configuration will still exist for a device that has been varied offline, if the device or paths to the device are varied offline, no I/O can be completed to it.

Exactly the opposite can also occur, where devices can be added to the IODF, this IODF activated and the device brought online to the operating system. All of these tasks can be completed while the operating system is running.

If changes are made dynamically to the I/O configuration these changes will only be temporary. They will be reversed at the next IPL. If these changes are to remain in place after the operating system is IPLed, the IODF should be updated. Changes must be also made to the load parameters and LOADxx member to ensure the new IODF is used, allowing the addition or removal of device definitions to be reflected in the LPAR definition post IPL.

This ability to manipulate the system configuration on-the-fly introduces the potential for audit and service issues. While any configuration changes will be reflected in the console log these are often only reviewed retrospectively, after an issue has occurred.

The dynamic addition or removal of an I/O device to an LPAR, or CPC's configuration could go unnoticed when it first occurs. If it is not actually causing a problem to the system this may not be picked up, and without making the change permanent by updating the load parameters, the next IPL could remove any trace of this activity.

The ability to make these dynamic changes forms part of the normal operations on

a system, but making them should be limited to staff who require these functions to complete their role. Any use of commands, and access to tools that allow these dynamic changes to be made should be fully audited to ensure that no misuse occurs.

This should include, but not be limited to, access to the HCD/HCM, as well as the ability to issue console commands such as VARY and SET. These commands are detailed in the next section.

5.1.10.1 MVS Operator Commands

z/OS is delivered with a number of significant operator commands which can be used to dynamically change the shape of the system. It can be quite difficult to track down all of the commands which can be used in this way because they come from a number of different subsystems.

The following table breaks the commands down into MVS commands and JESx (either JES2 or JES3 depending on the installation) commands. It details the command as usually issued at the console, the associated OPERCMDS external security profiles required to control the command and what impact the system would see. And finally there is an assessment of the risk level associated with the command.

Many audit checklists suggest that OPERCMDS should be protected which is good. But what they normally propose is a generic profile at the top level of the command - e.g. MVS.** would protect all of the MVS commands **but** it does not differentiate between low and high risk commands.

The most critical of the commands are listed in the following table. It is not an exhaustive list of all operator commands available on z/OS but there should be protection on these at the very least.

MVS Command	Associated OPERCMDS resource profile	Affects	Risk level
ACTIVATE	MVS.ACTIVATE	Activate new IODF	High
CONFIG	MVS.CONFIG	Configure devices, channels, etc. on and offline	High
CF CHP(xx), ONLINE	MVS.CONFIG	Place a CHPID online	High
K	MVS.CONTROL	Change displays, log routing etc.	Med
C	MVS.CANCEL.type.name	Cancel address space e.g. user, job	Med
DISPLAY	MVS.DISPLAY	Display device characteristics and status	Low
DUMP	MVS.DUMP	Display and/or clear dump datasets	Med
FORCE	MVS.FORCE.**	Force task/device	High
F	MVS.MODIFY.**	Pass info to job/task	High
QUIESCE	MVS QUIESCE	Quiesce z/OS	High
R	MVS.REPLY	Reply to WTOR messages	Med



MVS Command	Associated OPERCMDS resource profile	Affects	Risk level
E	MVS.RESET	Alter task e.g. performance group	Med
RESTART	MVS.RESTART	Resume after quiesce	High
T	MVS.SET.**	Dynamic changes to information held in storage	High
T PROG=xx	MVS.SET.PROG	Update system settings such as APF list, LNKLST, etc.	High
T SMF=xx	MVS.SET.SMF	Change SMF recording characteristics	High
I SMF	MVS.SWITCH.**	Switch SMF recording	High
V CN xxx, AUTH=xxx	MVS.VARYAUTH.CN	Alter Console Authority levels	High
V CN xxx, AUTH=MASTER	MVS.VARY.MSTCONS	Vary an alternative master console	High
V SYSLOG, HARDCOPY	MVS.VARY.HARDCOPY	Change destination to hard-copy logs.	Med
V xxx,OFFLINE, FORCE	MVS.VARYFORCE.DEV	Force Device Offline	High
V XCF,xxxx,OFF	MVS.VARY.XCF	Vary Coupling Facility	High
WRITELO	MVS.WRITELOG	Direct control over z/OS syslog	Med

JESx Command	Associated OPERCMDS resource profile	Affects	Risk level
\$B	JESx.BACKSP.DEV	Backspace a device	Low
\$C	JESx.CANCEL.**	Cancel a job/device output	High
\$T	JESx.DISPLAY.resource	Display resource characteristics	Low
\$D	JESx.DISPLAY.**	Display job, initiator	Low
\$Z	JESx.HALT.**	Halt device, initiator, spool	High
\$I	JESx.INTERRUPT.DEV	Interrupt device processing	Med
\$T	JESx.MODIFY.**	Alter a task	Med
\$O	JESx.RELEASE.**	Control output	Med
\$N	JESx.REPEAT.**	Repeat spooled output to a device	Med
\$E	JESx.RESTART.**	Restart a job, output task	High
\$S	JESx.START.**	Start a device, initiator, etc.	High
\$P	JESx.STOP.**	Stop a task, initiator, device, etc.	High

In order to keep control over who can issue operator commands from the console(s) it is vital that enforced signon to consoles is implemented. It is also critical that successful use of the commands is audited rather than just unsuccessful attempts. Unsuccessful means that the external security manager is doing its job well. Successful access to the command means that someone has changed the system!

The first step towards controlling access to these operator commands is by ensuring that only authorised persons have physical access to the consoles in the first place. If they can't get to the terminal the commands can't be issued. Once the console has been physically secured access for individual users is controlled via entries in the CONSOLxx member in PARMLIB, ensuring that users have to sign on to the console prior to issuing commands.

The successful issuing of these commands should also be captured for audit and review purposes to ensure that no commands have had an adverse effect on system integrity and compliance.





6 And Finally . . .

I'd like to say a big Thank You for sticking with me to the end. It's been a wild ride trying to de-mystify the risks associated with 21st Century use of IBM mainframes and **at the same time** offer direction on what you actually need to do. I hope that the resounding image that you are left with is that these environments represent the most securable platform commercially available but that they are not delivered that way. And that usage of this incredibly flexible platform has evolved to **way beyond** what is currently being catered for in most audits.

