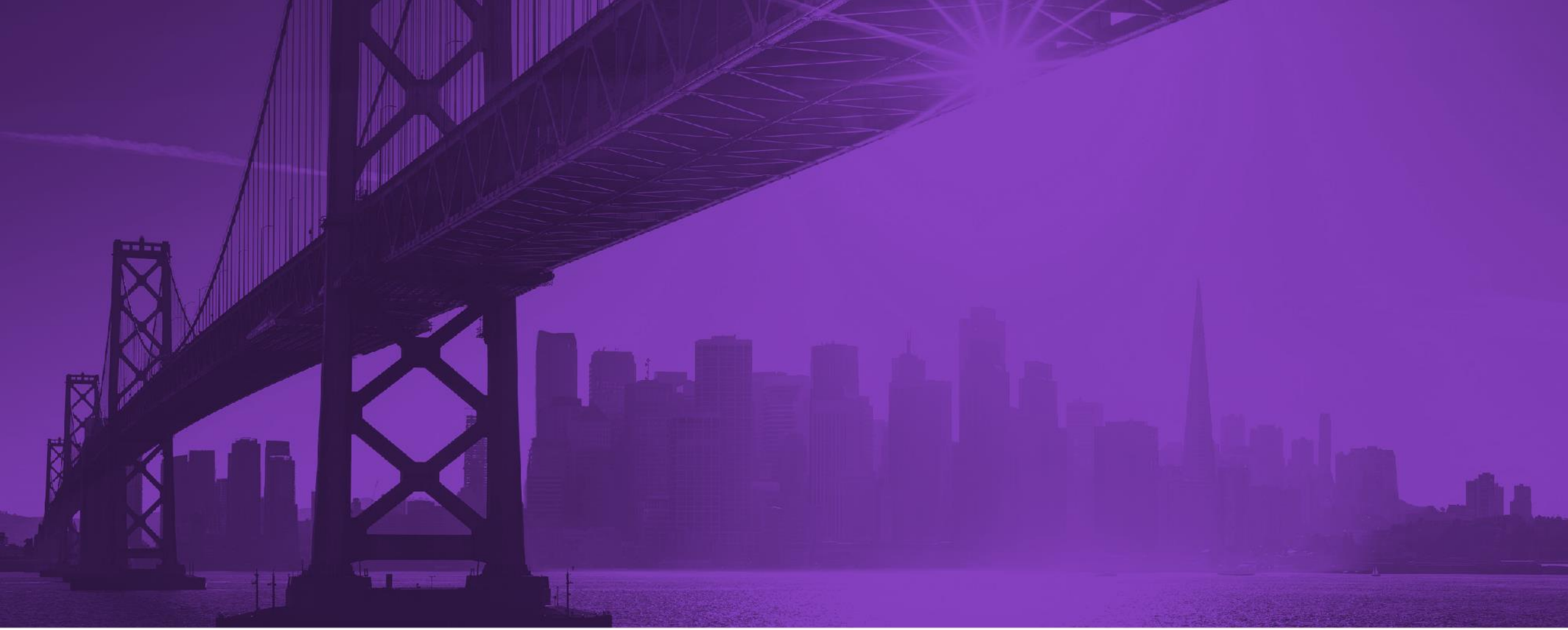


Session NH: Taming the Wild West

Life Cycle Management and Testing within IBM Workload Scheduler for z/OS

Taming the Wild West

- Using IWSz during the full test cycle can bring great power to your organisation
  - Consistency, speed and accuracy of tests can be improved
  - The scheduling, as well as the application gets tested
  - Developers gain more understanding of the Operational side
  - And don’t waste time reinventing the wheel

- With great power can come great irresponsibility
  - Test environments are usually uncontrolled, and unautomated
  - Human nature will take the easiest route - Not necessarily the best route
  - Le Bon’s crowd theory predicts a decline in personal responsibility
    i.e. The more users an environment has, the less each individual will feel inclined to keep it tidy and up to date

- The Wild West needs some law makers, and a Sheriff
  - Variables, JCL tailoring, Exits and WAPL can help round ‘em up

The Sheriff's charter

- To keep test and production as similar as possible
  - No point testing if your tests don't represent production
  - Keep physical changes between environments to a minimum
  - The closer the environments are, the more accurate the testing will be
  - And the happier the auditors will be
- Allow each environment to point at separate data
  - Ensure the JCL points to the data related to the environment being tested
  - Use security to enforce the separation
- Allow for agreed and understood processing differences between environments
  - Certain jobs will not run in certain environments
  - Housekeeping is often different between environments
  - Scheduling objects should enable these differences
- To not need full time operators for testing – Have true Dev Ops

If I’ve got a star does that make me Sheriff?

