Understanding Someone Else's ACS Routines

Neal Bohling *and* ***Chris Taylor***

IBM

August 13, 2015

Session# 17836

Copyright (c) 2015 by SHARE Inc. Except where otherwise noted, this work is licensed under http://creativecommons.org/licenses/by-nc-sa/3.0/

Agenda

- The Problem
- Quick ACS Review
- Tools and Preparation
- Scenarios
  - Debugging ACS
  - Adding a Rule
- Preparing for Transitions
- Considering a Rewrite?
- General ACS Tips

```
IF &LABEL ¬= &DSN(3)
      THEN
        DO
          WRITE '&LABEL NOT SET AS INDICATED BY &DSN(3)'
          WRITE '&LABEL NOT VERIFIED BY DATACLAS'
          EXIT CODE(16)
        END
      ELSE
    IF &DSN(4) = 'TWO'
      THEN
        DO
          IF &FILENUM = 2
            THEN
              DO
                WRITE '&FILENUM NOT 2 AS INDICATED BY &DSN(4)'
                WRITE '&FILENUM NOT VERIFIED BY DATACLAS'
                EXIT CODE(16)
              END
            ELSE
              DO
                WRITE 'DATACLAS ACS ROUTINE -
                VERIFIED &FILENUM = '&FILENUM' '
              END
        END
    ELSE
    IF &DSN(4) = 'ONE'
      THEN
        DO
          IF &FILENUM = 1
            THEN
              DO
                WRITE '&FILENUM NOT 1 AS INDICATED BY &DSN(4)'
                WRITE '&FILENUM NOT VERIFIED BY DATACLAS'
                EXIT CODE(16)
              END
            ELSE
              DO
                WRITE '&FILENUM NOT 1 AS INDICATED BY &DSN(4)'
                WRITE '&FILENUM NOT VERIFIED BY DATACLAS'
                EXIT CODE(16)
              END
        END
    ELSE
      THEN
        DO
          IF &FILENUM ¬= 2
            THEN
              DO
                WRITE '&FILENUM NOT 2 AS INDICATED BY &DSN(4)'
                WRITE '&FILENUM NOT VERIFIED BY DATACLAS'
                EXIT CODE(16)
              END
            ELSE
              DO
                WRITE 'DATACLAS ACS ROUTINE -
                VERIFIED &FILENUM = '&FILENUM' '
              END
        END
```

The Problem

- All routines require periodic updates
- Multiple writers may have different styles
- Special clauses can get added

- **Over time, these updates can cloud the original intent and logic**

The Solution!

- **Unfortunately there is no easy solution.**
- **But you can do to make it easier!**

- **The Purpose of this Presentation:**
  - Demonstrate how to find the info you need
  - Tips and tricks for making current ACS easier to read
  - Discuss concepts for good ACS

Part 1 – ACS Review

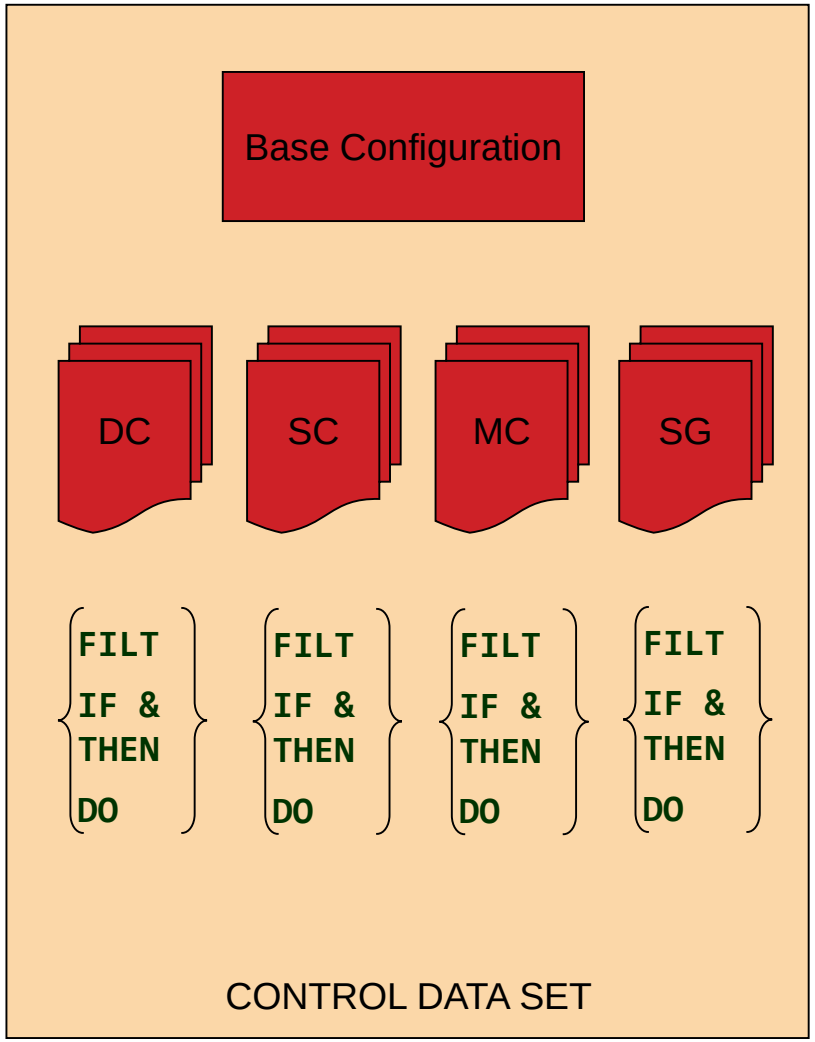
SMS Review

SMS has 4 classes:

- **Data class** – assigns allocation defaults (like size, volcnt)
- **Storage class*** – assigns performance attributes
- **Management class** – defines backup characteristics
- **Storage group*** – groups of volumes

Each class has it's own ACS routine

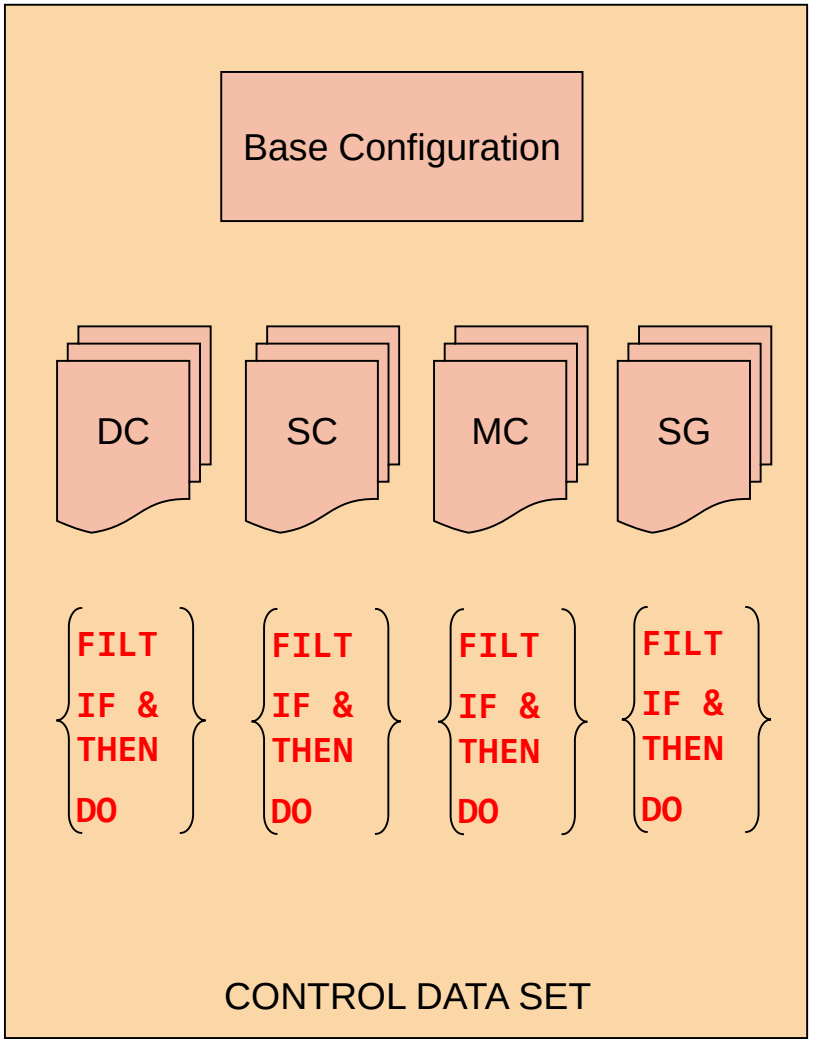
*required to be SMS-managed



What is ACS?