We've moved into a space that needs careful re-examination. And, because it is such a technical environment, bonds need to be forged between the Auditor(s) and the technical teams like Systems Programming and Hardware Management. Following this simple model won't lead to check lists which can be shared industry wide but it will lead to appropriate audits being in place given the specific workloads of the individual Customer.

I hope that you have found the book useful. I certainly found writing some of my long held beliefs down and seeking opinion from other industry specialists a valuable exercise.

When I first started in this industry, the Security Team was, if not actually feared then certainly, regarded with respect. We've lost a lot of that over the years as people started to see security as blocking the business. But now we have a spectacular opportunity to reclaim our natural position as the enabler of business. Working in partnership with the technical teams we can provide a highly visible service in hot topic areas like Identity Fraud and Hacking.

Good luck with your next steps and I hope you join me for the second book in this series (in which I will discuss the z/OS Operating System). The information in this book can be added to your audit toolkit, helping to ensure that you have a solid and secure hardware base to start from ☺

You also might like to add NewEra Software's free utility StepOne to that tool kit. This neat little application allows you to see exactly what's going on inside the IODF that YOU are interested in. And it does all of that whilst only requiring read access to the data.



7 What's Next?

I believe that it would be cruel and unusual punishment for me to share all of these potential problems surrounding zEnterprise configuration change management and Front Doors with you and then not give any idea of where to go next. So this final chapter is here to give me the opportunity to share what could become your starting point for all future audits of System z.

Now, remember that there are alternates on the market but what I want to talk about, albeit briefly, is StepOne. This tool is supplied for free and was written and is supported by NewEra Software, Inc.

What's unique about it is that Paul Robichaux (CEO and co-founder of NewEra Software, Inc.) and I worked hand in hand on StepOne. The product was written as a companion to this book and allows you, the Auditor, to step through the definitions hidden away in the IODF and the rest of the logical partition IPL process.

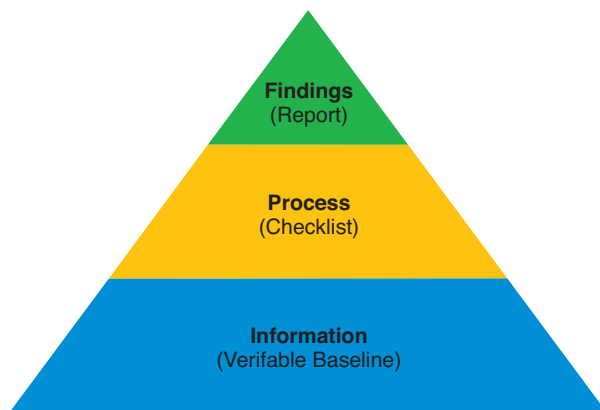
7.1 A New Problem?

No, not really. It's a problem which has been around since mainframes started to get Audited but we've always been able to "ignore" it in the past. New functions and features of zEnterprise mean that it is vital to revisit attitudes to Configuration Change Management and how it impacts the audit compliance of the virtualized environments supported.

As Robichaux commented recently:

"...the conventional wisdom of too many Audit Plans and Tools ignore the obvious and begin deep in the details of the Operating System (OS) and External Security Manager (ESM).

In doing so, these Plans and Tools often fail to establish an independently verifiable System Baseline. Without such a repository of system identity and configuration relationships, zEnterprise System Auditors can become disoriented, losing their way in a world represented to them only by a seemingly endless set of Audit Checklists."



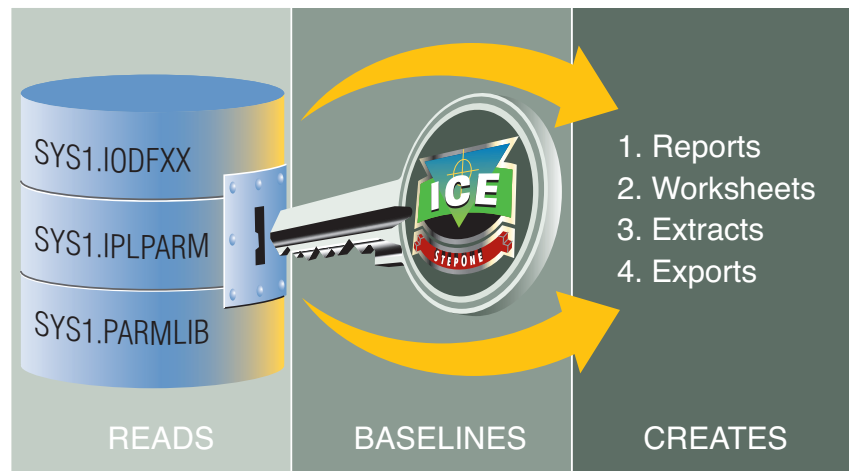
Take a look at the diagram above.

The typical checklists we currently use to audit System z do not address all of the areas which need to be secured. In fact most of the audits performed on System z don't even think about this wider picture. In part, that can be attributed to how difficult it can be to gather the data and somehow baseline it. So we end

what's next?

up not looking under the covers and data security ends up taking a back seat to compliance with various regulations.

7.2 A New Solution - StepOne



zEnterprise Hardware and Software Configurations

StepOne is a zEnterprise based application that unlocks key System Datasets turning their contents into an interactive set of zEnterprise wide system documentation designed to enhance the System Review Process initiated by System Auditors and Consultants that conduct them.

It is not a tool that is designed to replace the zEnterprise Systems Programmer. What it can do is help you to speak the same language as these highly specialized technicians. And with Auditors working alongside Systems Programmers, StepOne can help to identify the risk points in this complex, virtualized environment.

StepOne runs on the zEnterprise (not on a server somewhere running against copies of security data) providing the following advantages:

1. Data Access and Data Currency
Running StepOne on the mainframe means that real, live data can be examined without the need to coordinate offloads to other servers.
2. Chain of custody unquestioned
Removing the need to transfer zEnterprise data to other platforms means that there is no way for the data to be affected in transit.
3. Clear Text Download
StepOne is provided free of charge by NewEra Software Inc. as a clear text REXX program.

Using StepOne you'll explore the DNA of the most powerful Central Processing Complex (CPC) available to your business and governmental clients. The interactive native environment used by StepOne allows you to dynamically and automatically create a zEnterprise Configuration Baseline.