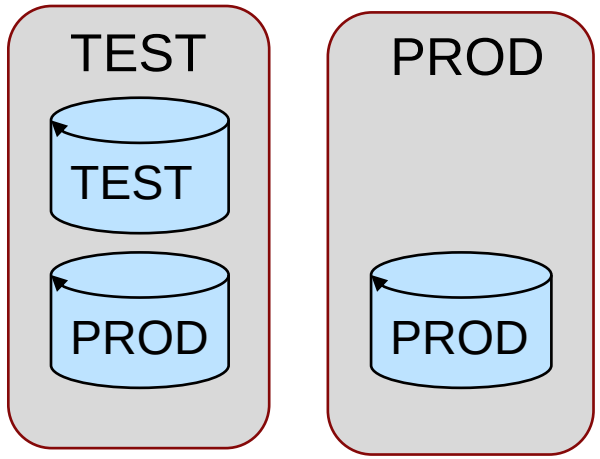
The Theory

Life cycle adaption – Migration time versus Run time

- At migration time – objects can be automatically adapted between environments
  - Naming conventions can create unique object names
  - WAPL, Urban Code Deploy and various other tools can enable this
- At run time – objects remain largely unchanged between environments
  - JCL variables and tailoring directives can point at different files and systems
  - JCL variables and tailoring directives can omit steps or even entire jobs
  - WAPL steps can be included to make more complex variations
  - Specific workstations can be CPU in one environment, but dummy in another to omit jobs
  - From 9.3 NOP, HOLD and Runcycle Groups can vary execution

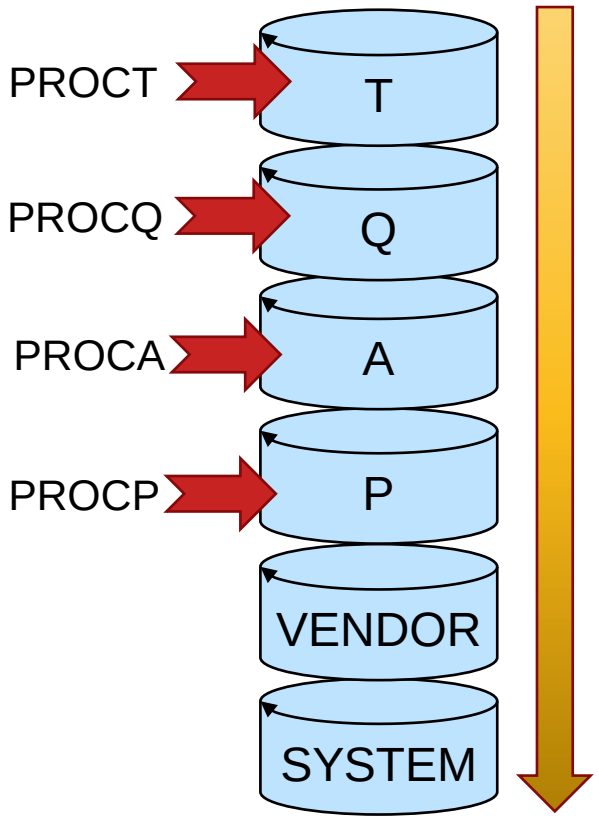
Job Concatenations

- Concatenations
  - for JCL only, other objects must be fully replicated to each controller
  - The Job-Library-Read exit can point to different concatenations
  - Only “changed jobs” need to be stored in each environment, picking up the Prod version if not overridden
- For concatenations to work, member name MUST be the same in all environments
  - This does NOT mean the jobname must be the same
  - The Job-Library-Read exit can read from alternate member names
  - The &OJOBNAME variable can handle the job name
- Job-Library-Read exit
  - This can be pointed at alternative concatenations per environment
  - This can point at an alternate member name, other than the job name
- Concatenations do not work for IWSz database objects
  - They must ALL exist in each controller if they are to be used



