AMERICA FIRST
CREDIT UNION

Operations Manager at AFCU

# My Background




2002

## Helpdesk

- Password Monkey™
- pOwErPoiNt
- Heavy monitors
- Floppy disks
- Introduced to Linux

2005

## Sysadmin

- SCO Unix (shudder)
- AIX (RS6000)
- Informix
- Physical Tape
- Driving a lot
- Kernel panics
- Impact printers
- Lots of Bash

2007

## Forensics

- Encase
- File systems
- Lots of Hex
- Progress bars
- Regular Expression
- Court Appearances
- Evidence Bags
- Misanthropy

2013

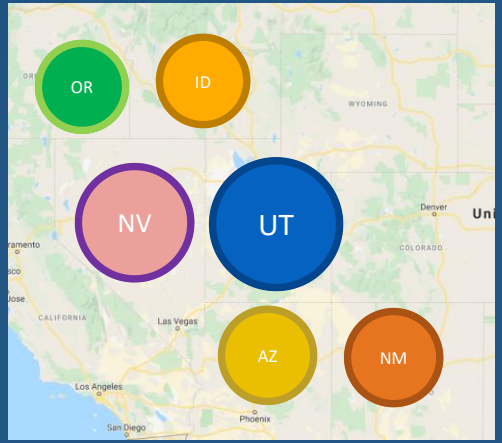
## The Mainframe

## Distributed Systems

# America First Credit Union

1939 – Fort Douglas Civilian Employees Credit Union

- Not for profit, member-owned organization
- More than 1 million members
- ~$15 billion in assets
- 131 branches (UT, NV, ID, AZ, OR, NM)



# VM at AFCU

- 30+ Years of VM
- Historically VSE and Linux hosted as guests, now primarily Linux
- IFL Growth
- SSI for LGR and DR
- 2013 – increased in-sourcing of VM knowledge
- Improvements in tooling (RACF, Operations Manager)
- Some things didn’t fit the bill
- 2021 - Separation of Workload
- Mentorship & The Community

# VM at AFCU

- 30+ Years of VM
- Historically VSE and Lin
- IFL Growth
- SSI for LGR and DR
- 2013 – increased in-sou
- Improvements in toolin
- Some things didn't fit t
- 2021 - Separation of W
- Mentorship & The Com


# Introduction to Operations Manager

# Finally Getting It

- Building the case to management
    - Demonstrating use cases
    - Hands-on PoC
    - Working with the SMEs…well, SME!
- Installation (INSTPROD vs. Program Directory)
- Learning by doing
- Groking the doc

```
 VMWRKSHP CONFIG   A1   F 80   Trunc=80
====>
00000 * * * Top of File * * *
00001 DEFRULE NAME AFCUOPS +
00002    MATCH '*PURCHASE COMPLETE*' +
00003    USER GHOWELLS +
00004    ACTION INSOPMGR
00005 DEFACTN NAME INSOPMGR +
00006    COMMAND 'EXEC YAY' +
00007    ENV ALL +
00008    NEXTACTN GETTOWRK
00009 * * * End of File * * *
```

# Out of the Box Benefits

- Guest visibility
- Consoles
    - Linux guest startup problems
    - What could have been with z/OS
    - VM Utility Guests
- Spool Files
- General System State

# Out of the Box Benefits

```
09:32:43 GRAF L0003 DIALED TO LEVEL2   0300 DIALED= 1     FROM 192.168.69.141
09:56:06 GRAF L0003 DROP FROM LEVEL2   0300 DIALED= 0
09:56:11 GRAF L0004 LOGON  AS  MAINT    USERS = 63    BY GHOWELLS FROM 192.168.69
09:56:11 GRAF L0004 LOGON  AS  MAINT    USERS = 63    BY GHOWELLS FROM 192.168.69
09:56:11 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
09:56:11 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
09:56:31 GRAF L0005 LOGON  AS  GHOWELLS USERS = 64    FROM 192.168.69.141
09:56:31 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
09:56:40 GRAF L0005 LOGOFF AS  GHOWELLS USERS = 63
09:56:40 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
09:56:47 ICH408I USER(GHOWELLS) GROUP(HUMANS  ) NAME(GERARD HOWELLS      )
09:56:47 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
09:56:47   LOGON/JOB INITIATION - INVALID PASSWORD
09:56:47 * -- Operations Manager Action SHIPLIN  scheduled for execution -- *
 PF01= SCROLL PF02=        PF03= END    PF04=        PF05= HOLD   PF06= FORMAT
 PF07= UP     PF08= DOWN   PF09=        PF10= LEFT   PF11= RIGHT  PF12= RECALL
```