- **Automated Class Selection**
- User-defined rules
- Assigns SMS classes

- **YOU** tell **SMS** how to act



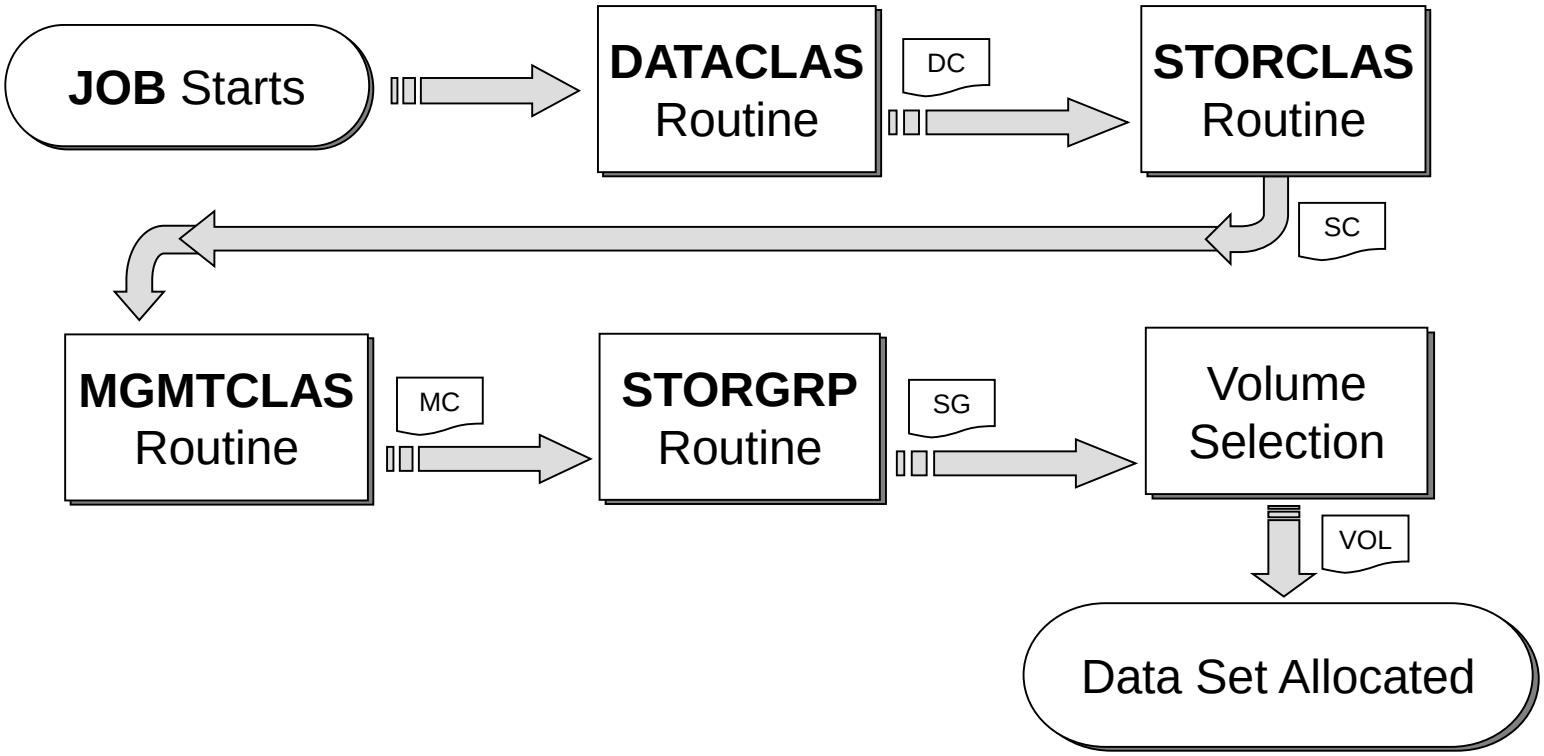
Sample ACS

```
PROC STORCLAS
FILTLIST DBVOLS INCLUDE(IMS*,DB2*)                    /* ALL DATABASE VOLUMES */
                EXCLUDE('IMS053','DB2007')
FILTLIST DBJOBS INCLUDE(IMS*,PROD*,ACCT*)               /* ALL DATA BASE JOBS */
FILTLIST VALID_UNITS
INCLUDE('3330','3340','3350','3375','3380',
        '3390','SYSDA','')                             /* VALID UNITS FOR SMS */
IF &UNIT ^= &VALID_UNITS
   THEN DO
      SET &STORCLAS = ''
      WRITE 'INVALID UNIT TYPE FOR SMS ALLOCATION'
      EXIT
   END
SELECT
   WHEN (&DSN = SYS1.**)                                     /* SYSTEM DATA */
      SET &STORCLAS = 'SYSTEM'
   WHEN ((&ALLVOL = &DBVOLS) && (&JOB = &DBJOBS))           /* DATABASE DATA */
      SET &STORCLAS = 'DBPOOL'
    WHEN ((&DSN(3) = 'CLEAR') | (&ANYVOL ^= TSO*))           /* NON-SMS DATA */
      SET &STORCLAS = ''
    WHEN (&DEF_STORCLAS ^= '')                            /* IF DEFAULTS EXIST */
      SET &STORCLAS = &DEF_STORCLAS;
    OTHERWISE
      SET &STORCLAS = 'COMMON'                               /* ALL OTHER DATA */
END
```

ACS Syntax Notes

- Always start with a **PROC**
- **END** everything – DO, PROC, SELECT
- **IF** needs **THEN**
- **SELECT** should have **WHEN** and **OTHERWISE**
- **/* Comments look like this */**
- Literals are in quotes: 'MY.DATASET' or '3390'
- Masks are not in quotes: MY.* or 33*
- +/- continue literals to the next line

Processing Order



ACS Philosophy

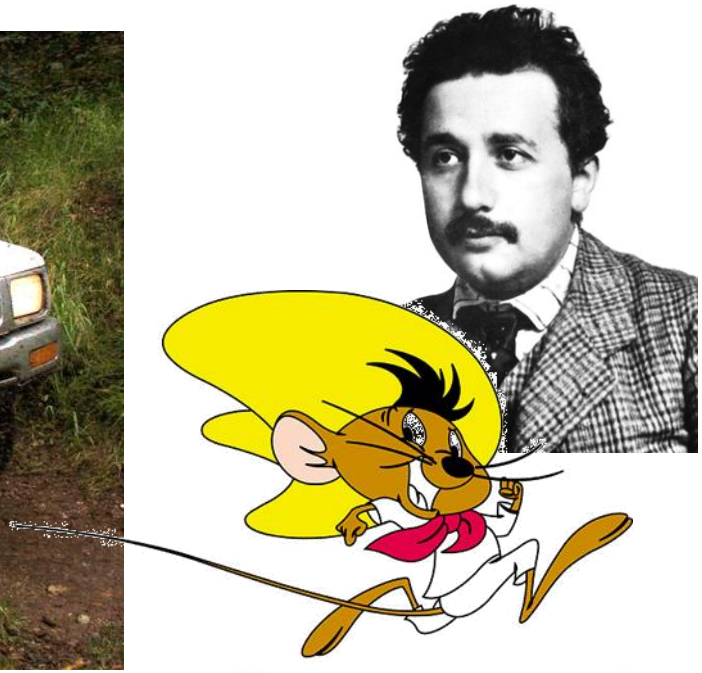
Several Approaches to ACS routines:

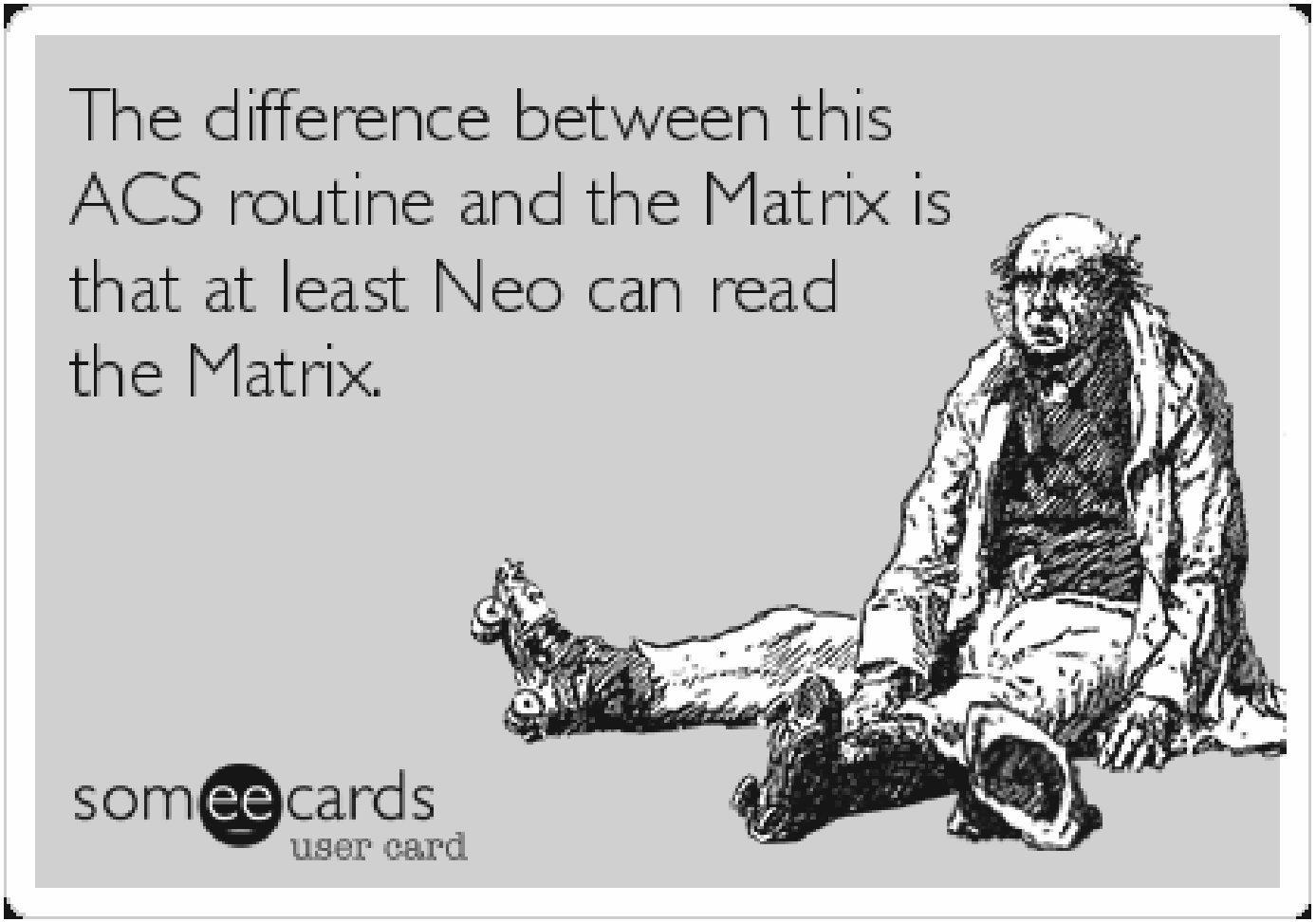
Speedy **Clever** **Maintainable**

(note: not mutually exclusive)

Best Method

- **No such thing** – very dependent on needs
- Good idea to lean toward **MAINTAINABLE**
- Don’t overcomplicate - KISS




The difference between this ACS routine and the Matrix is that at least Neo can read the Matrix.
someecards
user card

Part 2 – Preparation and Tools

Step 1 – Make a Pass through ACS

- **Make a copy of your ACS and SCDS**
  - For safety – do any edits on the copy
  - Example JCL at end of presentation
- Indent <optional>
  - Indent after every DO, IF, SELECT
  - Remove indent at every END, ELSE
  - Many text-editors have auto-indent
- Comment where you can -
  - /* document what you already know */
- Add WRITE statements to help trace routines
- Write down all the variables involved in decisions

Example

Old:

```
IF &DSN(1) = 'TEST' THEN DO
IF &UNIT ^= &VALID_UNITS
  THEN DO
SET &STORCLAS = ''
EXIT
END
ELSE SET &STORCLAS = 'TEST'
END
```

New:

```
/* Check for test data */
IF &DSN(1) = 'TEST' THEN DO
   /* Check if UNITs are VALID */
   IF &UNIT ^= &VALID_UNITS THEN DO
      /* if invalid, set NULL */
      SET &STORCLAS = ''
      WRITE 'INVALID UNIT TYPE'
      EXIT
   END
   ELSE DO
      SET &STORCLAS = 'TEST'
      WRITE 'TEST DATA'
   END
END
```

Step 2 – Chart the Logic

- Spreadsheet or Table
- Variables across the top
- Classes down the side

	A	B	C	D	E	F	G	H	I
1		&DSTYPE	&JOB	&DSORG	$DSN(2)	&DSTYPE	&SIZE	&USER	&DATACLAS
2	FLATSM								
3	FLATBIG								
4	LIBS								
5	VSAM								
6	CICSVS								
7	TEMPS								
8	JACKS								
9	WRONGDC								
10	ADMIN								
11									

Set Up Your Tools

- **Set Up Your Text Editor!**
  - Highlight DO/END pairs (ISPF HILITE LOGIC)
  - Highlight IF/ELSE pairs (ISPF HILITE LOGIC)
  - Enable auto-indent (if available)
  - Possibly build your own highlighting (if available)

- **ISMF Option 7.4 and NaviQuest – ACS Testing**
  - Run combinations of variables to see what comes out
  - Define sets of test cases and run them together
  - Compare results before and after ACS changes

Example of HILITE LOGIC

```
000001 PROC &STORCLAS                                                          .
000002 /*****************************************************************/      .
000003 /* MY ACS ROUTINES FOR STORCLAS                                  */       .
000004 /*****************************************************************/      .
000005 FILTLIST DBVOLS INCLUDE(IMS*,DB2*)                  /* ALL DATABASE VOLUMES */ .
000006                 EXCLUDE('IMS053','DB2007')                              .
000007 FILTLIST DBJOBS INCLUDE(IMS*,PROD*,ACCT*)              /* ALL DATA BASE JOBS */ .
000008 FILTLIST VALID_UNITS INCLUDE('3330','3340','3350',                      .
000009                      '3375','3380','3390','SYSDA','')                   .
000010 IF &UNIT ^= &VALID_UNITS THEN DO                                        .
000011    IF &DSN(1) = 'SYS1' THEN DO                                          .
000012       SET &STORCLAS = ''                                                .
000013       WRITE 'INVALID UNIT TYPE FOR SMS ALLOCATION'                      .
000014       EXIT                                                              .
000015    END                                                                  .
000016    ELSE DO                                                              .
000017       SET &STORCLAS = ''                                                .
000018       EXIT                                                              .
000019    END                                                                  .
000020 END                                                                     .
```

