M A News
Mainframe Audit News
January, 2002
Issue Number 02

Table of Contents

I N T H I S I S S U E

(YOU CAN USE the Find function of your email or Acrobat software to jump directly to a given section. For example, to jump directly to the Websites section, you would Find on "7)" or on "Websites".)

1) Introducing the Mainframe Audit News

This is the second edition of the Mainframe Audit News, a vehicle for sharing information about auditing IBM mainframe computers.

The MA News is a free, email, newsletter for auditors who need (or suspect that they will need) to be auditing IBM mainframe systems (primarily MVS, OS/390, z/OS, and the system software associated with them). This software includes: CICS, DB2, JES, VTAM, MQSeries, TSO, USS (UNIX System Services), TCP/IP, and others. It also includes the Websphere server software which connects a mainframe to the Internet. (Note, we will expand each of these acronyms and explain how the software works over the course of the next several issues.)

We expect to have a new issue at least every three months. We will never include attachments to the email.

The MA News is meant for auditors who are new to IBM mainframes, as well as for experienced MVS auditors who want to keep up to date with the latest developments from IBM. We will not make the list of subscribers available to anyone else for any reason.

To Sign Up for the Mainframe Auditors' Newsletter, see section 9) below.

---

2) How Does IBM Make UNIX Fit Into MVS? (What is USS?)

For a long time, it seemed that IBM wanted to ignore the UNIX operating system (originally developed at AT&T). IBM seemed to hope that instead everyone would use IBM operating systems, such as MVS on the mainframe computer.

In the past six or seven years however, IBM has gone out of its way to make it easy for non-IBM computers to connect to IBM computers. This has required IBM to support many cross-platform standards, such as TCP/IP, DCE, and the Internet. Many of these standards were first developed on UNIX omputers.

IBM wisely incorporated a standardized version of UNIX within the MVS operating system, and offered it for free. This version of UNIX was originally called OMVS (Open Edition/MVS), and later renamed (for reasons only IBM could understand) to USS (UNIX System Services). What follows is a description of how IBM made UNIX work well under MVS, while maintaining the security and control MVS has always provided.

MVS (like any other operating system) is a program. It can cause other programs to execute under its control (the same way that the Windows operating system can make your word processor execute under its control). When MVS starts up, it starts USS as a program under MVS' control. USS does all the standard functions that any UNIX supports, but is completely controlled by MVS.

For security, there are two major controls points: identifying the user (by means of a userid and password, usually), AND controlling who can read or write files.

To identify the user, most other versions of UNIX maintain a file containing userids and encrypted passwords. This file is usually called /etc/passwd (pronounced "ett-see password"). When a user logs on to UNIX, UNIX compares the userid and password typed by the user to the information in /etc/passwd. If a match is found, then the user is allowed to proceed. A large percentage of the break-ins to UNIX computers have been based on reading the /etc/passwd file and learning the passwords in it. This file is vulnerable to a variety of different attacks which can allow unauthorized sers to read it.

Rather than trying to fine-tune this situation, IBM wisely chose to eliminate the /etc/passwd file, replacing it with a call to the MVS security software (such as ACF2, RACF, or TopSecret). This provides more rigorous security, and integrates security administration so that users are defined in just one place for both MVS and USS.

To control who can read or write files, you need to understand how UNIX files are organized. On most UNIX systems, files are organized hierarchically into directories. Each disk drive has a "top" directory, which can contain files and also other directories (called "sub-directories"). Each subdirectory can contain files and also other directories, and so on. This file structure is called HFS, for Hierarchical File System.

(If this structure seems familiar, it is probably because this is the way Windows organizes the hard drive on your computer. Directories on Windows are also called "folders". Windows stole the concept from UNIX.)

UNIX uses slashes ("/") to separate the names of directories. So a file called january.dat in the payroll directory which is a sub-directory of the production directory might be named /production/payroll/january.dat. (Notice that UNIX uses forward slashes where Windows uses backwards slashes.)

In adding USS to MVS, IBM uses a large MVS dataset to contain the tree of directories, sub-directories, and files. This MVS dataset (called the "HFS dataset") is comparable to the hard drive on your personal computer, in that is contains the directory/sub-directory/file structure.

For each file (including for each directory), UNIX maintains a File Security Packet, which specifies who can read, who can write, and who can execute the file. When a user tries to read a file, UNIX compares the information in the File Security Packet to the user definition from the /etc/passwd file to decide whether to permit the access.

When IBM added USS to MVS, they kept the concept of File Security

Packet intact. When a user under USS tries to read a file, USS (UNIX) calls the MVS security software to compare the information in the File Security Packet to the user description from the security software, to decide whether to permit the access.

With this approach IBM has kept the design of UNIX intact. (USS is the first version of UNIX to be branded "UNIX".) They have also kept the MVS security mechanisms intact, and integrated security administration for MVS and for USS. The result has been a large success in the marketplace, since USS has been well received. On MVS computers, USS serves as the foundation for software which connects to the Internet.