# First Project: SIEM Integration

- What motivated this?
- Environment – Splunk, SYSLOG
- Setting achievable goals
    - Logins, Logouts, Violations
- Making sense of the data (key/value pairs)
- How did I do it?
- Let’s take a look!

# SIEM Integration

OPERATOR console logon message:

```
09:56:31 GRAF L0005 LOGON  AS  GHOWELLS USERS = 64    FROM 192.168.69.141
```

What I want Splunk to get:

```
ACTION=LOGON USERNAME=GHOWELLS IP=192.168.69.141
```

Let’s use some REXX!

```
/* Parse the arguments passed by OpsManager */
parse arg ipcsname rule msgebcdic
/* Count number of words in message text */
'PIPE VAR msgebcdic | COUNT WORDS | VAR wordnum'
/* Populate stem variable with message text */
message.0=wordnum
do i=1 to wordnum
 'PIPE VAR msgebcdic | SPECS W' i '| VAR tmp'
 message.i=tmp
end
/* Build Message to send to Splunk */
msgbld = 'ACTION=‘message.4 ‘USERNAME='message.6 'IP='message.11
```

# SIEM Integration

## Last couple of bits of REXX

```
/* Translate into ASCII for the outside world */
'PIPE var msgbld | XLATE E2A | var msgascii'
/* Use OpsManager function to send to Splunk */
"GOMCMD OPMGRM1 CMDIPCS NAME" ipcsname "COMMAND SEND REXXVAR msgascii"
```

## Operations Manager Rules and Actions

```
DEFRULE NAME LOGON +
  MATCH '*LOGON  AS * USERS = *FROM*' +
  USER OPERATOR +
  PARM 'LOGON' +
  ACTION HILOGON
*
DEFACTN NAME HILOGON +
  INPUT CBL +
  NEXTACTN SHIPLIN
*
DEFACTN NAME SHIPLIN +
  COMMAND 'EXEC E2AKVP SYSLOUT &P &T' +
  ENV LVM
```

# Guest Logon Automation at IPL

- What motivated this?
- Understanding the order
- Creating definitions and standards for naming
- Solving the identification problem
- Establishing the hooks
- Using OPERATOR console as a central point of coordination
- Making it “intelligent” (I’ll let you be the judge if I succeeded!)
- Let’s take a look!

# Guest Logon Automation at IPL

A **BUNCH** of EXECs

```
 MAINT     FILELIST A0   V 169  Trunc=169 Size=9 Line=1 Col=1 Alt=9
Cmd    Filename Filetype Fm Format Lrecl     Records      Blocks    Date      Time
       LNXIPL   EXEC     X1 V          82         123           2  2/26/20 14:43:12
       CHECKCOM EXEC     X1 V         105         102           1  2/14/20 11:38:57
       CHECKENV EXEC     X1 V          75          28           1  2/14/20  9:40:53
       FINIPL   EXEC     X1 V          75          25           1  2/14/20  8:30:04
       STGDONE  EXEC     X1 V          75          15           1  2/13/20 15:31:37
       STARTMDL EXEC     X1 V          82          63           1  2/13/20 13:33:47
       STARTDBS EXEC     X1 V          80          63           1  2/13/20 13:31:56
       STARTSUP EXEC     X1 V          79          58           1  2/13/20 10:10:45
       SMTPMSG  EXEC     X1 V          96         169           2  1/29/20 16:01:14
```