Think of this Configuration Baseline as a platform of knowledge upon which you can base ALL more detailed system investigation and remediation. At your fingertips are the specifics of all hardware configurations used for Power on Reset (POR) and operating system configurations used for Initial Program Loads (IPL).

For the first time you'll enjoy the advantages of seeing the zEnterprise Fabric from a low level security perspective. The significance of each individual configuration

component and its potential impact on the overall integrity of each Logical Partition (LPAR), the primary control boundary of the zEnterprise, becomes clear.

To understand the zEnterprise you need to envision the use and function of its resources. That is the processors, channel paths, controllers and devices and how those resources are connected, each to the other. All of which forms the "Fabric" of interconnectivity for each Logical Partition (from both the perspective of resource limits and the potential for resource sharing). To accomplish this, StepOne links you directly to the various Configuration Control Programs (CP) that reside at the heart of the system in the Input/Output Definition File (IODF), the Absolute zEnterprise Control Point.

When you logon, full screen interactive worksheets, with lots of supporting help, assist you through the complex picture outlined in this book. Collectively these worksheets let you start with any processor drilling down into the lowest level of detail. How low you ask? How does a level where you can match UCWs and UCBs to isolate device orphans sound? Don't know about those? Don't worry, it is all explained in the documentation.

Each Operating System Configuration Program (OSCP) is identified by a unique Configuration Identifier. This "CONFIGID" is called by name from the IODF Keyword that appears in the LOADxx Member used during the Initial Program Load (IPL) of an LPAR. With StepOne the time consuming, but required, task of matching CONFIGIDs and LOADxx Members in order to identify configuration orphans is fast, efficient, and totally automated.

One of the principal advantages of any Logical Partition (LPAR) is its ability to share the common resources of the zEnterprise Fabric with other LPARs within the same CPC or other CPCs. Such sharing, common in a Sysplex environment (meaning multiple LPARs acting as a single processing unit), can open the door to exploitation when access to critical data is accomplished along a common path or isolated to a common controller or device.

For example a Test partition connected to Production data would be outside the scope of a typical SoX audit but would immediately negate the validity of any such audit performed only at the LPAR/security database level. Using integrated baseline and compare functions, StepOne isolates permitted access, exposing potential configuration weaknesses.

Once uploaded to the zEnterprise, StepOne is run directly from the z/OS Time Sharing Option (TSO) Command Shell where it "sits very lightly" on the system, automatically cleaning up after every time it is used. In fact, you must take some positive action (such as saving reports produced out to DASD) for it to be apparent that StepOne has been run at all.

I urge you to consider incorporating StepOne into your System z Audit tool kit. Where else are you going to find a product that allows you to expose weaknesses in areas that you hadn't even previously thought of?

www.newera.com/stepone/product_description.pdf



8 References

¹ Gibson and Nolan's Managing the Four Stages of EDP Growth published in 1973, the same year that IBM first issued the Statement of Integrity for the mainframe.

² A copy of the "IT Doesn't Matter" article is freely available on Mr. Carr's blog:
www.routhtype.com/archives/2007/01/it_doesnt_matte.php

³ Going Green with IBM Systems Director Active Energy Manager:
www.redbooks.ibm.com/redpapers/pdfs/redp4361.pdf



9 Glossary of Terms

This will be more than just a z/OS glossary from an IBM manual. It is a collection of basic z/OS and auditing terms that you should be familiar with.

4LA / FLA - Four Letter Acronyms.

ACID – CA Top Secret term representing userid

ACS routine - A feature of SMS used to control and manage dataset creation based on a set of site defined rules.

Address space - A virtual space created and maintained by z/OS in which separate tasks run. For example a batch job would run in separate address spaces. Each address space has a unique id called the ASID (Address Space ID, not to be confused with ACID which is often pronounced the same way).

Anchor point - A term indicating a technical IT element which can be used to provide a fixed base for any audit i.e. a really critical IT element.

API - Application Program Interface, a documented interface point that allows applications or applications and users to interact in a controlled manner.

Batch job - Aka Job. A set of JCL instructions, including the execution of at least one program, that performs a unit of work as a separate task to that which created it. Batch jobs are submitted to JES which uses the supplied JCL to determine where, when and what runs. A single user may submit many batch jobs which may or may not run concurrently depending on the requirements. Batch jobs are used for short running tasks and are executed by JES in a special sub-set of address spaces called Initiators which are under the control of JES.

BCP - The Base Control Program provides all of the base functionality for the z/OS platform. It provides the foundations for all of the functions that run under z/OS, dealing with elements such as Workload Manager and the System Management Facility. Without BCP the operating system would not function.

CA ACF2 - An external security manager product from Computer Associates.

CA Top Secret - An external security manager product from Computer Associates.

Channel Path - The logical path used to describe the physical route data takes from the mainframe through a channel cable to a physical device.

CHPID - CHannel Path IDentifier, Traditionally a unique logical 2 byte identifier allocated to each channel path that represented a logical channel port on the mainframe with the same value. Since z10 this is a purely logical identifier that is unique within an LCSS. The physical aspect is now covered by PCHPID. CHPIDs are often used by Systems Programmers to automate system operations.

CIB - Coupling over InfiniBand, The latest high

speed channel architecture supported by system z which uses the InfiniBand protocol.

CICS - Customer Information Control System, IBM's premier application environment offering.

CLIST - Basic scripting language found on z/OS that is often used by Systems Programmers to automate system operations tasks. Used less frequently than in older releases of MVS in favor of Rexx.

CMS - Conversational Monitor System. An application and interactive user environment found on VM LPARs. It performs a role similar to that which TSO does under z/OS.

Compliance - operational transparency that results in organizations adopting the use of consolidated and harmonized sets of compliance controls in order to ensure that all necessary governance requirements can be met without the unnecessary duplication of effort and activity.

Coupling Facility aka **CF** - A specialized LPAR that does not require an OS whose only role is facilitate the controlled sharing of data between LPARs within a sysplex.

CPC - Central Processing Complex. Another name for a collection of processors, sometimes called a BOOK, forming the heart of the Mainframe.

CSYSTEM - Connected SYSTEM. A connected system represents a processor to which an InfiniBand CHPID may connect.