ACS Testing – ISMF 7.4

```
 Panel  Utilities  Help
_____________________________________________________________________________

                          ACS APPLICATION SELECTION
Command ===> ______________________________________________________________

Select one of the following options:
4  1. Edit            - Edit ACS Routine source code
   2. Translate       - Translate ACS Routines to ACS Object Form
   3. Validate        - Validate ACS Routines Against Storage Constructs
   4. Test            - Define/Alter Test Cases and Test ACS Routines
   5. Display         - Display ACS Object Information
   6. Delete          - Delete an ACS Object from a Source Control Data Set



If Display Option is Chosen, Specify:

  CDS Name  . . 'NEAL.SMS.SCDS'
                                (1 to 44 Character Data Set Name or 'Active')



Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

ACS Testing – Define Testcase

```
 Panel  Utilities  Help
 ____________________________________________________________________________

                              ACS TEST SELECTION
 Command ===> ____________________________________________________________

 Select one of the following Options:

   1  1. DEFINE      - Define an ACS Test Case
      2. ALTER       - Alter an ACS Test Case
      3. TEST        - Test ACS Routines


If DEFINE or ALTER Option is Chosen, Specify:

  ACS Test Library . . 'NEAL.SMS.ACS'                                  _
  ACS Test Member  . . TSTDA___
```

ACS Testing – Define Testcase

```
Panel  Utilities  Scroll  Help
──────────────────────────────────────────────────────────────────────────────
                              ACS TEST CASE ALTER                Page 1 of 4
Command ===>

ACS Test Library  : NEAL.SMS.ACS
ACS Test Member . : TSTDA

To ALTER ACS Test Case, Specify:
  Description ==> TESTING NEW RULE
  Expected Result DC = BDAMSTUF
  DSN (DSN/Collection Name) . . SPECIAL.JKTEST.NONVSAM
  MEMN (Object Name)  . . . . .
  Sysname . . .          Xmode . . . .            Def_dataclas  . .
  Sysplex . . .          ACSenvir  . . ALLOC      Def_mgmtclas  . .
  DD  . . . . .          Dataclas  . .            Def_storclas  . .
  Dsorg . . . . DA       Mgmtclas  . .            Dsntype . . . . .
  Recorg  . . .          Storclas  . .              If Ext  . . . .
  Dstype  . . .          Storgrp . . .            Seclabel  . . . .
  Dsowner . . .          Size  . . . .            Space_Type  . . .
  Expdt . . . .          Maxsize . . .            Second_Qty  . . .
  Retpd . . . .          Blksize . . .
Use ENTER to Perform Verification; Use DOWN Command to View next Panel;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

ACS Testing – Running a Test

```
                                ACS TEST SELECTION            MEMBER TSTDA SAVED
Command ===>

Select one of the following Options:

  3  1. DEFINE      - Define an ACS Test Case
     2. ALTER       - Alter an ACS Test Case
     3. TEST        - Test ACS Routines


If DEFINE or ALTER Option is Chosen, Specify:

  ACS Test Library . . 'NEAL.SMS.ACS'
  ACS Test Member  . . TSTDA
```

ACS Testing – Running a Test

```
                                TEST ACS ROUTINES
Command ===>

To Perform ACS Testing, Specify:

  CDS Name . . . . . . 'NEAL.SMS.SCDS'
                                (1 to 44 Character Data Set Name or 'Active')
  ACS Test Library . . 'NEAL.SMS.ACS'
  ACS Test Member  . . TSTDA    (fully or partially specified or * for all
                                members)
  Listing Data Set . . LISTING2
                                (1 to 44 Character Data Set Name or Blank)

Select which ACS Routines to Test:

  DC ===> Y  (Y/N)  SC ===> Y  (Y/N)  MC ===> Y  (Y/N)  SG ===> Y  (Y/N)



Use ENTER to Perform Verification and Testing;
Use HELP Command for Help; Use END Command to Exit.
```

ACS Testing - Results

```
                                ACS TESTING RESULTS

CDS NAME          : NEAL.SMS.SCDS
ACS ROUTINE TYPES: DC SC MC SG
ACS TEST LIBRARY : NEAL.SMS.ACS

  ACS TEST
  MEMBER       EXIT CODE   RESULTS
  ---------    ----------  ------------------------------------
  DESCRIPTION: TESTING NEW RULE
  EXPECTED RESULT: DC = BDAMSTUF
  TSTDA                 0  DC = BDAMSTUF
                        0  SC = SMS
    MSG : STORCLAS=SMS
                        0  MC = NULL VALUE ASSIGNED
                        0  SG = SGA
    MSG : STORGRP=SGA


ACS TESTING RC:  00
```

NaviQuest – ISMF 11

- Subset of ISMF Panels (option 11)
- Generates Libraries of Test Cases
  - Data from ISMF Lists, DCOLLECT data, SMF data
- Compare test results from before / after changes
- Generates reports
- **Can be done via BATCH**
- *For more info, see Share Seattle session 17045*

```
 6  Storage Group              - Specify Volume Names and Free Space Thresholds
 7  Automatic Class Selection - Specify ACS Routines and Test Criteria
 8  Control Data Set           - Specify System Names and Default Criteria
 9  Aggregate Group            - Specify Data Set Recovery Parameters
10  Library Management         - Specify Library and Drive Configurations
11  Enhanced ACS Management    - Perform Enhanced Test/Configuration Management
C   Data Collection            - Process Data Collection Function
G   Report Generation          - Create Storage Management Reports
L   List                       - Perform Functions Against Saved ISMF Lists
P   Copy Pool                  - Specify Pool Storage Groups for Copies
R   Removable Media Manager    - Perform Functions Against Removable Media
```

Computers are like
Old Testament gods;
lots of rules and no mercy.”

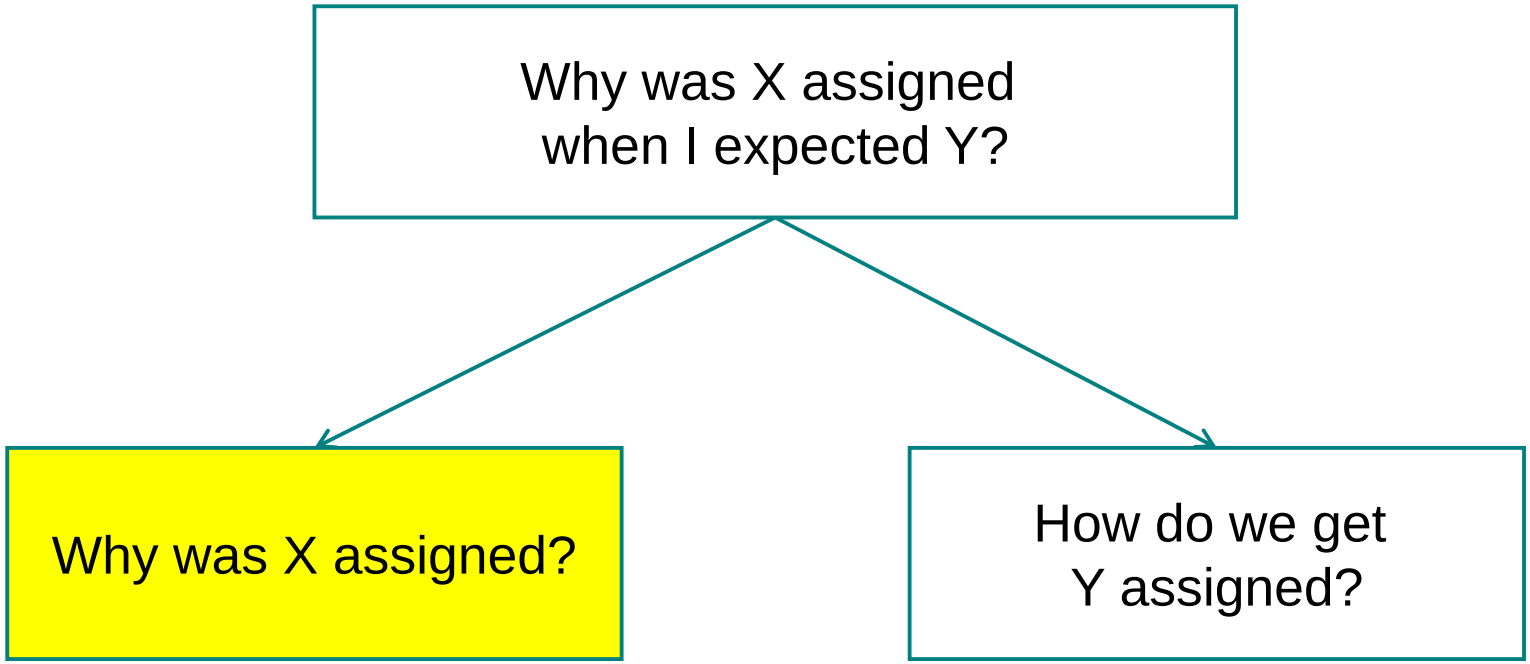
- Joseph Campbell

Part 3 – Scenarios

The Common Question

“Why was X assigned when I expected Y?”