IBM has added several security features to UNIX which make USS perhaps the most secure, commonly used UNIX available. These include mechanisms for delegating authority with greater granularity, use of Access Control Lists to supplement the File Security Packets, and controls over privileged programs.

If you plan to audit USS, you will want to learn about the UNIX file system and the File Security Packet, as well as how USS works with the mainframe security software to enhance standard UNIX security.

When planning audits for the up-coming year, you should consider what applications are executing under USS, what Internet connections your mainframe supports, and the quality of security controls over them all. You will want to collect this information before starting the audit planning and budget process.

---

3) What's the Difference Between IMS and DB2?

IMS and DB2 are both database mangagement software for IBM mainframe computers.

We explain these concepts in these sub-sections:

3A) What is database management software?
3B) What is IMS?
3C) How is DB2?
3D) What This Means to the Auditor

3A) >>>>What is Database Management Software?

Database management software is a program which manipulates large amounts of related data in a way that lets several other programs read and write the data at the same time. For example, all the customer information for a company might be stored in a CUSTINFO database. Order entry staff might be using programs which need to read and write information such as customer name and address, quantity ordered, and so on. At the same time, Accounts Receivable might need to update the customer information regarding payments received. The Marketing Department might want to use other programs reading the same data to produce sales analysis reports. Database management software, such as IMS and DB2, makes this possible.

Database management software keeps a single copy of the data, so that updates need to be made in only one place (a plus for data integrity). It manages concurrent requests to read and write data from other programs in a way that avoids conflicts. It usually provides for backup and recovery of data if the computer hardware should fail. It allows data analysis from a variety of viewpoints.

Originally, IMS was IBM's premier product for database management. It has now been superceded by DB2, as described below.

3B) >>>>What is IMS?

IMS (Information Management System) is database software from IBM that keeps track of the relationships between records by maintaining pointers from one record to another. For example, the record describing a customer might have the pointer (actual address on the disk drive) of a record describing an order from that customer. To answer the question "What orders has this customer made?", IMS would use the pointer in the customer record to find the order record.

In the 1960s and 1970s, IMS was considered one of the few database products that could handle very large amounts of data with good response time.

IMS supports security mechanisms to control: who can use it, what transactions each user can execute, and what data each user can read or write. To audit IMS security you evaluate how effectively these mechanisms are implemented.

There are two types of security mechanism for IMS: internal and external. The internal mechanisms are older and less reliable, since they

were designed for IMS only. The rules for IMS internal security are maintained in a special dataset by a program called SMU (rhymes with "moo"), for Security Maintenance Utility.

The external mechanisms consist of calls to external security software (such as ACF2, RACF, or TopSecret). The external security is more reliable, and permits people to use the same user identification and password for IMS that they use for other software.

Whether the security is internal or external, it still consists of controlling: who can use IMS, what transactions a user can execute, and what data the user can read or write.

Users can connect to IMS by signing on at a terminal (that is, prove who they are by entering a userid and a password). Users can also connect to IMS by executing some other program (for example, CICS, as described in Issue 01). The other program would then pass requests for data to IMS. A third way to connect to IMS is from another computer, for example a Windows computer. In this case, users would sign onto the Windows computer, and the Windows computer would pass the requests to read and write data to IMS on the mainframe computer. Each of these three ways of connecting to IMS involves different techniques for proving who the user is. The reliability of these techiques is often a significant concern in IMS audits.
IMS is older than DB2, and becoming obsolete. However, your installation may still have critical systems which rely on IMS databases. Converting these to the more modern DB2 would be an unreasonably difficult project. Newer applications are more likely to make use of DB2.

3C) >>>>What is DB2?

IBM mathematicians spent much time and energy studying how a database management system SHOULD work. They developed a mathematical model, called the "relational model", which was so good that IBM used it as the basis for a new database product: DB2 (DataBase 2). DB2 does not have pointers from one record to another. Instead, it maintains data in two-dimensional tables, made up of rows and columns. To manipulate this data, IBM developed a standardized lanquage called SQL (Structured Query Language).

(Side note: Larry Ellison read some of the research on the relational model, and based his own database management software on it. That software, of course, is Oracle, and it has made Larry Ellison's company (also called Oracle) one of the market leaders in database management software. Because of Oracle's and DB2's common origins, you will find that they both use the SQL

langugage. Many other database software products use SQL as well, including Access and SQL Server. SQL is not difficult to learn, and can prove useful to an auditor in many situations.)

Instead of pointers from one record to another, DB2 relates information in tables by the values in different columns. For example, there might be a customer table, with one row per customer. The customer table might have a column containing the customer number. Orders might be stored as rows of a separate table, also having a column containing the customer number. To answer the question "What orders has this customer made?", DB2 would examine the order table, scanning for all rows which have a matching customer number in the customer number column.