DASD - Direct Access Storage Device. Traditionally a DASD unit related to a single physical drive unit. With the increase in drive capacity a single physical drive unit may contain multiple DASD units. Each DASD unit contains a single z/OS DASD Volume. These volumes are used to contain the z/OS file systems that contain all of the z/OS datasets. A DASD unit may be allocated to a single LPAR or shared by several.

Dataset - The term used to describe files within the z/OS file structure. The term File may sometimes be used in its place. Each dataset is defined with a specific organization or format which is used to determine which access method is used to access data held within the dataset.

Detective – the real-time identification of actions processes or configuration change issues that may cause audit failure points to be flagged.

EAV – Extended Address Volume enables larger disk volumes to be available to z/OS environments. This support is provided by DFSMS.

ESCON - Enterprise Systems CONnection, The first Fiber Optic based channel connection architecture for mainframes. It has been superseded by the newer and faster FICON and InfiniBand connections.

ESM - External Security Manager, the product used to control access to z/OS resources. See RACF, CA-TOP SECRET, CA-ACF2.

Exit - aka Exit Point. An interface within z/OS or other IBM and ISV system software and the associated code. Normally written in Hi-Level Assembler. To be treated with caution as they can be used to alter the way in which any activity occurs. Some of these exits are well documented, however many are not.

FICON - Fiber CONnectivity, The channel connection architecture that replaced ESCON. Faster than ESCON channels but slower than InfiniBand channels.

Green Initiative – an IBM policy forming part of the Smarter Systems for a Smarter Planet initiative which aims to lower the environmental impact of running z hardware. This can be measured using carbon output, power requirements and impact of building the hardware.

HSA - Hardware System Area, A fenced off section of mainframe storage used by the mainframe itself which varies in size based on the contents of the IODF. Pre z10 the amount used came from the memory paid for by the customer.

HMC - Hardware Management Console. The PC that controls a mainframe. Also known as the Support Element by the hardware folks.

Hypervisor - A software layer that allows multiple operating systems to share the same physical hardware. For System z this role is performed by z/VM and/or PR/SM.

IBM Health Checker – an IBM product that is used to identify potential configuration issues that may impact system integrity and availability.

IMAGE - aka z/OS image. Can be used to describe any z/OS LPAR but technically it indicates a z/OS LPAR that is a member of a Parallel Sysplex. Specifically where each LPAR in the SYSPLEX is an image of the others.

InfiniBand - Newer technology/protocol which enables a Storage Area Network type structure to be implemented in the System z environment. Seen as the future of I/O connectivity on System z.

Integrity Model – a formal state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity.

IOCDS - I/O Configuration DataSet, It represents the hardware part of the configuration envelope

IOCP - I/O Configuration Program also used to describe the output or “deck” produced by HCD using IOCP.

IODF - I/O Definition File, The file containing the hardware blueprint and how all of the LPARs connect to it. It is used with the IOCDS by IOCP. The IODF contains 3 configuration elements: the IOCP, OSCP and SWCP. Each element may in turn contain many unique operational entities. In the case of the IOCP the elements are Processors or PROCIDs. For the OSCP

the elements are Configurations or ConfigID and for the SWCP, Switch Configurations or SWIDs.

IPL - Initial Program Load, The start-up process for a system z/OS where every part of the environment is loaded from scratch. Use of dynamic changes has reduced the need for IPLs which in turn has increased the risk associated with an IPL, unless these dynamic changes are correctly reflected in locations used by the load process.

IRD - Intelligent Resource Director, A system z feature that expands the role of dynamic workload management out into the physical hardware layer. It can manage your LPAR CPU and Channel Path workloads plus the Channel Subsystem Priority I/O Queuing workloads on a goal-orientated basis.

ISPF - Interactive System Productivity Facility, An application environment that makes TSO usable by humans.

JCL - Job Control Language. A set of instructions which when combined into a JCL stream describe an environment in which programs may be executed. STCs use a subset of the JCL instruction set whilst batch jobs have access to the entire instruction set.

Back in the days when MVS was first introduced, the only real method of entering data into the system was punched cards. JCL has not radically changed since those days. It is still an 80 character line length representing the maximum that would fit on a standard IBM punched card. For this reason it is often referred to as a JCL Deck by the Grey Haired Gurus aka sysprogs.

JES2 / JES3 - Software based resource management layer found on z/OS systems. A z/OS image will use JES2 or JES3 never both. Amongst its many responsibilities is output management. JES2 is more commonly installed but JES3 still has a steady user base.

LCSS - Logical Channel SubSystem, The fixed allocation of physical hardware channels into logical groups. Introduced with the z10, LCSS is the method used by IBM to increase the maximum number of supportable LPARs within a single mainframe past certain architectural limits.

Logstream - A relatively new method of storing large amounts of data, controlled by the system logger task, such as activity records (SMF) that a modern sysplex can generate. Although not suitable for all data it represents an optimized I/O path and improved granularity for key system activity.

LPAR - Logic Partition, A mainframe concept used to allocate resources such as CPU (both in number and percentage terms), physical memory or DASD units to a specific operating system instance. Each LPAR is ring fenced at the CPU and memory level to provide data integrity.

LSYSTEM - Logical/Local System Name, an

alternative 8 character name that may be assigned to the IODF processor definition. Used in conjunction with InfiniBand CHPIDs for channel to channel communication.

MSU - a million service units is a measurement of the amount of processing work a single mainframe can perform in one hour. Many ISVs now charge by MSUs consumed (aka "workload-based charging") rather than the older, MIP rating for a specific processor.

MVS - A term used by the older generation of IT staff instead of z/OS.

NAIC - Insurance industry compliancy regulation. North America in origin.

NIST - National Institute of Standards and Technology. North American in origin.

Object Map - The picture which can be drawn to show the connection between technical IT elements and audit/business control objectives.

OSCP - Operating System Control Program defined using HCD and resides within the IODF. Formerly known as MVSCP.

Parallel Sysplex - A hardware and software management layer that allows multiple LPARs running on multiple mainframes to be combined into a single service. A correctly configured Sysplex offers 24x7x52 availability with hardware and software fault tolerance plus dynamic workload balancing and dynamic increases in processing capacity.

PARMLIB - A z/OS dataset that contains a set of PARMLIB members used to configure system parameters and values. This can be a reference to a dataset used by a single product or the chain of datasets used to flesh out a z/OS image.