Break it Down



Why was X assigned?

- Two+ ways to solve this.

Start at the Beginning

- Trace ACS
- Take each IF that matches
- End when you find the right SET

- **Pros / Cons**
  - Can be long
  - There could be multiple paths
  - May miss fall-through logic

Start at the End

- Find all SETs that match
- Work backwards, noting IF/SELECT requirements
- Eliminate redundant requirements

- **Pros / Cons**
  - Shorter
  - Need less DS information

Working Backwards - Example

- “WRONGDC” data class was incorrectly assigned
- DSNAME : 'SPECIAL.JKTEST.NONVSAM'

- **Step 1** – Find where “WRONGDC” is set..
- Two places, with IF just before each SET:
  - IF &DSN(2) = ‘JKTEST’
  - IF &DATACLAS = ‘’

- **Step 2** – Find IF statement before IF statements:
  - IF &JOB = &CICSJOB
  - ELSE (aka IF &JOB NE &CICSJOB)

```
PROC DATACLAS

FILTLIST ADMINS   INCLUDE('BOB','LARRY','MOE')
FILTLIST CICSJOBS INCLUDE(CICS*,'CISPECL')

/* DATA CLASSES */
/* FLATSM  - flat files, small, < 50mb */
/* FLATBIG - flat files, big > 50mb    */
/* LIBS    - PDS, PDSE                 */
/* VSAM    - VSAM files                */
/* CICSVS  - VSAM files for CICS       */
/* TEMPS   - temporary                 */
/* JACKS   - jacks testing DC          */
/* WRONGDC - invalid combination       */

IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMPS'

IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC'
   END
ELSE DO
   IF &DSORG = 'PS' THEN DO
      IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
      IF &SIZE > 50MB THEN SET &DATACLAS = 'FLATSM'
      ELSE SET &DATACLAS = 'FLATBIG'
   END

   IF &DSORG = 'PO' THEN SET &DATACLAS = 'LIBS'

   IF &USER = &ADMINS THEN DO
      SET &DATACLAS = 'ADMIN'
      EXIT
   END

   IF &DSORG = 'VS' THEN DO
      SET &DATACLAS = 'VSAM'
      IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
      END

   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
 END
 END
```

```
IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC'
   END
ELSE DO
   ..<omitted for brevity>..

   IF &DSORG = 'VS' THEN DO
      SET &DATACLAS = 'VSAM'
      IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
      END

   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
END
END
```

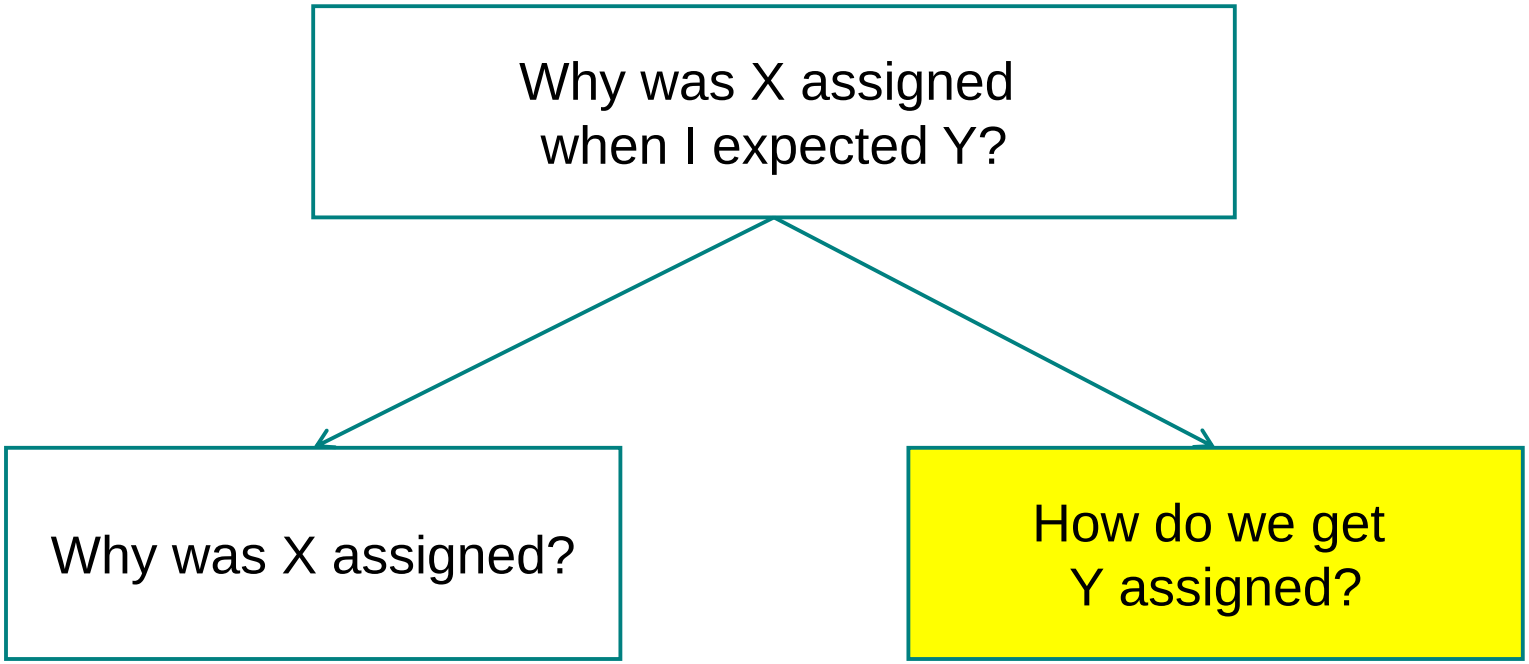
Working Backwards - Example

- Combine set of IF statements before SET to make a rule:
  - IF &JOB = &CICSJOB AND &DSN(2) /= 'JKTEST'
  - IF &JOB /= &CICSJOB AND &DATACLAS = '' (not set)

- **Now we have the 2 cases where WRONGDC gets set**
- Update the table:

	&JOB	&DSN(2)	&USER	&DATACLAS
WRONGDC	&CICSJOBS	NOT 'JKTEST'		
WRONGDC	NOT &CICSJOBS			null

Break it Down



How do we get Y assigned?

- Let's say the DS is supposed to have DC = 'JACKS'

- Identify all places JACKS is set:
  - IF &DSN(2) = 'JKTEST'
  - IF &JOB = 'JKTEST'

- Find Second-level tests:
  - IF &JOB = &CICSJOBS
  - IF &JOB /= &CICSJOBS

```
IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC‘
   END
ELSE DO
   ..<omitted for brevity>..

   IF &USER = &ADMINS THEN DO
      SET &DATACLAS = 'ADMIN'
      EXIT
   END
   ..<omitted for brevity>..

   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
END
END
```

Update Table Again

- Now we know how to get what we want:
  - &JOB = &CICSJOB AND &DSN(2) = 'JKTEST'
  - &JOB /= &CICSJOB AND &JOB = 'JKTEST'

	&JOB	&DSN(2)	&USER	&DATACLAS
WRONGDC	&CICSJOBS	NOT ‘JKTEST’		
WRONGDC	NOT &CICSJOBS			null
JACKS	&CICSJOBS	JKTEST		
JACKS	NOT &CICSJOB & JKTEST			

So what went wrong?