Procedure Concatenations

- JES2 allows procedure library concatenations
  - With most customers, procedure names tend to NOT have a life cycle based name
  - Usually because symbolic parameters are used to make them lifecycle phase neutral
  - You CAN have a concatenation for each phase of your life cycle
- If your life cycle goes -
  Unit Test (T) – System Test (Q) – Acceptance Test (A) – Production (P)
  - PROCT could be T.PROCLIB,Q.PROCLIB,A.PROCLIB,P.PROCLIB,VENDOR.PROCLIB,SYSTEM.PROCLIB
  - PROCQ could be Q.PROCLIB,A.PROCLIB,P.PROCLIB,VENDOR.PROCLIB,SYSTEM.PROCLIB
  - PROCA could be A.PROCLIB,P.PROCLIB,VENDOR.PROCLIB,SYSTEM.PROCLIB
  - PROCP could be P.PROCLIB,VENDOR.PROCLIB,SYSTEM.PROCLIB
- Each proc only needs to exist at the levels where it changes
  - Testing early in the life cycle is automatically done with changes progressing through the life cycle
  - You could also create exclusive proclibs for when you don't want to pick up other environments
    e.g. PROCTX could be T.PROCLIB,VENDOR.PROCLIB,SYSTEM.PROCLIB



A good naming convention

- The core of self translating JCL is having a good naming convention, including things like
  - Life cycle phase – Covers primary differences between each phase of the lifecycle
  - Business application – e.g. PY=Payroll, allows for application level settings
  - Region – Could be geographical regional differences, or simply a separate testing region
- You can use Application name or a Variable table to drive this information
  - Using application name is fixed, and would be extracted in the job by JCL directives
  - Using a variable table allows for run time variation, choosing different tables with different values
- A good naming convention for variables and tables is also essential
  - A prefix to keep IWSz variables away from JCL symbols
  - Use of multiple tables allows for variable concatenations
  - You can set defaults for a life cycle phase, and override them at application or region level
- If you refer JCL symbolic parameter values in executing JCL
  - Consider setting VARFAIL – quick and easy but can mask errors
  - Add empty variables for referenced JCL symbolic parameters with REQD=N

Lifecycle neutral job JCL

- Variables can be used to make JCL universal
  - For environmentally named IWSz objects the input table(s) can be assigned automatically
  - JCL SETVAR directives can pick lifecyle elements out of the application or variable table name
  - Otherwise tables can be assigned at run time
- Variable line elements can be tricky to handle
  e.g. An element that is 2 lines of JCL in one environment, but 1 line in another
  - But includes can be used to manage those
- Neutral JCL enables use of concatenations to pick up production versions in lower environments
  - Where source management has adapted JCL concatenations cannot be used
  - Environment specific member names cannot do this either
  - But sometimes this is the only choice

Production names versus Environment specific names

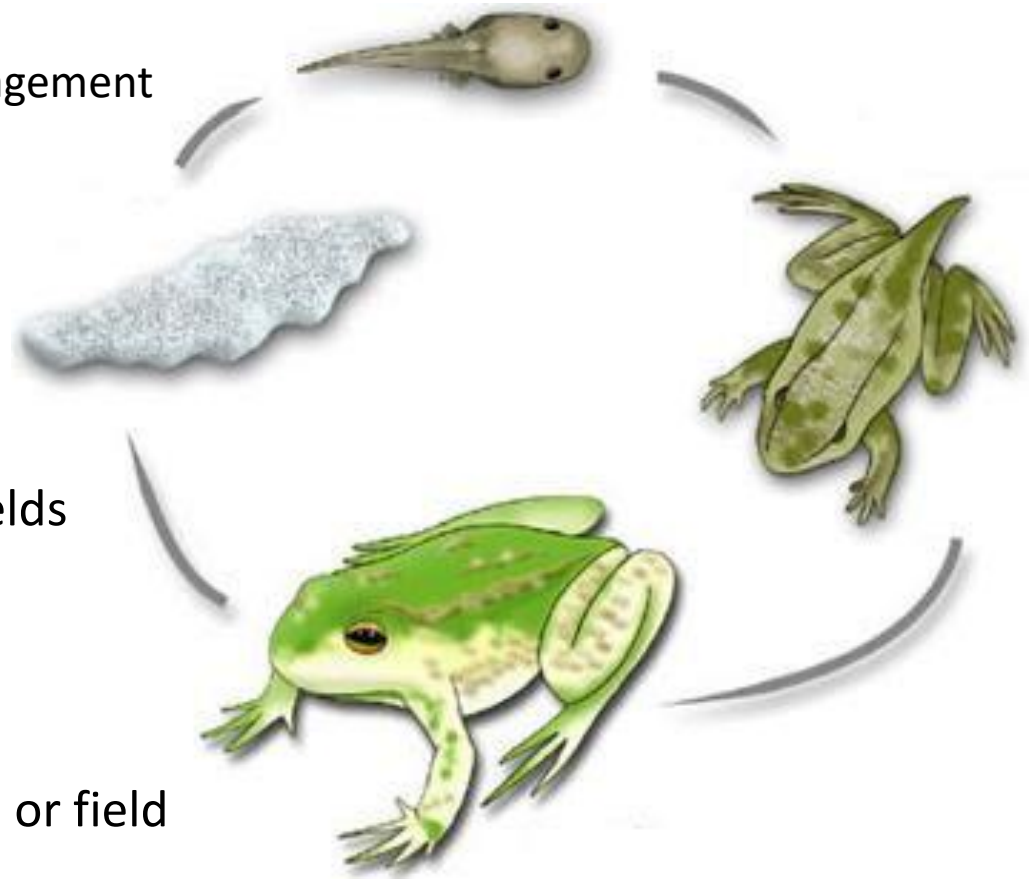
Object	Fixed names	Environment specific names
Jobname	- Delays in JES2 - Unable to discern environment running	+ Environments can run in parallel without delay + Clarity of what is running
JCL member name	+ Concatenation is possible - May need additional info in application	- All jobs must exist in each environment + Picked up automatically by application
Application	- Unable to discern environment running - Run time data needed to drive exit - Run time data needed to select variable table - **Dependencies may cross environments**	+ Clarity of what is running + Exit driven automatically + Table selected automatically + Dependencies match environments
Special resource	- **Environment clash in IWSz**	+ Environments can run in parallel
Workstation name	+ Destinations/Type can be adapted - Environments can't vary in one controller	- Needs more workstations per controller