Partition - See LPAR

PCHID - An IODF value that identifies a physical channel port to the z hardware. This is then logically mapped to a CHPID which the host OS then communicates with. This mapping, when configured correctly, can provide balanced communications with no single point of failure.

PCI - IT Compliancy regulation. European in origin.

POSIX - Portable Operating System Interface for UNIX, a set of standards that define various aspects of the UNIX operating system. USS is fully POSIX compliant.

Preventive - an action or series of actions that prohibits or mitigates the possibility of an event or series of events.

PR/SM - A hypervisor feature available on mainframes that enables multiple LPARs to share the same hardware.

PROCID - Processor ID. An 8 character field used to describe the processor that an IODF element can be used on.

RACF - Resource Access Control Facility, the IBM external security manager product. Market leader amongst the External Security Managers.

Remediation - acting on any audit findings to ensure that any actions that would have normally caused failures are no longer able to be completed by unauthorized persons or processes.

REXX - A very powerful scripting language available on z/OS (and other platforms including the Open Source community). Modified versions of REXX can also be found in certain IBM and ISV products that provide a base REXX environment with additional product specific functionality e.g. CA-OPS/MVS OPS/REXX.

SAF - System Authorization Facility, A term used to describe the process used to validate an access request to a resource controlled by the ESM.

Side Car/Cage - the separate cabinet coupled to zEnterprise that contains the zBX Blade-Center Extension.

SAS7.0 - IT Compliancy regulation. North American in origin.

SMF - System Management Facility, The underlying system activity audit trail for z/OS. SMF can be configured to cut information records detailing much of the activity that occurs on z/OS. For example RACF can be configured to generate an SMF record when access to a resource is denied. These records are written out to datasets that are normally backed up and retained for audit processing.

SMFID - The name assigned to an individual LPAR, must be unique within the sysplex that is used to match activity recorded by the Systems Management Function (SMF) to that LPAR. As a required system symbol it will be referred to in all sorts of places.

SMS - System Managed Space, A z/OS feature that can be used to manage use of your DASD space in conjunction with an ACS routine.

SNA - a network protocol used by VTAM. This includes the LU 2 and 6.2 protocols.

SOX - aka Sarbanes Oxley. IT Compliancy regulation. North American in origin but affects anyone who does business with North America too.

STACK - all of the software installed on a defined LPAR. This can include in-house applications written by the customer. Can also be used to refer to a specific set of software e.g. the TCP/IP stack.

STC - Started Task. STCs are defined using JCL and controlled by z/OS and JES in a similar manner to batch jobs. The key differences are that a batch job is designed to perform a single discreet work process and then complete where as an STC is designed to process requests on an ongoing basis. Most STCs also interact with other tasks or users to perform designated tasks.

Switch Fabric - A concept from the InfiniBand protocol dealing with how traffic is handled within an InfiniBand network.

Symbolics - aka System Symbols, The ability to

use a symbol such as &SMFID which the operating system will resolve at runtime into a pre-set value. Useable in many areas of z/OS they can simplify many system management tasks. Caution there is such a thing as too many symbols.

Sysplex - See Parallel Sysplex

Sysplex Timer aka **ETR** - A really accurate, required, clock used to synchronize the activity that occurs across a parallel sysplex.

System logger - A relatively new method of storing large amounts of data, controlled by the system logger task, such as activity records (SMF) that a modern sysplex can generate. Although not suitable for all data it represents an optimized I/O path for quicker response and improved granularity for key system activity.

System z - Term used to describe both the IBM mainframe hardware and software environments.

Systems Programmer - aka Sysprog, An endangered species of technical specialists responsible for maintaining the z/OS software stack. Whilst best approached with caution and humility they can be good sources of information on vulnerabilities within a system. Often partial to beer / donuts and may be split into specific sub groups with responsibilities for specific areas such as CICS, z/OS or DB2.

TCO - Total Cost of Ownership, the cost per annum of owning and running zEnterprise (or zSeries) hardware.

TCP/IP - The Internet Protocol stack supported by z/OS which allows z/OS based applications to communicate with IP based hosts/clients. A fully functional TCP/IP stack appears to have been the driving factor in the renewed vigor of System z.

TLA - 3 letter acronym

Transparency - a readily understood process

TSO - Time Sharing Option, An optional z/OS application environment and interface found at most z/OS sites. Mostly used in conjunction with ISPF and is the primary interface for most technical mainframe IT staff.

UACC - aka UACC(NONE). Universal ACCess level. This defines the access level to be granted to a resource where the requestor has not been granted specific access nor have they been granted access by membership of a group. A UACC level above NONE allows the defined level of access unless the requestor has been specifically excluded potentially even if the requestor is not a valid userid.

UID - UNIX System Services representation of userid OR CA ACF2 userid

URM - Unified Resource Manager, a new package supported on the latest zEnterprise mainframes that allows the configuration and management of the entire zEnterprise environment. This includes the configured LPARs and the Blade servers installed in the zBX BladeCenter Extension (if present).

USERID - A z/OS security concept label of 1 to 8

characters assigned to an entity through which all access to resources is granted or rejected. Not to be confused with UID.

USS - UNIX System Services, a fully POSIX compliant implementation of the UNIX operating system that runs as a service under the control of z/OS. A comparatively new part of the z/OS offering that became a required service when IBM rewrote their TCP/IP application to run using USS. As a result of the way USS was introduced there is a tendency for the management, in all aspects including security, to be less well structured and documented than is expected in other areas such as CICS.

VSAM - Virtual Storage Access Method, one of a number of methods of storing and accessing data on z/OS systems. Data held in databases such as DB2 is actually stored within VSAM files with DB2 providing the database table structure and access layer. Each VSAM file consists of one or more (depending on the type of VSAM file) z/OS datasets.

VTAM - Virtual Telecommunications Access Method, the basic network layer used by z/OS to transport data between various areas such as between users and the applications running on z/OS. It uses the SNA protocol and includes CICS LU 6.2 connections. VTAM network traffic is not generally encrypted, although communications can be encrypted using Session Level Encryption. Overall use of VTAM is declining in favour of TCP/IP. Access to applications can be restricted, using an external security manager, based on the VTAM terminal ID.