This approach, and the relational model on which it is based, provides significant advantages for data integrity, processing power, and efficiency. DB2 now dominates the market for database software on mainframes. IBM has created versions of DB2 for other types of computers, including Windows NT, where, some people contend, DB2 also dominates the market.

To audit DB2 security, you would audit the security mechanisms to control: who can use it, and what data each user can read or write. (There are no controls over who can execute transactions, since DB2 does not support transactions.)

As with IMS, there are both internal and external security mechanisms. The internal mechanisms are older, and specific to DB2. On the mainframe, the internal security consists of seven tables where the security rules are stored.

The external mechanisms consist of calls to the external security software (such as ACF2, RACF, or TopSecret). Whether internal or external, the mechanisms address the same issues: who can use DB2, and what data each user can read or write.

Users cannot log onto DB2 from a terminal. It does not support terminals. Instead, a user must log onto some other application (such as CICS, IMS, a batch job, or a Windows computer), and connect to DB2. The way this connection is controlled determines how the user is identified to DB2 security. This often becomes a significant part of the findings in a DB2 audit.

3D) >>>>What This Means to the Auditor

For planning and scoping the audits, you will want to know how many copies of IMS and DB2 are being used. With IMS, each copy is called a "region". There might, for example, be a "test IMS region" and a "production IMS region". Or there might be separate regions for each of Marketing, Finance, and Inventory.

Copies of DB2 are called subsystems, and each has a name of up-to-four characters. There might be "DB2T, the test subsystem" and "DB2P, the production subsystem". Before you commit to "audit IMS and DB2", find out how many IMS regions, and how many DB2 subsystems exist, and if necessary, narrow your scope to just one. For DB2, you need to know the names of the subsystems, since this name will be used by the security mechanisms.

For any IMS or DB2 system you audit, you will want to know whether it uses internal or external security. For IMS internal security, you might want access to the rules specified in the SMU output. For DB2 internal security, you might want to be able to select (that is, to read) the information in the seven security tables.

For either system using external security, you will want the ability to read the security software rules which control access to data and use of resources.

For both IMS and DB2, your audit will likely need to address the way in which the software determines who a user is. Much of this information can be requested before the audit actually starts.

4) What Is VTAM?

VTAM (Virtual Telecommunications Access Method) is the system software which controls all telephone and communication connections to a mainframe computer. Every terminal connected to the computer is controlled by VTAM. No terminal can be used by an MVS computer system unless it is defined to VTAM.

4A) >>>>What does VTAM Do?

VTAM maintains definitions of the terminals which can connect to the mainframe, and of the programs which terminals can be connected to. (These programs are called "applids". Examples include: CICS, TSO, and IMS, as discussed in this issue and in the previous issue. The definition of an applid is "a program to which you can connect from a terminal".

VTAM provides the logical connections between terminals and applids. Such connections are called "binds". By controlling which terminals can connect to which applids, VTAM can provide one part of the system's security architecture.

When you sit down at a terminal connected to the mainframe, you must first request a "bind" to whatever applid you want to work with. (In some cases, this bind is requested automatically for you, or is converted from your selection off a menu.) Once you have the bind, then VTAM makes sure that everything you type on the terminal gets sent to that applid.

It is only after you have the bind (in most cases), that the applid can call the security software to verify your userid and password. If the applid does not call the security software (perhaps it has its own, hard-coded list of userids and passwords), then you almost certainly have an audit finding. The risk of using hard-coded lists of userid and passwords instead of calling the security software is this: when a user terminates employment, standard procedure is to revoke his or her privileges, but only within the security software. This means that the former employee, who is computer literate, can still access the system by going through the applid with hard-coded lists.

4B) >>>>Where Are Terminals and Programs Defined to VTAM?

The VTAM system programmer defines all the terminals and applids in a dataset usually called SYS1.VTAMLST. It may have a different name in your installation, but if you ask for read access to "VTAMLST", people will know what you mean. (You will need to get someone to show you how to "read VTAMLST", since the terminal and applid definitions are not straight-forward.)

This dataset is a very powerful control, since terminals can't use the system unless they are defined here. Programs also cannot use terminals unless they are defined in SYS1.VTAMLST. The VTAM system programmer can define restrictions within SYS1.VTAMLST over which terminals can bind to which applids.

Of course, the security software must protect SYS1.VTAMLST, preventing unauthorized users from updating it. The security software can also restrict which terminals can bind to which applids.

4C) >>>>What Does This Mean to the Auditor?

VTAM is a powerful source of control for MVS systems. It is often overlooked because most auditors, and many security administrators, do not understand it. To prepare for a VTAM audit, you will want to request read access to SYS1.VTAMLST (or to have someone present you with a valid list of applids and terminals). You should ask for a list of the controls within VTAM that have been implemented to restrict binds between terminals and applids. (These might include: USS tables, cross-domain definitions, path definitions, and others beyond the scope of this article.)