Doc Holiday's Office

The Practice

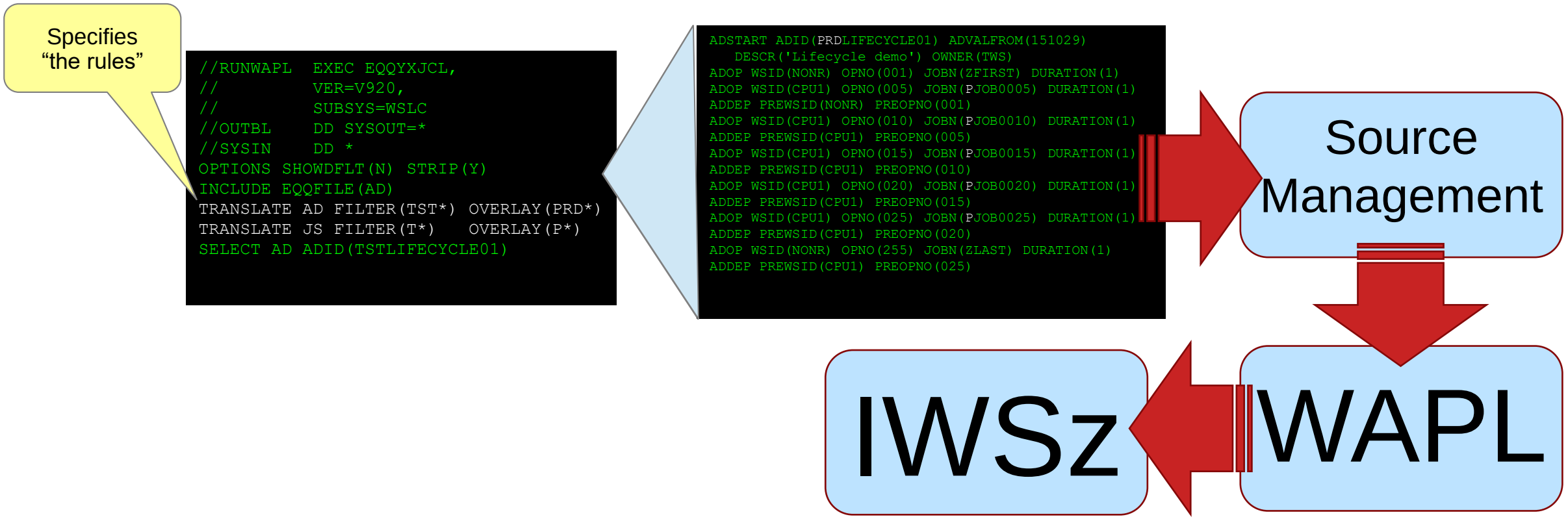
Lifecycle management with WAPL

- WAPL can transform objects with the TRANSLATE command
  - The TRANSLATE command is NOT a golden bullet
  - Good naming conventions are essential for lifecycle management
- There are two ways to translate
  - By LIST, every object has an OLD and NEW value
  - By RULES, every object has a FILTER and an OVERLAY
- TRANSLATE can handle types of field and individual fields
  - Types include – AD, CL, JS, OW, PR and SR
  - Individual fields, specified by their name