WLM - WorkLoad Manager, The IBM process used by system z, and other platforms, to handle the workload vs physical resource battle using a goal-orientated process. Although originally restricted to working with a mostly fixed base of physical resources within a sysplex it can now also control significant parts of the entire mainframe resource pool in the same manner. The majority of the cross platform (e.g. AIX and UNIX) work load management tasks have been switched to Unified Resource Manager with the release of the zEnterprise, centralizing Work Load Management to a single interface.

zEnterprise - the latest of IBM's zSeries Mainframe hardware systems launched in Q3 2010

z/OS aka **MVS** - The most advanced operating system known to the human race, unfortunately only legal on mainframes. There is no such thing as the 'blue screen of death' under a fully configured and properly managed z/OS system - ever.



10 Index

- ACF2**, 95, 96
- ACID**, 95
- ACS**, 57, 95, 98
- Address space**, 95
- Anchor point**, 95
- API**, 95
- Batch job**, 95
- BCP**, 95
- Channel Path**, 34, 52, 95, 97
- CHPID**, 36, 41, 52, 53, 55, 58, 60, 61, 82, 92, 95, 98
- CIB**, 40, 95
- CICS**, 5, 9, 30, 45, 48, 74, 79, 95, 99, 100
- CLIST**, 95
- Coupling Facility**, 27, 42, 43, 48, 92, 95
- CF**, 42, 92, 95
- CPC**, 77, 81, 82, 83, 91, 96
- CSYSTEM**, 40, 96
- DASD**, 96, 97
- database**, 100
- Dataset**, 96
- DB2**, 99, 100
- ESCON**, 34, 54, 56, 96
- ESM**, 96, 98
- Exit**, 96
- FICON**, 34, 53, 54, 55, 56, 96
- Hiperspace**, 96
- HSA**, 35, 96
- Hypervisor**, 22, 96
- IMAGE**, 96
- InfiniBand**, 40, 54, 95, 96, 99
- IOCDs**, 52, 55, 96, 97
- IOCP**, 34, 35, 49, 52, 53, 58, 61, 97
- IODF**, 32, 33, 34, 35, 36, 38, 40, 42, 43, 47, 48, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 64, 66, 67, 68, 69, 70, 71, 75, 76, 78, 80, 81, 82, 83, 84, 90, 91, 92, 96, 97, 98
- IPL**, 11, 35, 43, 52, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 87, 88, 91, 97
- IRD**, 54, 97
- ISPF**, 30, 36, 44, 69, 97, 99
- JCL**, 95, 97, 99
- JES2**, 97
- JES3**, 97
- LCSS**, 41, 95, 97
- LID**, 97
- LPAR**, 96, 97
- LSYSTEM**, 40, 97
- MVS**, 97, 98, 100
- Object Map**, 19, 97
- OSCP**, 34, 55, 59, 66, 71, 76, 80, 81, 97
- PARMLIB**, 62, 63, 66, 68, 69, 70, 72, 73, 74, 75, 76, 84, 85, 86, 87, 88, 89, 90, 98
- Partition**, 42, 48, 64, 67, 77, 82, 90, 97, 98
- PCHID**, 41, 52, 61, 82, 98
- POSIX**, 98, 100
- PR/SM**, 22, 42, 44, 58, 96, 98
- PROCID**, 40, 98
- RACF**, 5, 96, 98
- REXX**, 98
- SAF**, 98
- SMF**, 98
- SMFID**, 34, 98, 99
- SMS**, 57, 95, 98
- SNA**, 98, 100
- STC**, 99
- Switch Fabric**, 99
- Symbolics**, 71, 72, 99
- Sysplex**, 96, 98, 99
- Sysplex Timer**
- ETR**, 99
- System logger**, 99
- System z**, 9, 18, 19, 21, 22, 23, 25, 27, 28, 29, 31, 33, 34, 35, 36, 39, 40, 41, 42, 43, 44, 45, 46, 47, 54, 55, 56, 57, 58, 59, 61, 64, 65, 66, 67, 74, 75, 84, 90, 96, 99
- Systems Programmer**, 99
- Sysprog**, 99
- TCP/IP**, 5, 99, 100
- Top Secret**, 95
- TOP SECRET**, 96
- TSO**, 44, 48, 77, 95, 97, 99
- UACC**, 89, 99
- USERID**, 100
- USS**, 98, 100
- VSAM**, 100
- VTAM**, 98, 100
- WebSphere**, 5
- WLM**, 53, 54, 100
- z/OS**, 5, 9, 95, 96, 97, 98, 99, 100



11 Appendices

11.1 CA Technologies Security Management Solutions www.ca.com

The CA Technologies security management solutions for mainframe increase automation, reduce administration time, and simplify the monitoring and management of security resources and applications. The result is a more productive way to manage security that minimizes data risks, as it maximizes infrastructure investment and improves mainframe security operations.

For more information go to <http://www.ca.com/us/mainframe-security.aspx>

CA Auditor for z/OS

CA Auditor for z/OS from CA Technologies helps identify the system, application, and security exposures in z/OS environments that arise from improper system configuration and operational errors, as well as intentional circumvention of controls and malicious attacks. Proper use of CA Auditor helps you establish and maintain the platform operating system integrity to meet such compliance regulations as HIPAA, SOX, GLBA and PCI-DSS.

CA Auditor provides powerful features that help automate routine auditing tasks. The batch "silent auditor" facility allows you to automate all or part of your regular audit regimen. Additionally, the baseline analysis facility allows you to automatically monitor key z/OS configuration elements and detect specific configuration changes.

CA Compliance Manager for z/OS

CA Compliance Manager from CA Technologies provides your organization with a single source for real-time, compliance-related information and events occurring within the mainframe environment. It does all this without the need for additional mainframe staff or manual processes to extract data needed for compliance reporting and risk mitigation.

Help organizations meet challenges of minimizing cost, maintaining compliance for z/OS systems, addressing increasing workloads, and decreasing mainframe skill sets by allowing organizations to easily manage and audit their mainframe environment. This is accomplished through continuous, real-time monitoring and collection of compliance and security-related information, policy alerting, and an intuitive reporting interface for compliance and security event reporting. CA Compliance Manager provides powerful internal controls to help ensure that only authorized access is allowed. It also gives you the comprehensive auditing skills that you need to prove your compliance to IT and risk-management auditors.

CA Cleanup

The CA Cleanup products from CA Technologies helps reduce the effort and pressure associated with maintaining current regulatory, statutory and audit requirements by removing obsolete, unused, redundant and excessive access rights through easily automated, virtually unattended and continuous cleanup of CA ACF2, CA Top Secret and IBM RACF security databases.