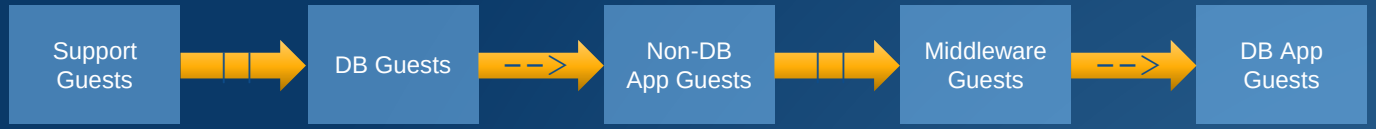
# Guest Logon Automation at IPL

A **TON** of Rules and Actions

```
*****************************************************************
*****************************************************************
* Support Linux Guest IPL Section (SUP)                         *
* -------------------------------------                         *
* The following rules and actions log on the support Linux guests *
* triggiering off of strings from guest consoles and SYSLOG output. *
*************
* SUP START *
*************
*
* ENVCHCK: Watch for message from environment section completion exec
*          and start IPL of support Linux guests.
*
DEFRULE NAME RENVCHCK +
  MATCH '*STAGE ENV COMPLETE*' +
  USER OPERATOR +
  PARM SUP +
  ACTION ACMNLNXI
*
*DEFACTN NAME AENVCHCK +
*  COMMAND 'EXEC STARTSUP' +
*  ENV SVM +
*  NEXTACTN ACMNHIWH
*
* SUPSTRT: Watch for messages from STARTSUP EXEC and issue XAUTOLOGs
*          for support Linux guests based on rules.
*
DEFRULE NAME RSUPSTR1 +
```

# Guest Logon Automation at IPL

## IPL CONFIG file – Establishing Order, Defining Things




```
*********************************************************************
* Definitions                                                       *
* -----------                                                       *
* Support Guests - Serve support functions such as NFS, DNS and     *
*                  performance monitoring.                          *
* DB Guests      - DB2/Other database-centric guests.               *
* Non-DB App     - Application guests that are not reliant on DB    *
*                  guests that are not fully started.               *
* Middleware App - Middleware application guests (IIB/MQ)           *
* DB-App         - Guests that run applications which are dependent *
*                  upon databases that run on DB guests.            *
* ||             - Sequential flow dependent on completion of parent. *
* -->            - Concurrent flow, executed in parallel.           *
* RSSS....       - Indicates a Rule name, SSS is the three character *
*                  Section name identifier.                         *
* ASSS....       - Indicates an Action name, SSS being the three    *
*                  character Section name identifier.               *
```

# Guest Logon Automation at IPL

## Getting Started – AUTOLOG1 (or 2, thanks RACF)

```
/* A few lines from &LPARNAME EXEC, called by PROFILE EXEC */
'CP XAUTOLOG OPMGRM1 CMD IPL'
'CP SLEEP 5 SEC'
'MESSAGE OPERATOR STARTING IPL'
```

## Operations Manager Rules and Actions

- Dividing into sections, 3 letter identifier used in rules/actions
- Flow from top to bottom (mostly!)
- Common functions defined to be called from within sections

### Sections

| | | | | | | |
|---|---|---|---|---|---|---|
| DFL: Default Rules, used to establish OpsManager environment. | CMN: Common Rules, callable for shared functions. | ENV: Checks the environment prior to logging on Linux guests. | SUP: Starts support Linux guests. Things like DNS/NFS and Monitoring. | DBS: Starts database Linux guests. Things like DB2 LUW servers. | MDL: Starts Middleware Linux guests. IIB/Broker etc. | FIN: Cleans up and starts OpsManager in normal mode. |

# Guest Logon Automation at IPL

Let’s look at some rules and actions!