You will want to review security software rules protecting SYS1.VTAMLST, as well as rules controlling use of terminals and applids.

Security software can also control the VTAMAPPL resource, which prevents someone from writing a program which pretends to be a valid applid. This is another frequently overlooked control in VTAM.

You will want to review the security policy to determine whether every applid is required to call the security software to verify userids and passwords.

Since VTAM can automatically provide encryption for certain network connections, you should determine the organization's policy regarding network encryption, and then evaluate how well the policy is implmented.

All of this will add up to an effective assessment of how well the paths into the system are controlled.

5) Question and Answer Column

(Readers are invited to send Questions to stu@stuhenderson.com, along with an indication of whether we should print your name. If you need an immediate reply to your question, please indicate this in your email. We made up the first question for the first issue. The second question is for real.)

Our second question is from Anna Moore of CSC.

Q) What are the recommended techniques to audit mainframe access by external users and facilities, including, but certainly not limited to APPC, NJE, RJE, dial-in, web, etc?

A) This is a smart question, since you can't audit something unless you understand the architecture, including all the control points. Each of the

items you list is a path into the system. As an auditor, you need to identify every path into the system, so that you can evaluate the controls over each path.

There are two sources to learn the paths into the system:
1) The standard list of POEs (Ports of Entry), that is a place where work can enter the MVS system. These include: the card reader, TSO logons, the internal reader (a method for one piece of software to submit work to the system), NJE, and RJE. (NJE is Network Job Entry, a method for one computer to submit work to another computer over the network. RJE is Remote Job Entry, a method for a dumb terminal to submit work to the computer over the network.) All of thes POEs can be controlled by the security software (such as ACF2, RACF, or TopSecret). In your audit, you should determine what the organization's policy specifies for control over the POEs. Then determine (mostly from VTAM and the security software) how well those policies are implemented.

The definitions for NJE and RJE connections may be found in the control file for the JES software (which we will discuss in a future issue). It is quite reasonable for you to request the JES system programmer to show you the NJE and RJE definitions in the JES control file. Then request the security administrator to show you the rules controlling these paths.

2) VTAM (Virtual Telecommunications Access Method), since all connections to the system come through VTAM. The VTAM system programmer maintains a dataset, often called SYS1.VTAMLST, in which every connection is described. (Please refer to previous article.) SYS1.VTAMLST will include descriptions of terminal, dial-in ports (which are treated like terminals), NJE and RJE connections, APPC (Advanced Program to Program Communication) links, and TCP/IP (Transport Control Protocol / Internet Protocol, used by UNIX and over the Internet).

   SYS1.VTAMLST will also specify the definition of every applid. An applid is a program which VTAM lets you connect to from a terminal. For effective security, policy should require every such program to call the security software to verify the userid and password typed in by the user at the terminal.

You will want to determine the organization's policy regarding control of each of these paths, and then determine how well the policy is implemented.

6) Software Tools for Mainframe Audits

Auditors often use tools such as CA/Examine to conduct MVS audits, and often end up unsure what to do with the results. We will list several such tools, with contact information about the vendors, as well as a list of functions that the tools support. We will then explain how to go about using the results.

6A) >>>>Names of Tools

The tools listed below are the ones we have heard the most about. If you know of other tools for which the vendor can supply the names of three satified users, please tell us and we will print the name of the tool in a future issue.

We do not evaluate any of these tools, nor recommend one over the other. We do however believe that any auditor looking for such a tool should consider at least the ones listed here. Of course, to evaluate each tool you should ask for the names of at least three satisfied customers and call them to find out what they like and dislike. You might also consider requesting a demo disk or other trial version, so that you can guage the "look and feel" of the product. If you audit many locations, consider requesting a "travelling license" which will let you use a single copy of the product at more than one site.

> CA-Examine (http://www.cai.com or (631) 342-6000)
> Consul (http://www.consul.com or (888) 323-0880)
> ICU from Janus (http://www.janussecurity.com) (203)251-0200)
> Vanguard Analyzer (http://www.viplink.com (714)939-0377)

6B) >>>>Functions the Tools Perform

The MVS operating system has a secure architecture, which can be compromised by system programmers "opening back doors" to let certain programs obtain the same privileges that MVS has. Examples of such back doors (which we will explain in future issues) include: APF authorization, User SVCs, and the Program Properties Table (which we discussed in Issue 01). These "back doors" are often necessary to have the system function effectively. The question for auditors is whether the back doors are opened in a way which provides adequate protection against abuse.

Unless you are an experienced MVS system programmer, it will be difficult for you to identify all the back doors, and even more difficult to evaluate whether each one is adequately protected. These tools provide help by listing the back doors which have been opened in your installation. Some of them provide limited analysis of how well certain back doors are protected. However, we know of know product which provides comprehensive analysis. (Darn, now you have to write the audit report yourself!)