- There can only be one TRANSLATE statement per type or field
  - Each statement can specify OLD/NEW or FILTER/OVERLAY pairs
  - Or specify a LIST or RULES member



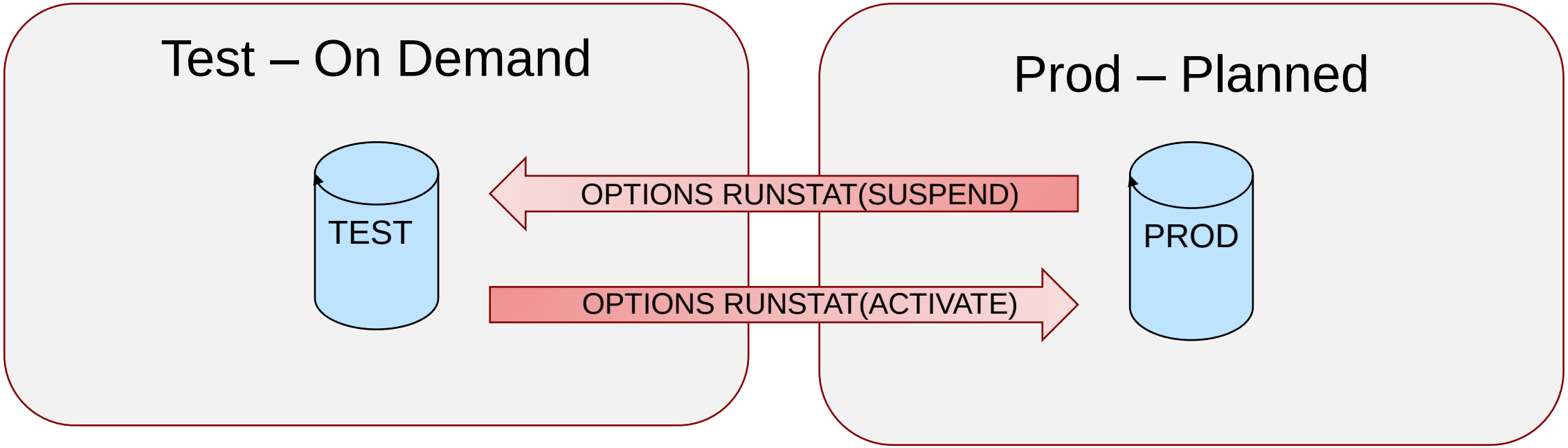
TRANSLATE in action

- All application and job names translated by two commands
- The TRANSLATE precedes the SELECT



Controlling the running

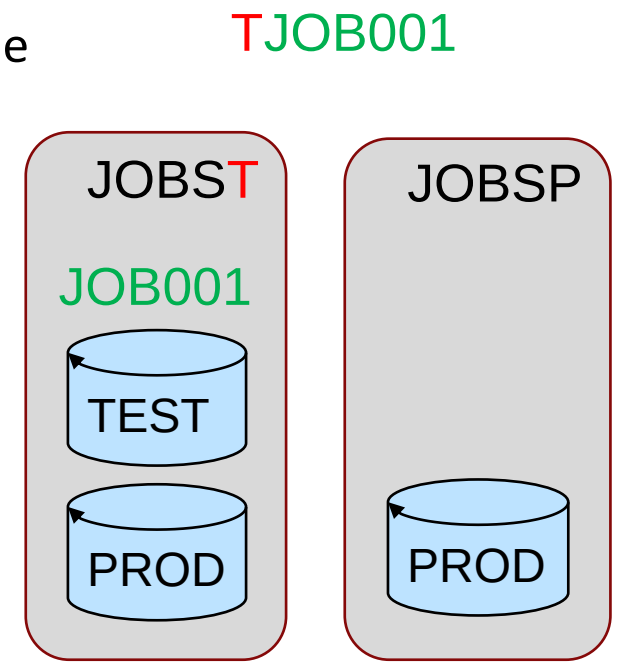
- PENDING status stops and application being planned, but also stops it being submitted
  - Putting run cycles out of effect ONLY stops an application being planned