After Environment checking finishes, the support Linux guests start

```
* ENVCHCK: Watch for message from environment section completion exec
*          and start IPL of support Linux guests.
*
DEFRULE NAME RENVCHCK +
  MATCH '*STAGE ENV COMPLETE*' +
  USER OPERATOR +
  PARM SUP +
  ACTION ACMNLNXI

* CMNLNXI: Common action to IPL Linux guests based upon type.
* Rules that call this action should contain a parameter indicating
* the type of guests to be IPLed.
*
DEFACTN NAME ACMNLNXI +
  COMMAND 'EXEC LNXIPL &P' +
  ENV SVM
```

# Guest Logon Automation at IPL

## LNXIPL EXEC highlights

```
/* Access LNXMAINT's 191 Disk, contains PARM files for Linux Guests */
'VMLINK LNXMAINT 191 <FEFE P> (QUIET'
/* Gather list of guests to start and populate stem variable */
'PIPE CMS LISTF LNX* PARM P | STEM guests.'
/* Read parm files from guests stem variable and populate stem with */
/* TYPE= values. */
types.0 = guests.0
do i=1 to guests.0
   'PIPE <' guests.i '| LOCATE /TYPE=/ | SPEC 6-* | VAR types.'i
end
/* Detach LNXMAINT 191 */
'PIPE CP DETACH FEFE | LOCATE /FEFE/ | DROP'
/* Determine which Linux environment we're working with */
select
when env = 'SUP' then do
  call startsup
end
...
```

# Guest Logon Automation at IPL

## LNXIPL EXEC highlights

```
/* Subroutine to start Support guests */
startsup:
/* Parse types stem and get list of guests with TYPE=SUPPORT */
do i=1 to guests.0
  'PIPE LITERAL' guests.i types.i '| LOCATE /SUPPORT/ | SPEC W1 | >> GST LST A'
end
'PIPE < GST LST A | STEM supguests.'
'ERASE GST LST A'

/* Send Message to OPERATOR and logon guests*/
'CP MESSAGE OPERATOR LOGGING ON SUPPORT GUESTS'
do i=1 to supguests.0
  'CP XAUTOLOG' supguests.i
  'CP SLEEP 2 SEC'
end
/* Send message to OPERATOR when done */
'CP MESSAGE OPERATOR SUPPORT GUESTS LOGGED ON'
exit 0
```

# Guest Logon Automation at IPL

Let’s look at some rules and actions!

LNXIPL wrote a message to OPERATOR that matches a rule:

```
* SUPDONE: Watch OPERATOR console to see when SUPPORT
*          guests are done logging on.
*
DEFRULE NAME RSUPDONE +
 MATCH '*SUPPORT GUESTS LOGGED ON*' +
 USER OPERATOR +
 PARM SUP +
 ACTION ACMNSTGD
```

```
* CMNSTGD: Common action to execute STGDONE EXEC when IPL stages
* complete and report-in to OPERATOR.
*
DEFACTN NAME ACMNSTGD +
  COMMAND 'EXEC STGDONE &P' +
  ENV SVM +
  NEXTACTN ACMNHIGR
```

# Guest Logon Automation at IPL

STGDONE EXEC

```
/* Parse variable passed by action */
parse arg stg

/* Send message to OPERATOR */
'CP MESSAGE OPERATOR STAGE' stg 'COMPLETE'
```

Next rule in IPL CONFIG matches this condition

```
* SUPCHCK: Watch for message from support section completion exec
*          and start IPL of Database/AppNoDB Linux guests.
*
DEFRULE NAME RSUPCHCK +
  MATCH '*STAGE SUP COMPLETE*' +
  USER OPERATOR +
  ACTION ADBSSTRT
```

The cycle then repeats until all of the Linux guest start sections are complete

# Guest Logon Automation at IPL

Finishing up

Once the last Linux logon stage completes, we call an EXEC through a rule/action to complete the IPL and put Operations Manager into “normal” mode.

```
* MDLCHCK: Watch for message from Middleware section completion exec
*          and start IPL of Middleware/AppDB Linux guests.
*
DEFRULE NAME RMDLCHCK +
  MATCH '*STAGE MDL COMPLETE*' +
  USER OPERATOR +
  ACTION AFINSTRT
*
* FINSTRT: The beginning of the end!
*
DEFACTN NAME AFINSTRT +
  COMMAND 'EXEC FINIPL' +
  ENV SVM
```