Below is a partial list of information reported by one of these products about back-doors which have been opened on the system. Space does not permit us to list every feature of every product, nor to perform a comprehensive evaluation. Each vendor should be able to provide you with a list of the information its product reports, and the analytic functions it provides.

System Settings and Software Levels, System IPL Parameters, SMF Options,
IO-Appendages, MPF List Suppressing Console Messages,
Program Call Definitions, MVS Subsystem Definitions,
MVS Program Properties Table, Common Writable Storage,
Virtual Storage Report, SVC Definitions, TSO Authorized Commands,
APF Authorized Programs, Users with USS UID=0,
USS File System Mount Points, USS Controlled Program Files

6C) >>>>How to Use the Results of Such Tools

Ask the system programming manager for a list of all back-doors (such as User SVCs, APF authorized libraries, and so on; he will know exactly what you mean) which have been authorized on the system. (The system programming manager, in theory at least, is the one who should have authorized them.) Some of the back-doors may be the result of installation modifications to the operating system. Most of them will be the result of installing purchased software which makes use of them. Request for each back-door a description of how we know that it is adequately protected.

Use the output of the software tool to tell you what back-doors are actually opened on the system. Compare this to the list provided by the system programming manager. If they don't match you have one finding. If the system programming manager doesn't have such a list, and can't produce one within three days, you have a different finding. If there is no evidence to demonstrate that the back-doors which have been opened are known to be adequately controlled, then you have yet a different finding.

PLEASE DO NOT MAKE THIS MISTAKE, which is very common, and which

causes system programmers to laugh hysterically: If an auditor tries to interpret the output of these software tools without having the knowledge of a system programmer, he or she can will often finish by writing audit findings which make no sense. A common example: if the software tool documents 217 APF-authorized libraries [don't worry what APF means for now, just understand that it is one of the possible back-doors], some auditors have written findings like: "This installation has 217 APF-authorized libraries, which seems a high number. We recommend that you reduce this number."

There are usually very good reasons why the APF-authorized libraries which exist are there. The number of them is not significant. Audit findings like this often generate comments in closing meetings like: "Oh yeah, well, what's the RIGHT number?" (There is, of course, no "right" number. This question is unanswerable, to the delight of the auditee.)

The point of your audit should be to determine whether only authorized back-doors have been opened, and that they are adequately controlled. You can use the software tools to tell you what back-doors have been opened. You compare this to the "authorized" list of approved back-doors, if there is one. You review the change-control management procedures for updates to system datasets, and the security software rules supporting these procedures, to evaluate whether unauthorized changes could be made.

7) Websites for Mainframe Auditors, Six "How To Audit" Seminars, and the Proverb of the Day

7A) >>>>Websites for Mainframe Auditors

Here are more websites useful to mainframe auditors. We do not endorse (nor disparage) any of their products, nor the quality of their websites, since we have not had an opportunity to evaluate them. However, we find them interesting, and hope you will too.

http://www.auditnet.org for Jim Kaplan's Auditnet (considered by some to be THE web site for auditors)

http://www.loftcam.com/magic.html for a magic trick to test your brain

http://www.techweb.com for the business technology network, including a glossary and search capability

http://www.plr-services.com/cicsmqs.htm for links on CICS security

http://www.theiia.org/itaudit for the Institute of Internal Auditors IT audit

section

http://www.atstake.com for @stake research and tools on digital security

7B) >>>>Six "How to Audit" Seminars

Six new "How to Audit..." courses are available for IT auditors:
How to Audit Cross-Platform Applications

> How to Audit Mainframe/Internet Connections

> How to Audit TCP/IP

> How to Audit CICS

> How to Audit RACF

> How to Audit MVS

To learn more about them, go to
http://www.stuhenderson.com/XAUDTTXT.HTM

7C) >>>>This Issue's Proverb of the Day

"Most programs you need to audit have a single control file where the system programmer specifies all the options, including security and other audit-related options."

For example, for MVS the control file is named SYS1.PARMLIB. For VTAM, it is called SYS1.VTAMLST. Before an audit starts, you might want to find out the name of the relevant control file, and get either a printed copy of it or READ access to it. This provides a large amount of information to you before the audit begins. Auditees who know that you have the printout (and may even be able to interpret parts of it) are more likely to be forthcoming in interviews. Having the printout in your working papers can help to demonstrate that you have been diligent in collecting and verifying relevant data.

8) Tell Us What You Think

We'd love to hear from you, in particular on these topics:

--What do you like/not like about the MANEWS?

-- What websites do you know that you want to share with other auditors?

-- What topics and/or columns would you like to see in future issues?

-- Is your mainframe connected to the Internet and do you plan to audit the security implications of this connection?