11.2 IBM Corporation www.ibm.com

IBM Health Checker for z/OS

IBM's Health Checker for z/OS provides a foundation to help simplify and automate the identification of potential configuration problems before they impact system availability. It compares active values and settings to those suggested by IBM or defined by your installation. The IBM Health Checker for z/OS consists of:

- The framework, which manages functions such as check registration, messaging, scheduling, command processing, logging, and reporting. The framework is provided as an open architecture in support of check writing. The IBM Health Checker for z/OS framework is available as a base function starting with z/OS V1.7.

IBM Health Checker for z/OS is also available for z/OS V1.4, V1.5, and V1.6 as a z/OS Web download. Make sure that you review the PSP bucket as described in the Web download program directory. There is required service that you must install. This code is part of z/OS V1.4 and V1.5, which have reached end of service. This code is provided without service for those releases.

- Checks, which evaluate settings and definitions specific to products, elements, or components. Checks are provided separately and are independent of the framework. The architecture of the framework supports checks written by IBM, independent software vendors (ISVs), and users. You can manage checks and define overrides to defaults using the MODIFY command or the HZSPRMxx parmlib member.

IBM-supplied checks can be integrated with the product, element, or component, or they can be provided as APARs in between z/OS releases. To obtain checks that are provided as APARs, use the Technical help database for mainframe Preventive Service Planning buckets. Once there, scroll down to the Find the bucket by type, category, and release heading, select Function for the Type field, select HCHECKER for the Category field, and click Go.

There are checks available for multiple z/OS components and elements.

Note: Many of the checks are also supported on multiple z/OS releases. Review the check APARs for the specific releases supported.

zSecure

IBM's zSecure is a multi-part family of products which provides cost-effective security administration, improves service by detecting threats, and reduces risk with automated audit and compliance reporting.

www-01.ibm.com/software/tivoli/products/zsecure/

The zSecure suite consists of multiple products for IBM z/OS® and IBM z/VM® designed to help you administer your mainframe security, monitor for threats, audit usage and configurations, and enforce policy compliance.

The zSecure suite helps improve the efficiency and manageability of the mainframe security environment. Administration, provisioning and management products can significantly reduce administration overhead, contributing to improved productivity, faster response time and reduced training time needed for new security personnel.

These offerings include:

- IBM Security zSecure Admin
- IBM Security zSecure Visual
- IBM Security zSecure CICS®Toolkit

Audit, monitoring and compliance products help ease the burden of compliance audits, improve security and incident handling, and increase overall operational effectiveness. Offerings include:

- IBM Security zSecure Audit
- IBM Security zSecure Alert
- IBM Security zSecure Command Verifier

Combined audit and administration functions for the IBM Resource Access Control Facility (RACF®) feature on z/VM are provided by:

- IBM Tivoli® zSecure Manager for RACF z/VM

In addition, the Security zSecure suite also allows you to feed mainframe security information into an enterprise audit and compliance solution, through seamless integration with IBM Tivoli Security Information and Event Manager.

11.3 Key Resources www.vatsecurity.com

VAT Vulnerability Analysis Tool from KRI

z/OS is known as a secure operating system. This does not mean that there are no vulnerabilities in it. Did you know that system integrity exposures can be found in Supervisor Call (SVC) Interfaces, Operating System Exits, Program Call (PC) routines and Linkage Indexes (LX) interfaces, in addition to Authorized Program Function (APF) authorized programs? These exposures exist in software supplied by IBM, ISVs and locally developed programs and exits. We continuously hear about zero-day vulnerabilities in Windows and apply patches sent by Microsoft every week, but what about the z/OS mainframe? Does it have similar exposures? The answer is yes.

- VAT helps safeguard your organization
- Prevents financial loss through fraud
- Provides compliance
 - o PCI Requirement 11.3
 - o NIST 800-53
 - o ISO 27001
 - o etc.

VAT has identified close to 100 exposures in z/OS and its sub-systems. There are fixes for many of the exposures, but they are classified as integrity fixes by IBM (or the ISVs involved), so you won't know (and neither would the person at the Vendor Help Desk) details about the fix. IBM distributes these PTFs with no description other than they are marked as Integrity PTFs. So, be sure to apply all Integrity PTFs! And there is nothing you can do to address the issues with system settings or operational practices, you have to depend on the vendors involved to address these issues!

11.4 NewEra Software, Inc. www.newera.com

Image Control Environment (ICE)

Image FOCUS is an Image Control Environment (ICE) Application whose primary function is to provide Inspection and Baseline services to users of z/OS, its sub-systems and Parallel Sysplex.

Image FOCUS ensures, to the extent possible, the maximum availability of a z/OS Sysplex and its Images. To accomplish this, Image FOCUS and its companion tools - Change Detection and Inspection Server – are grouped into “Views”. Each view – Production, Workbench and Recovery – is designed to support a focused

set of management activities: New Release Analysis, Change Analysis, System Recovery and Image/Sysplex Inspection. Each enables the Image FOCUS user to quickly gain a full understanding of the complete z/OS configuration.

Inspection Services performs a “Virtual IPL” of each Image beginning with the validation of the IPL Unit Address and LOADPARM, PARMLIB and PROCLIB. Members are checked for syntactical correctness and related datasets for referential integrity and attribute characteristics that would result in a future IPL failure. Subsystem and Sysplex relationships are inspected and cross-checked with other Images.

Baseline Services builds and stores a “Blueprint” of valid, viable configurations. Each contains the content of configuration members and/or files discovered during the “Virtual IPL”. Each Baseline is automatically updated at a defined monitoring interval. Continuous updates ensure working configuration copies and provide the basis for configuration change detection.

The Controls Environment (TCE)

The Control Editor is a Controls Environment Application that bridges the gap between system security requirements and system programming needs by providing to its users an unobtrusive ISPF environment from which they identify and report on the changes to and the dynamic use of system configuration settings.

Conventional z/OS Change Management methodologies, often directly supported by the preventive function of the External Security Manager, have served well for authorizing and documenting what is to be changed and who has been given implementation authority but they do little if anything to document and verify what has actually changed. To keep you in the loop, verify change activity and ensure compliance, a native z/OS real-time change environment – The Controls Environment (TCE) – is an absolute business necessity.