- **Sort Rules by &JOB** to consolidate
  - IF &JOB = &CICSJOB
    - IF &DSN(2) = 'JKTEST' – what we want
    - ELSE – what we don't want
  - IF &JOB /= &CICSJOB
    - IF &JOB = 'JKTEST' – what we want
    - IF &DATACLAS = '' - what we don't want

- We know &JOB was NOT &CICSJOB

	&JOB ▼	&DSN(2)	&USER	&DATACLAS
WRONGDC	NOT &CICSJOBS			null
JACKS	JKTEST			
~~**WRONGDC**~~	~~&CICSJOBS~~	~~NOT 'JKTEST'~~		
~~**JACKS**~~	~~&CICSJOBS~~	~~JKTEST~~		

What went wrong?

- Only two rules left:

	&JOB ▼	&DSN(2)	&USER	&DATACLAS
WRONGDC	NOT &CICSJOBS			null
JACKS	JKTEST			
~~**WRONGDC**~~	~~&CICSJOBS~~	~~NOT 'JKTEST'~~		
~~**JACKS**~~	~~&CICSJOBS~~	~~JKTEST~~		

- Difference is &JOB and &DATACLAS test…
- **However**, note that for &DATACLAS to be null (rule 1), these other rules could not have hit:
  - IF &DSORG = 'PS' or 'PO' or 'VS'
  - IF &USER = &ADMINS
  - IF &JOB = 'JKTEST'

```
IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC‘
   END
ELSE DO
   ..<omitted for brevity>..

   IF &USER = &ADMINS THEN DO
      SET &DATACLAS = 'ADMIN'
      EXIT
   END
   ..<omitted for brevity>..

   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
END
END
```

Finish the Table

- Update the Table with this new Info

	&JOB ▼	&DSN(2)	&USER	&DC	&DSORG
WRONGDC	NOT &CICSJOBS NOT ‘JKTEST’		NOT &ADMINS		NOT ‘PS’ or ‘PO’ or ‘VS’
JACKS	JKTEST		NOT &ADMINS		
~~**WRONGDC**~~	~~&CICSJOBS~~	~~NOT ‘JKTEST’~~			
~~**JACKS**~~	~~&CICSJOBS~~	~~JKTEST~~			

Getting it Right

- We Now Know:
  - Variable combinations for WRONGDC
  - Variable combinations for JACKS

- So to get JACKS, what needs to change?

- We're still missing some information…
- Gather available information about the job that ran:
  - &JOB = 'MYTEST'
  - &DSN(2) = 'JKTEST'
  - &USER = 'JACK'
  - &DATACLAS = TBD
  - &DSORG = 'DA' (BDAM)

Add allocation knowns

- Add a row to the table with known information:

	&JOB ▼	&DSN(2)	&USER	&DC	&DSORG
WRONGDC	NOT &CICSJOBS NOT 'JKTEST'		NOT &ADMINS		NOT 'PS' or 'PO' or 'VS'
JACKS	JKTEST		NOT &ADMINS		
MYDS	**'MYTEST'**	**'JKTEST'**	**'JACK'**	**?**	**'DA'**

- FILTLIST reference:
  - FILTLIST ADMINS INCLUDE('BOB','LARRY','MOE')
  - FILTLIST CICSJOBS INCLUDE(CICS*,'CISPECL')

What went wrong?

- Jack was doing some CICS testing under a different job name than usual: **‘MYTEST’ instead of ‘JKTEST’**
- To get DC = 'JACKS', for NON-CICS, the **JOBNAME needs to be 'JKTEST'**
- This was just a simple example – working through the logic this way can help with bigger, more complex ACS
- Use ACS testing to verify results!

ACS Testing Example

```
                              ACS TEST CASE ALTER                 Page 1 of 4
Command ===>

ACS Test Library  : NEAL.SMS.ACS
ACS Test Member . : TEST1

To ALTER ACS Test Case, Specify:
  Description ==> TESTING JKTEST
  Expected Result
  DSN (DSN/Collection Name) . . SPECIAL.JKTEST.NONVSAM
  MEMN (Object Name)  . . . . .
  Sysname . . .          Xmode . . . .            Def_dataclas  . .
  Sysplex . . .          ACSenvir  . . ALLOC      Def_mgmtclas  . .
  DD  . . . . .          Dataclas  . .            Def_storclas  . .
  Dsorg . . . . DA       Mgmtclas  . .            Dsntype . . . . .
  Recorg  . . .          Storclas  . .              If Ext  . . . .
 Job  . . . . . MYTEST   Pgm . . . . .            Vol . . . . .
```

Run once with MYTEST (TEST1)
Run again with JKTEST (TEST2)

ACS Testing Results

```
                              ACS TESTING RESULTS

CDS NAME          : NEAL.SMS.SCDS
ACS ROUTINE TYPES: DC
ACS TEST LIBRARY : NEAL.SMS.ACS

  ACS TEST
  MEMBER       EXIT CODE   RESULTS
  ---------    ----------  ------------------------------------
  DESCRIPTION: TESTING JKTEST
  EXPECTED RESULT:
  TEST1                 0  DC = WRONGDC


ACS TESTING RC:  00
```

```
  ACS TEST
  MEMBER       EXIT CODE   RESULTS
  ---------    ----------  ------------------------------------
  DESCRIPTION: TESTING JKTEST
  EXPECTED RESULT:
  TEST2                 0  DC = JACKS


ACS TESTING RC:  00
```

A few notes:

- Logic Chart works best where all SETs are mutually exclusive
- If you have fall-through logic, you may consider using an ORDER field as well to sequence logic

- Try to avoid testing &DATACLAS in DC routines
  - Often a failsafe, but complicates logic
  - Or remove / add WRITE when debugging

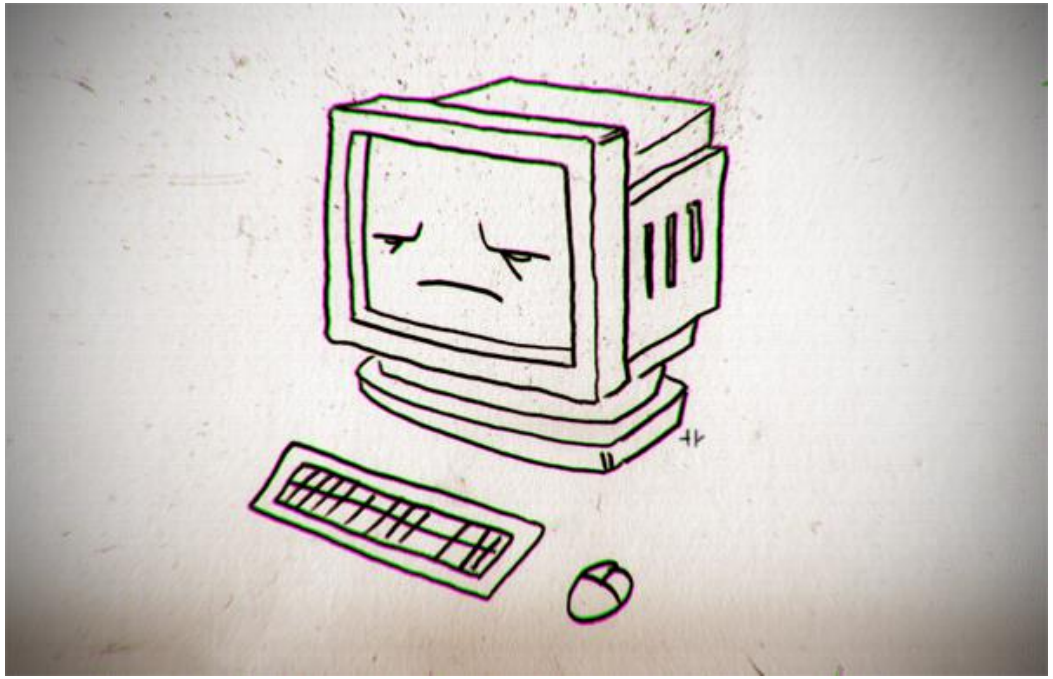
- Adding EXIT after each SET is good practice, but can complicate logic

- Strategic WRITE statements can help demonstrate decision logic

- Use ACS testing to run your modified ACS to track logic

Summary of Scenario 1:

- Tracked backwards from the SET
- Created a TABLE or MAP to detail conditions / rules
- Compared actual rules to expected result
- Found culprit
- Tested changes with ACS Tester



“To err is human - and to blame it on a computer is even more so.”