Please email your comments to stu@stuhenderson.com. Thanks.

9) How to Subscribe/Unsubscribe

This section shows you how to:

9A) Subscribe,
9B) Unsubscribe,
9C) Request back issues,
9D) Take advantage of our free technical support for mainframe auditors.

9A) >>>>To Subscribe to the Mainframe Auditors' Newsletter (MA News)

Send an email to: stu@stuhenderson.com
with the subject field set to: MA News
and in the body of the email just this word: SUBSCRIBE

9B) >>>>To Unsubscribe from the Mainframe Auditors' Newsletter (MA News)

Send an email to: stu@stuhenderson.com
with the subject field set to: MA News
and in the body of the email just this word: UNSUBSCRIBE

9C) >>>>To Request Back Issues of the Mainframe Auditors' Newsletter (MA News)

Send an email to: stu@stuhenderson.com
with the subject field set to: MA News
and in the body of the email just this phrase
(for example to request issues 1 and 2): Back Issues: 1, 2

9D) >>>>To Get Questions Answered from the Mainframe Auditors' Newsletter (MA News)

Send an email to: stu@stuhenderson.com

with the subject field set to: MA News
and in the body of the email the word: Question: (followed by
your question, and tell us whether you want your name included or not if we
decide to publish your question and answer)

Whether we print your question and answer or not, we will try to email you
back an answer to your question within five business days. If you need a
faster answer, please phone your question to (301) 229-7187, leaving the
question on the machine. Please repeat your phone number slowly and clearly.

10)Feature Article: SYS1.PARMLIB Developments Auditors Need to Know

SYS1.PARMLIB Developments Auditors Need to Know

(* indicates a technical term explained at the end of the article.)

Most software packages have a single control dataset where the system
programmer specifies the options or parameters for that installation of the
software. Auditors benefit by knowing what these datasets are, and how to
interpret them. The data gathering and analysis stages of the audit are much
simpler with this knowledge.

For the MVS (or OS/390 or z/OS) operating system, this control dataset
is called SYS1.PARMLIB. Auditors often include printouts of it in working
papers, since it is where the system programmer specifies all the "backdoors"
to system security, such as APF-authorized libraries and user SVCs. (These
backdoors will be discussed in a future issue.) This parmlib is where system
programmers also specify other options, such as what records are logged in SMF
data.

MANEWS 02

Two recent developments in SYS1.PARMLIB make life easier for the system programmer. Auditors of MVS systems need to be familiar with them. They are:

System Symbols (a way of specifying substitution values for variables)

AND

Concatenated Parmlibs (a way of combining several parmlibs to treat them as one)

1) System Symbols

System symbols are special variables whose names begin with an ampersand (&). For example, **&SYSNAME** is usually set to the name of the system. **&SYSCLONE** is usually set to the last two characters of &SYSNAME. These symbols make it easier to share a single copy of certain control files, including SYS1.PARMLIB. Other symbols are defined by IBM and by your system programmers.

For example, imagine three CPUs (Central Processing Units, that is, three computers, or three systems) in one data center named: **SYSTEM0A**, **SYSTEM0B**, and **SYSTEM0C**. The value of &SYSNAME on each system might be set to
SYSTEM0A, SYSTEM0B, and SYSTEM0C, respectively.

The value of &SYSCLONE might be set to **0A**, **0B**, and **0C**, respectively. Now if the system programmer specifies APF=&SYSCLONE, then MVS interprets this as **APF=0A** or **APF=0B** or **APF=0C**, depending which system is executing.

Another example: if the name of the SMF dataset is specified as **SMF.&SYSNAME..DATA**, then MVS will interpret this as **SMF.SYSTEM0A.DATA**, etc., depending upon which system is executed.

These symbols make it possible for the system programmer to use one copy of SYS1.PARMLIB for several systems, since the system symbols are automatically replaced on each system with the value appropriate to that system.

These symbols are also used in calls to security software (such as RACF, ACF2, or TopSecret). For example, to secure mainframe connections to the Internet, security software uses the resource class **SERVAUTH**. Rules in this class can have names made up of pieces which include the value of **&SYSNAME**.

The values of the system symbols are specified in the **IEASYMxx** members* of parmlib described below. At system start-up, they are listed on the console in message IEA009I.

CPUs, LPARs, Sysplexes, and Systems

Before introducing concatenated parmlibs, we should explain these four terms. A **CPU** (Central Processing Unit, similar to the Pentium chip in your personal computer) is the part of the computer which performs arithmetic and logic functions. Each CPU can execute its own copy of the MVS operating system (the same way a Pentium chip can execute a copy of Windows).

If a CPU is logically split into two or more pieces, each piece is called an **LPAR or Logical Partition**. Each LPAR appears to be its own CPU and can execute its own copy of the MVS operating system.