It will ensure that z/OS configuration baselines are maintained, configuration changes – both staged and dynamic – are identified and that changes to the policies that dictate the operation of your External Security Manager (RACF, CA ACF2 or CA Top Secret) or changes to those system policies that define the state of your z/OS Sysplex don't go unnoticed.

TCE's Event Notification System (ENS) helps you verify z/OS changes now, not later. Certainly finding out what's changed tomorrow after post-processing millions of SMF records will do little to improve your reaction time to real-time z/OS change events. It becomes your change management partner, blending seamlessly with existing processes. You will know what is changing as it changes and gain the ability to route, as defined by you, captured events in real time to the members of your control team by email, to your Security Information Management (SIM) System as TEXT or XML or HTML, and to any other MVS or web-based repository all in real time as they occur. ENS affords you the flexibility to stay on top of what is happening to your z/OS Sysplex, as it happens, no matter how complex the environment.

StepOne

StepOne is a shareware software audit tool designed specifically for Information Systems Auditors facing the challenge of auditing IBM's zEnterprise as outlined in this eBook.

Using StepOne, you will explore the DNA of the most powerful Central Processing Complex (CPC) available to your business and governmental clients. The interactive native environment used by StepOne allows you to dynamically and

automatically create a zEnterprise Configuration Baseline.

For the first time, you will enjoy the advantages of seeing the zEnterprise Fabric from a MACRO perspective. The significance of each individual configuration component and its potential impact on the overall integrity of each Logical Partition (LPAR), the primary control boundary of the zEnterprise, becomes immediately clear.

11.5 Vanguard Integrity Professionals www.go2vanguard.com

Vanguard Analyzer™

Vanguard Analyzer delivers expert-level vulnerability assessments and audit results for System z® in minutes, using Vanguard's extensive knowledge base of security best practices. The software provides comprehensive system integrity, verification and auditing capabilities, including assessment, risk identification, threat analysis, and specific instructions on how to fix identified problems. By providing an in-depth overview of current system status, identifying exposures in simple business risk language, and prioritizing findings for immediate action, Vanguard Analyzer simplifies the audit process.

Auditors can use Vanguard Analyzer to quickly take a system snapshot, perform specific individual audit functions, or perform a full audit automatically. Vanguard Analyzer ranks messages by importance so that the most critical issues can be addressed quickly and appropriately. After an audit is run, a panel detailing an audit's result, or findings, are displayed with highlighted areas that identify where potential security exposures may exist.

Without Vanguard Analyzer, a high level of z/OS® system expertise was required to determine if an audit finding was significant, define what the finding really meant for IT security integrity, and explain how it could be fixed. Vanguard Analyzer runs thousands of integrity checks automatically, provides clear explanations of findings in terms of business risks, includes detailed recommendations for corrective action based on practices implemented by major auditing firms, and offers expert-level policy implementation guidelines that both security staff and management can understand.

Vanguard Analyzer provides the following benefits:

- Ensures expert-level audit results.
- Reduces the number of manual tasks required for a system assessment.
- Improves the quality and consistency of the assessment.
- Enables management to demonstrate due diligence in regulatory compliance and audit readiness.
- Delivers the needed information to the correct people quickly.

Vanguard Enforcer™

Vanguard Enforcer provides proven, real-time intrusion protection, detection and management solutions for the z/OS mainframe that prevent human error and deliberate attacks. By providing 24x7 protection for critical information and resources hosted on the mainframes, Vanguard Enforcer guarantees that z/OS and RACF® security standards, profiles, rules and settings are not compromised. In less than two seconds, the software can automatically detect and notify personnel when threat events on the mainframe and network occur, then respond to deviations from the security baseline with corrective actions that reassert the approved security policy.

With Vanguard Enforcer, organizations can more easily meet the demands of

regulatory compliance standards that require continuous oversight to ensure that approved IT/IS controls are in place and stay that way. And, organizations can be confident that their z/OS and RACF security implementation is protecting critical data and resources and continuously adhering to best practices standards. Vanguard Enforcer is the only intrusion detection solution for the mainframe to be awarded Common Criteria certification

(EAL 3+), and is the 2009 and 2010 winner of Government Security News Magazine's Homeland Security Award for the Best Intrusion Detection/Intrusion Protection systems.

To maintain security Vanguard Enforcer:

- Provides continuous, periodic scanning of RACF security profiles looking for deviations from the policy baseline and taking automated corrective action.
- Monitors and detects threat-related security events without human intervention.
- Notifies security personnel on a wide variety of intrusion-related events via text message, cell phones, MVS console, email, etc.
- Automates security measures to reduce operating expenses.
- Improves the productivity and effectiveness of scarce security staff.
- Manages the security implementation baseline that enforces an organization's security policies.

Vanguard Configuration Manager™

Vanguard Configuration Manager is an automated software scanner that enables continuous monitoring of System z security configuration settings. The software supports implementing and utilizing the z/OS and RACF configuration checklist from the National Checklist Program (NCP) of the National Institute of Standards and Testing (NIST) and the Department of Homeland Security (DHS).

Vanguard Configuration Manager significantly reduces the cost and time required for government agencies and contractors to test and assess their compliance with NCP guidelines for IBM® z/OS and RACF. Since April of 2011, the NCP recognizes the Vanguard Security Technical Implementation Guidelines (STIG), based upon the Defense Information Systems Agency (DISA) STIGs for z/OS and RACF, as the minimum required configuration controls for z/OS and RACF systems. Vanguard Configuration Manager is the only fully automated scanner for the z/OS and RACF STIGs.

Organizations using Vanguard Configuration Manager are saving thousands of hours each year when performing quarterly NCP and/or DISA STIG assessments. Those that implement continuous monitoring will save tens or hundreds of thousands of hours each hour by deploying Vanguard Configuration Manager.

Vanguard Configuration Manager provides the following benefits:

- Dramatically reduces the costs of configuration control testing and reporting based on NIST/DHS standards.
- Significantly enhances z/OS security.
- Provides built-in intelligence about z/OS and RACF STIG details.
- Automates the testing of more than 300 System z STIG checks.
- Produces accurate DISA STIG compliance reports in minutes.
- Enables implementation of continuous monitoring.
- Easy to deploy and use.
- Reduces human error in the compliance checking and reporting process.