- Robert Orben

Scenario 2

“We added a class, and now we need to update the routines.”

Updating Routines - Overview

- Varying difficulty, depending on new rules
- Simpler with a table or MAP

- Basic steps:
  - Copy and work on copy!
  - Find logic section that matches
  - Insert new rule
  - Test, fix, test, fix

Updating Routines – Don't

- Don't just add a simple rule to the beginning:
  - IF &DSORG=”VS” THEN SET &DATACLAS = “VSAM”
  - You’ll bypass all your old routines!

- Don't just add a simple rule to the end:
  - It might never get run
  - It might overwrite a different set (watch out for fall through)

- Don't forget to comment
- Don't use variables that are not used anywhere else
  - You'll end up with non-exclusive paths
  - Try to use consistent logic

Understand the Logic

- How do the routines select classes?
- What variables are used?
- Build a table or MAP:

	A	B	C	D	E	F	G	H	I
1	CLASS	&DSTYPE	&JOB	&DSORG	$DSN(2)	&DSTYPE	&SIZE	&USER	&DATACL
2	WRONGDC		NOT &CICSJOBS and NOT JKTEST	DA				NOT &ADMIN	
3	FLATSM		NOT &CICSJOBS	PS			< 50M		
4	FLATBIG		NOT &CICSJOBS	PS			>= 50M		
5	LIBS		NOT &CICSJOBS	PO					
6	VSAM		NOT &CICSJOBS	VS					
7	TEMPS	TEMP	NOT &CICSJOBS	PS | VS					
8	ADMIN		NOT &CICSJOBS					&ADMIN	
9	JACKS		JKTEST					NOT &ADMIN	
10	CICSVS		&CICSJOB	VS					
11	JACKS		&CICSJOB		JKTEST				
12	WRONGDC		&CICSJOB						
13									

Adding Rules - Example

- We want to add a new class **BDAMSTUF**
  - Non-CICS
  - DSORG='DA'
- Add to the MAP:

	A	B	C	D	E	F	G	H	I
1	CLASS	&DSTYPE	&JOB	&DSORG	$DSN(2)	&DSTYPE	&SIZE	&USER	&DATACL
2	WRONGDC		NOT &CICSJOBS and NOT JKTEST	DA				NOT &ADMIN	
3	FLATSM		NOT &CICSJOBS	PS			< 50M		
4	FLATBIG		NOT &CICSJOBS	PS			>= 50M		
5	LIBS		NOT &CICSJOBS	PO					
6	VSAM		NOT &CICSJOBS	VS					
7	TEMPS	TEMP	NOT &CICSJOBS	PS | VS					
8	ADMIN		NOT &CICSJOBS					&ADMIN	
9	JACKS		JKTEST					NOT &ADMIN	
10	CICSVS		&CICSJOB	VS					
11	JACKS		&CICSJOB		JKTEST				
12	WRONGDC		&CICSJOB						
13	BDAMSTUF		NOT &CICSJOBS	DA					
14									

Adding Rules - Example

- Compare to other rules by Variable
  - Only two in play: **&JOB and &DSORG**
- Fits in our NOT &CICSJOB section
- Fits next to other &DSORG tests

	A	B	C	D	E	F	G	H	I
1	CLASS	&DSTYPE	&JOB	&DSORG	$DSN(2)	&DSTYPE	&SIZE	&USER	&DATACLAS
2	CICSVS		&CICSJOB	VS					
3	JACKS		&CICSJOB		JKTEST				
4	WRONGDC		&CICSJOB						
5	JACKS		JKTEST					NOT &ADMIN	
6	BDAMSTUF		NOT &CICSJOBS	DA					
7	LIBS		NOT &CICSJOBS	PO					
8	FLATSM		NOT &CICSJOBS	PS			< 50M		
9	FLATBIG		NOT &CICSJOBS	PS			>= 50M		
10	TEMPS	TEMP	NOT &CICSJOBS	PS | VS					
11	VSAM		NOT &CICSJOBS	VS					
12	ADMIN		NOT &CICSJOBS					&ADMIN	
13	WRONGDC		NOT &CICSJOBS and NOT JKTEST	DA				NOT &ADMIN	

```
IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMPS'

IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC‘
   END
ELSE DO
   IF &DSORG = 'PS' THEN DO
      IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
      IF &SIZE > 50MB THEN SET &DATACLAS = 'FLATSM'
      ELSE SET &DATACLAS = 'FLATBIG'
   END

   IF &DSORG = 'PO' THEN SET &DATACLAS = 'LIBS'

   IF &USER = &ADMINS THEN DO
      SET &DATACLAS = 'ADMIN'
      EXIT
   END

   IF &DSORG = 'VS' THEN DO
      SET &DATACLAS = 'VSAM'
      IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
    END

   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
END
```

NOT &CICSJOB

&DSORG Section

Adding Rules - Example

- Insert the new rule:
  - IF &DSORG='DA' THEN SET &DATACLAS = 'BDAMSTUF'
- This many IF statements should be a SELECT
- Use the best practices already in use in the routine
  - Don't change practices unless you're ready to rewrite
- Note your update in the CHANGELOG
  - If one doesn't exit, CREATE ONE!

```
PROC DATACLAS

FILTLIST ADMINS   INCLUDE('BOB','LARRY','MOE')
FILTLIST CICSJOBS INCLUDE(CICS*,'CISPECL')

/* DATA CLASSES */
/* FLATSM  - flat files, small, < 50mb */
/* FLATBIG - flat files, big > 50mb    */
/* LIBS    - PDS, PDSE                 */
/* VSAM    - VSAM files                */
/* CICSVS  - VSAM files for CICS       */
/* TEMPS   - temporary                 */
/* JACKS   - jacks testing DC          */
/* WRONGDC - invalid combination       */

/* CHANGE LOG                                        */
/* 2013.08.13 – Added BDAMSTUF to NON-CICS group     */
/*            - Also switched from IF to SELECT block */
```

```
IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMPS'

IF &JOB = &CICSJOBS THEN DO
   IF &DSORG = 'VS' THEN SET &DATACLAS = 'CICSVS'
   IF &DSN(2) = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   ELSE SET &DATACLAS = 'WRONGDC‘
   END
ELSE DO
    SELECT ( &DSORG )
       WHEN ('PS') DO
          IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
          IF &SIZE > 50MB THEN SET &DATACLAS = 'FLATSM'
          ELSE SET &DATACLAS = 'FLATBIG'
       END
       WHEN ('PO') SET &DATACLAS = 'LIBS'
       WHEN ('VS') DO
          SET &DATACLAS = 'VSAM'
          IF &DSTYPE = 'TEMP' THEN SET &DATACLAS = 'TEMP'
       END
       WHEN ('DA') SET &DATACLAS = 'BDAMSTUF'
       OTHERWISE   SET &DATACLAS = 'WRONGDC'
    END /* END SELECT &DSORG */

   IF &USER = &ADMINS THEN DO
      SET &DATACLAS = 'ADMIN'
      EXIT
   END
   IF &JOB = 'JKTEST' THEN SET &DATACLAS = 'JACKS'
   IF &DATACLAS = '' THEN SET &DATACLAS = 'WRONGDC'
END
END
```