Moving conceptually in the other direction, several CPUs and/or LPARs can be linked together into a single image called a **sysplex**. Each CPU and LPAR in a sysplex executes its own copy of MVS. However, the CPUs and LPARs in a sysplex are connected by fiber optic cables. These cables are used to exchange data, to share the workload across the CPUs, and to bind the CPUs and LPARs into a single image.

For purposes of this article, a system is either a CPU which is not broken into LPARs, OR a single LPAR. A **system** corresponds to a single copy of the MVS operating system.

2) Concatenated Parmlibs

Originally, each MVS system had its own copy of parmlib (that is, parameter library, where the system program specifies the system options or parameter settings). This **parmlib** was always named SYS1.PARMLIB. The starting point in analyzing it was always the member* named **IEASYS00**.

The member **IEASYS00** has pointers to other members, for example, in IEASYS00, "**APF=03**" would point to the member **IEAAPF03** (also in SYS1.PARMLIB).

With concatenated parmlibs now, there may be up to 10 parmlibs concatenated with SYS1.PARMLIB. ("Concatenated" means that the up-to-eleven datasets are logically considered to be stuck together, to form what amounts to a single dataset. Members like IEAAPF03 may be found in any of the up-to-

eleven datasets, which are processed in the order in which they are concatenated.)

Often SYS1.PARMLIB will be used to contain IBM-supplied values, while other parmlibs concatenated with it will contain values specific to your installation.

Along with this increase in the number of parmlibs, IBM has added an earlier step to the start-up process. When MVS starts up, it now reads a member named **LOADxx** from a certain library. (The library will be described below. The computer operator can specify the value of xx [which is often set to 00, resulting in the name LOAD00].) A single LOADxx member can provide the starting point for all the systems at once.

This single starting point works like this: Each system has three key filters which identify it. They are: **HWNAME** (hardware name, that is the name of the CPU), **LPARNAME** (logical partition name), and **VMUSERID** (the VM userid of the guest virtual machine under which MVS is executing. If your installation doesn't use the VM operating system, you may ignore this.) The value of HWNAME is the default value for **&SYSNAME**.

The **LOADxx** member specifies certain values for each system (that is for each copy of MVS executing). These values include information about: device configuration, catalog names, parmlibs, the value of **xx** for **IEASYMxx**, and the value of **xx** for **IEASYSxx**.

The values are associated with each system by means of the three filters. For example, here is a possible **LOADxx** member for an installation with three systems. Two of the systems are LPARS on the same CPU (**BIGJOHN**). The other system is a CPU (**LITTLJOE**) which is not broken into LPARs.

Extract of LOAD00

*** ANY LINE STARTING WITH AN ASTERISK IS A COMMENT**
THE FOLLOWING SETS VALUES FOR BIGJOHN, LPAR JOHNAA
HW BIGJOHN
LPARNAME JOHNAA
SYSPARM 0A
PARMLIB SYS1.AA.PARMLIB

*** FOLLOWING SETS VALUES FOR BIGJOHN, LPAR JOHNBB**
HW BIGJOHN
LPARNAME JOHNBB
IEASYM 0B
SYSPARM 0B
PARMLIB SYSX.BB.TEST.PARMLIB
***(SYSPARM 0B SAYS TO USE IEASYS0B; IEASYM 0B SAYS TO USE IEASYM0B)**
HW LITTLJOE
SYSPARM 0C
IEASYM 0C
PARMLIB SYSX.0C.PARMLIB
PARMLIB SYSX.JOE.PARMLIB
PARMLIB SYSX.LITTLJOE.PARMLIB

Note that the entry for **JOHNBB** specifies SYSPARM is equal to **0B**. This would point to a member named **IEASYM0B**, where the system symbols are defined for that system:

Extract of IEASYM0B
SYSDEF HWNAME(BIGJOHN)
LPARNAME(JOHNBB)
SYSNAME(SYSTEM0B)
SYSCLONE(0B)

Logic Steps for MVS Start-Up

To understand how these changes affect the audit, it is useful to review how they are processed when the MVS operating system starts up.

1) When MVS starts up, it sets values for the three filters (**HWNAME**, **LPARNAME**, and **VMUSERID**).
2) MVS reads one LOADxx member. It finds it by looking in these libraries in order: SYSn.IPLPARM on the IODF* disk pack (where n is a digit from 0-9), SYS1.PARMLIB on the IODF* disk pack, and SYS1.PARMLIB on the sysres* pack.