# Guest Logon Automation at IPL

FINIPL EXEC

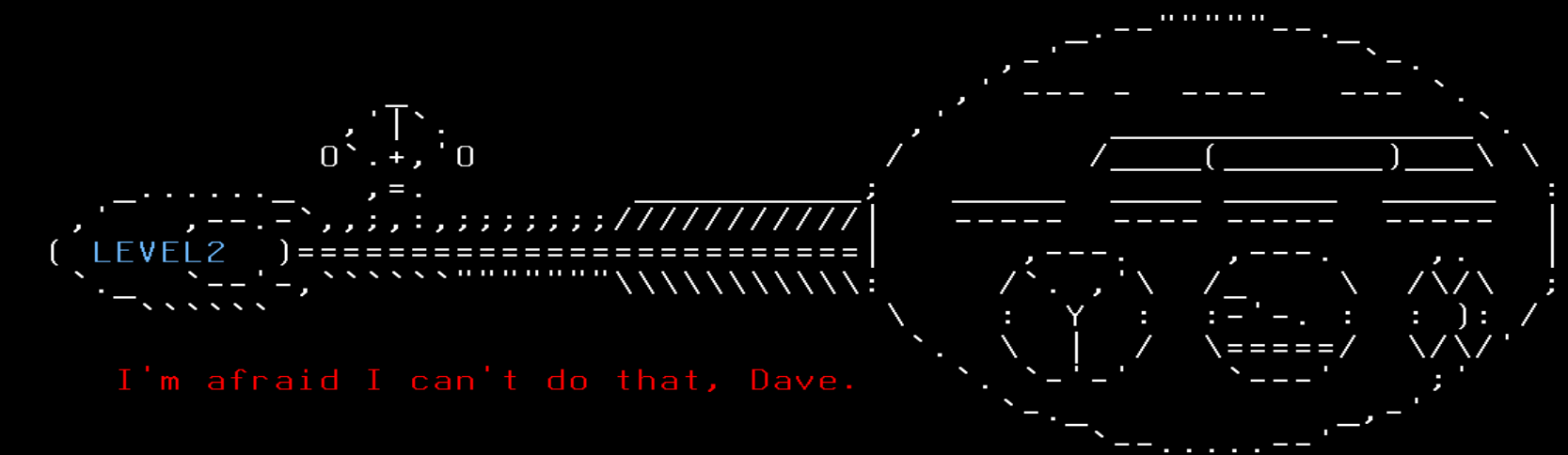
```
/* Figure out what LPAR we're on */
'PIPE CP QUERY USERID | SPECS W3 | VAR lpar'

/* Issue a CONFIG command to read in the common config file */
'CP MESSAGE OPERATOR LOADING OPERATIONS MANAGER COMMON CONFIGURATION'
'GOMCMD OPMGRM1 CONFIG FN OPMGRM1 FT CONFIG FM E CLEAR ALL'
'GOMCMD OPMGRM1 CONFIG FN COMMON FT CONFIG FM E'

/* Issue a CONFIG command to read in the LPAR-specific config file */
'CP MESSAGE OPERATOR LOADING OPERATIONS MANAGER' lpar 'CONFIGURATION'
'GOMCMD OPMGRM1 CONFIG FN' lpar 'FT CONFIG FM E'

/* Send message to OPERATOR confirming end of IPL */
'CP MESSAGE OPERATOR IPL COMPLETE'
exit
```

IPL COMPLETE!




I'm afraid I can't do that, Dave.

==========
Greetings Program ----------
==========
>>

# Talking done.

# Questions?

Contact Info
ghowells@americafirst.com
(801) 827-8353

AMERICA FIRST

CREDIT UNION