Test Before / After

- Defined testcase with:
  - DSN: SPECIAL.JKTEST.NONVSAM
  - DSORG: DA
  - ACSENVIR: ALLOC
  - JOB: MYTEST
- BEFORE:

```
ACS TEST
  MEMBER        EXIT CODE   RESULTS
  ---------     ----------  ------------
  DESCRIPTION: TESTING NEW RULE
  EXPECTED RESULT:
  TSTDA                  0  DC = WRONGDC
```

- AFTER:

```
ACS TEST
  MEMBER        EXIT CODE   RESULTS
  ---------     ----------  ------------
  DESCRIPTION: TESTING NEW RULE
  EXPECTED RESULT:
  TSTDA                  0  DC = WRONGDC
```

Test, Fix, Test

- **Use the ACS routine tester!**
- Run a series of tests to verify new changes work
- Run a series of tests to ensure old rules still work
- If not, figure out why and fix
- Repeat
- Test again after activation

- You can use NaviQuest to build suites of tests
  - *See Seattle session 17045 – NaviQuest: Streamlining SMS*

Adding Rules - Summary

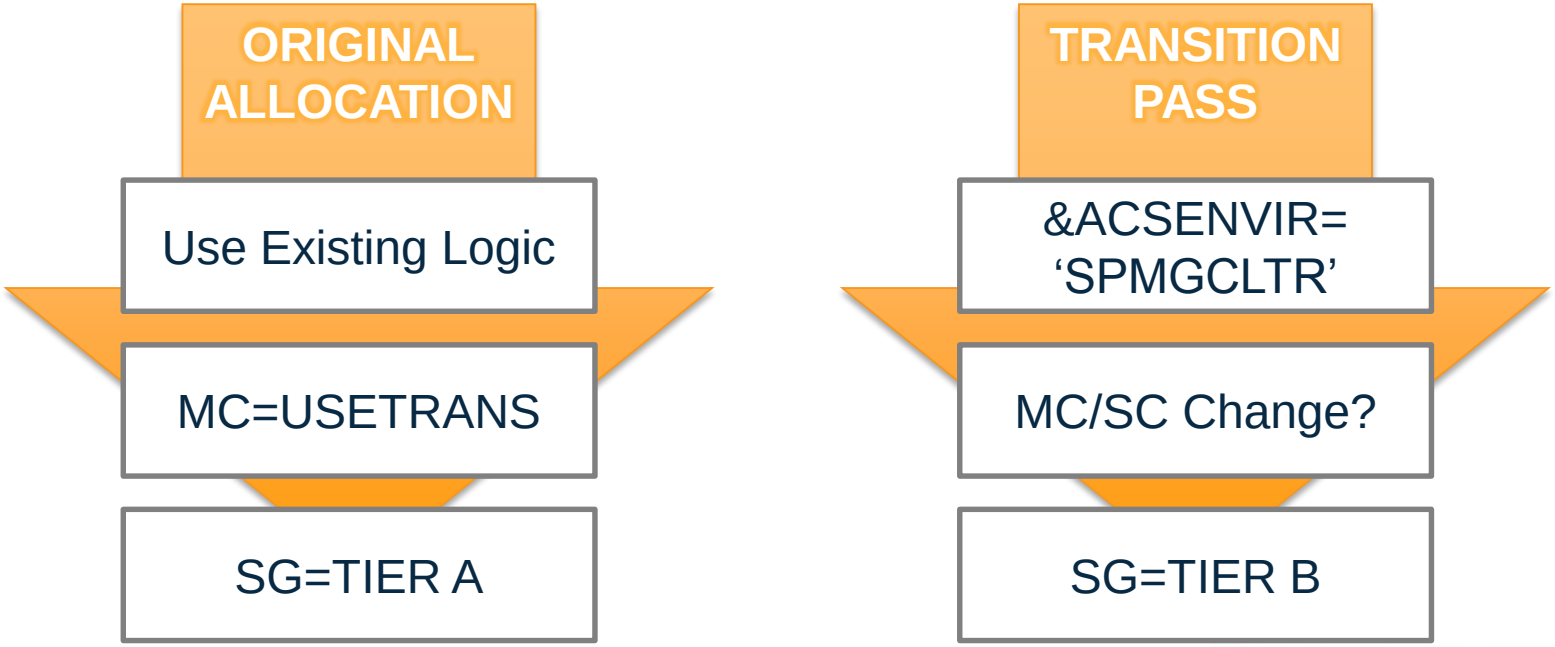
- Make a copy
- Understand the logic
- Find relevant section
- Update CHANGELOG
- Add rule
- Avoid breaking anything
- Test, test, test

Part 4 – Preparing for Transitions

Some tips and things to consider

HSM Transitions

- HSM Transitions feature will re-run ACS routines to assign to new classes (except DC)
- &ACSENVIR = ‘SPMGCLTR’
  - SPace ManaGement CLass TRansitions



HSM Transitions

```
SELECT (&ACSENVIR)
   WHEN('SPMGCLTR') DO
      /* Transition from one to another */
      /* Re-use previous logic, but set phase 2 class */
      IF( &DSN(2) = 'LOGDATA') THEN SET &STORGRP = …

      /* Base decision off first assignment (not recommended) */
      IF( &DATACLAS = 'PASS1' ) THEN SET &STORGRP = …

      /* Completely new logic */
      WRITE 'DONT FORGET THE WRITE STATEMENTS'
   END
   OTHERWISE DO
      /* OLD LOGIC, UNTOUCHED */
      IF …
   END
END
```

Part 5 – Considering a Rewrite?

Some tips and things to consider

Rewrite Process

- Make sure you understand the logic
- Have clearly defined rules BEFORE you start
- Consider how things might change in the future

- Group rules by variable in order of importance / frequency
- Code rules using nested SELECT / WHEN

- Create a suite of ACS Test cases (see NaviQuest)
- Run before / after to ensure logic works the same

- COMMENT your logic flow

Rewrite DOs

- Prefer SELECT over IF
- Try to make logic mutually exclusive (one big decision tree)
- Code rules in order of most to least specific
- Use FILTLIST names that are helpful
- Use COPYFILT to keep common FILTLISTS the same
- EXIT after a SET
- Use WRITE statements liberally
- Test repeatedly

Rewrite DON'Ts

- Use lists of IF statements (resist nesting)
- Mirror FILTLISTs between routines unless needed
- Base your logic of the DC or SC assignment
- Use huge numbers of variable
- Work on active SCDS / ACS
- Forget to Test, Test, Test!

References

- ***DFSMS Implementing System-Managed Storage***
  - SC26-7407
- ***DFSMSdfp Storage Administration***
  - SC26-7402

Understanding Someone Else's ACS Routines

Neal Bohling and Chris Taylor
IBM

August 13, 2015
Session# 17836

Copyright (c) 2015 by SHARE Inc. Except where otherwise noted, this work is licensed under http://creativecommons.org/licenses/by-nc-sa/3.0/

Notices & Disclaimers

Product information and data has been reviewed for accuracy as of the date of initial publication. Product information and data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the products and/or programs described herein at any time without notice.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Consult your local IBM representative or IBM Business Partner for information about the product and services available in your area.

Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead. It is the user's responsibility to evaluate and verify the operation of any non-IBM product, program or service.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS"WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR INFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.

Notices & Disclaimers

The performance data contained herein was obtained in a controlled, isolated environment. Actual results that may be obtained in other operating environments may vary significantly. While IBM has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere.

The responsibility for use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's or user's ability to evaluate and integrate them into their operating environment. Customers or users attempting to adapt these techniques to their own environments do so at their own risk. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO,
LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY.

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

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Trademarks