3) From the **LOADxx** member, MVS finds the entries which match the three filters. From these entries it extracts information about: device configuration, catalog names, parmlibs, the value of xx for members named IEASYMxx [from IEASYM], and the value of xx for members named IEASYSxx [from the SYSPARM value].
4) MVS uses the parmlib information to build the list of up to ten parmlibs which are concatenated in front of SYS1.PARMLIB. For **SYSTEM0B** in our example, the parmlib concatenation would be **SYSX.BB.TEST.PARMLIB**.
5) MVS then reads the IEASYMxx members in these parmlibs to determine the names and values of the system symbols. For **SYSTEM0B** in our example, these would be from the extract of **IEASYM0B** illustrated above.
6) MVS then reads the **IEASYSxx** members in these parmlibs to find the pointers to other members. For example, **APF=0A** in **IEASYSxx** would point to a member named IEAAPF0A where the names of the APF authorized libraries are specified. (We will describe APF-authorized libraries in a future issue.) MVS reads these other members to determine all the system option settings, and sets them accordingly. (Note that the values of xx for IEASYSxx can be specified in the LOADxx member, in the IEASYMxx member, and by the operator.)

Should You Be Looking at the Specific Settings or the Controls OVER Them?

There is probably not enough time in most audit budgets to examine every setting. The auditor would need to be an experienced system programmer in any case. It is much more effective to review the controls over the option settings. (You still need to have some understanding of the settings in order to demonstrate why the controls are important, and in some cases, to be ready to demonstrate specific, actual risk.)

To evaluate the controls over the system settings, you need to know what datasets are involved. You can learn these by following the logic steps above. You then need to examine the change control process over these datasets, and the security software rules which define who can update them. (Shouldn't system programmers go through the same sort of quality assurance process as application programmers who want to put a test program into production?)

What This Means for Audit Scoping and Planning

Two key parameters for audit scoping are the **count** (how many things are we auditing, (for example, how many MVS systems, or how many CICS regions) and the **aspect** ("Are we auditing for security, efficiency, effectiveness, user satisfaction, or something else?").

To plan an MVS audit (or the upcoming year's audit budget), you need to know how many MVS systems there are, and which ones you plan to include in the audit scope. This information should be kept in the audit department's standing files, and updated periodically.

You can get this information by interviewing the system programmers, and/or by reviewing the parmlibs yourself. Whether you want to verify what the system programmers tell you, or you want to collect the information directly yourself, you will want to be at least familiar with the concepts of system symbols and concatenated parmlibs.

What This Means for Real Life

Sometimes there is little scoping or planning. You get a directive like: "**Go audit MVS. You have 31 hours. Don't come back without good findings**." For an accelerated (read "quick and dirty, yet worthwhile") audit, you would : learn the names of the control datasets (including SYSn.IPLPARM and all the parmlib concatenations); get printouts of at least some of them; ask the security administrator who is permitted to update them; and ask the system programming manager about the change control process. Then write your draft report, review it with the auditees. Put your working papers in a single stack for later filing.

What This Means for Work Papers

To minimize paperwork, while providing a foundation for audit findings, you might want to keep either a summary, or actual printouts of not just SYS1.PARMLIB, but also all its concatenations, especially the LOADxx members.

One way to summarize this information is to create a chart like the one that follows (Ask your system programmer to review it and help you update it periodically.)

It would not be difficult to develop software to read all the control files and generate a summary of this information. Already, tools such as CA-Examine, ICU/MVS, Vanguard, and CONSUL can provide the auditor with summaries of what options are actually being used at any time.

An Example of a Summary Chart for Your Mainframes

Standing File - Summary of MVS Systems at XYZ Co.

(In this case, there are two CPUs. The CPU named **BIGJOHN** is split into two LPARS (**JOHNAA** and **JOHNBB**).

System✍	Sys A	Sys B	Sys C	Future
Variable				
System Name	SYSTEM0A	SYSTEM0B	SYSTEM0C	
Purpose	Development	Production	Internet	
HWNAME (CPU)	BIGJOHN	BIGJOHN	LITTLJOE	
LPARNAME	JOHNAA	JOHNBB		
VMUSERID				
SYSPLEX Name				
&SYSNAME	SYSTEM0A	SYSTEM0B	SYSTEM0C	
&SYSCLONE	0A	0B	0C	

Terms Used in this Article

- *IODF (Input Output Definition File) is a dataset where the IO devices for the system are defined.
- *A member is like a mini-dataset, which is part of a real dataset. Members have names of up-to-eight characters. The member named IEAAPF00 In the dataset SYS1.PARMLIB could be referred to as SYS1.PARMLIB(IEAAPF00).
- * Sysres (System Residence) is the disk pack from which the system was started up or IPL'd (Initial Program Load). It is where much of the operating system is defined. Think of a personal computer where you can IPL off a diskette (the A: drive) or off the hard drive (the C:). The A: drive or the C: drive would be the sysres pack, depending on which one you IPL'd from.

This article is extracted from an issue of the Mainframe Audit News, and is reprinted with the generous permission of its editors. For further information about the MA News, or to get a free, email subscription, go on the Internet to www.stuhenderson.com.

Stu Henderson, (301) 229-7187, stu@stuhenderson.com, www.stuhenderson.com