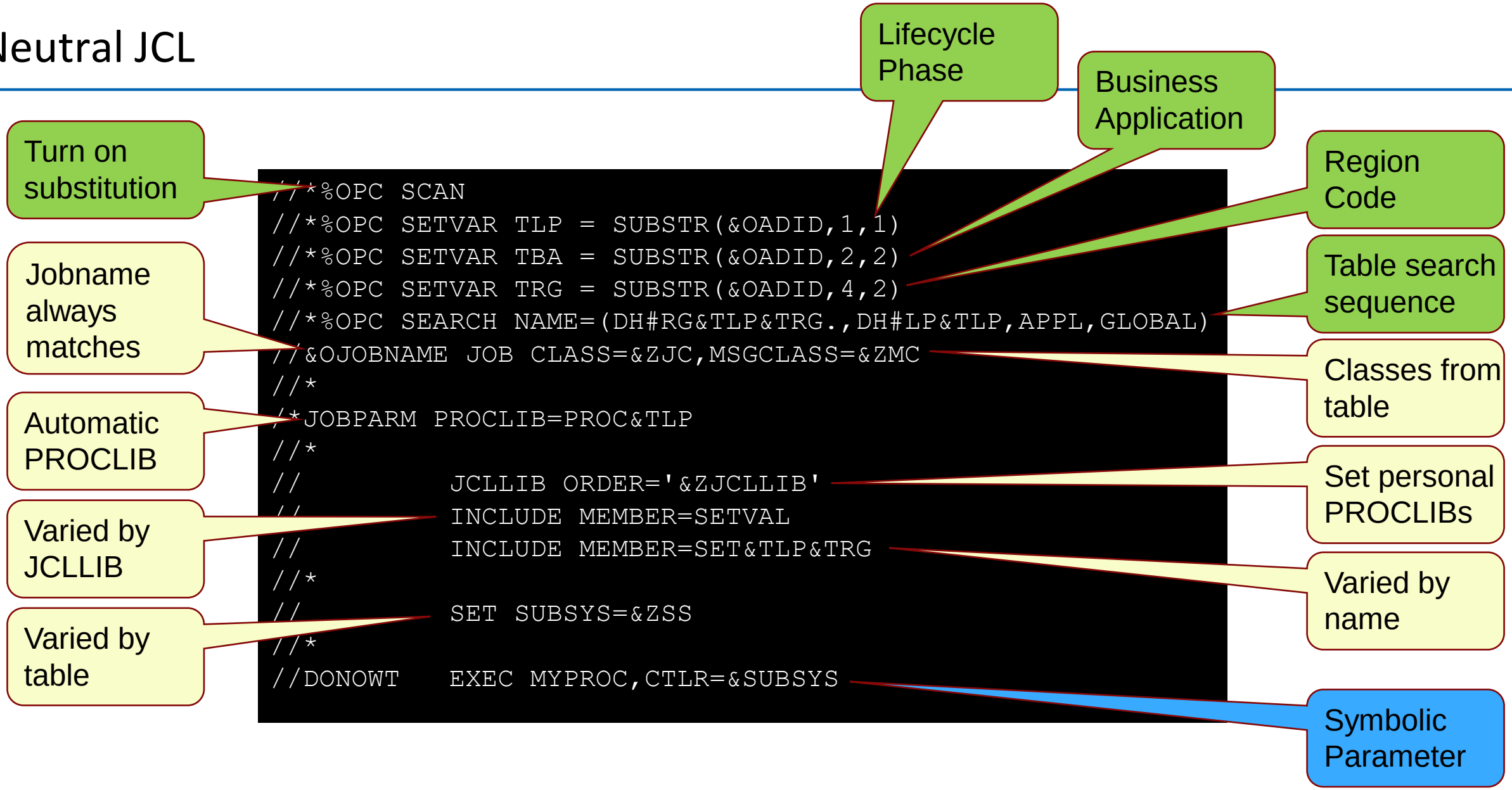
- OPTIONS RUNSTAT alters status of Run cycles
  - SUSPEND puts the Out of Effect for low date (72/01/01)
  - ACTIVATE takes any low date out of effect run cycles and makes them active again

JCL selection via Job-Library-Read exit

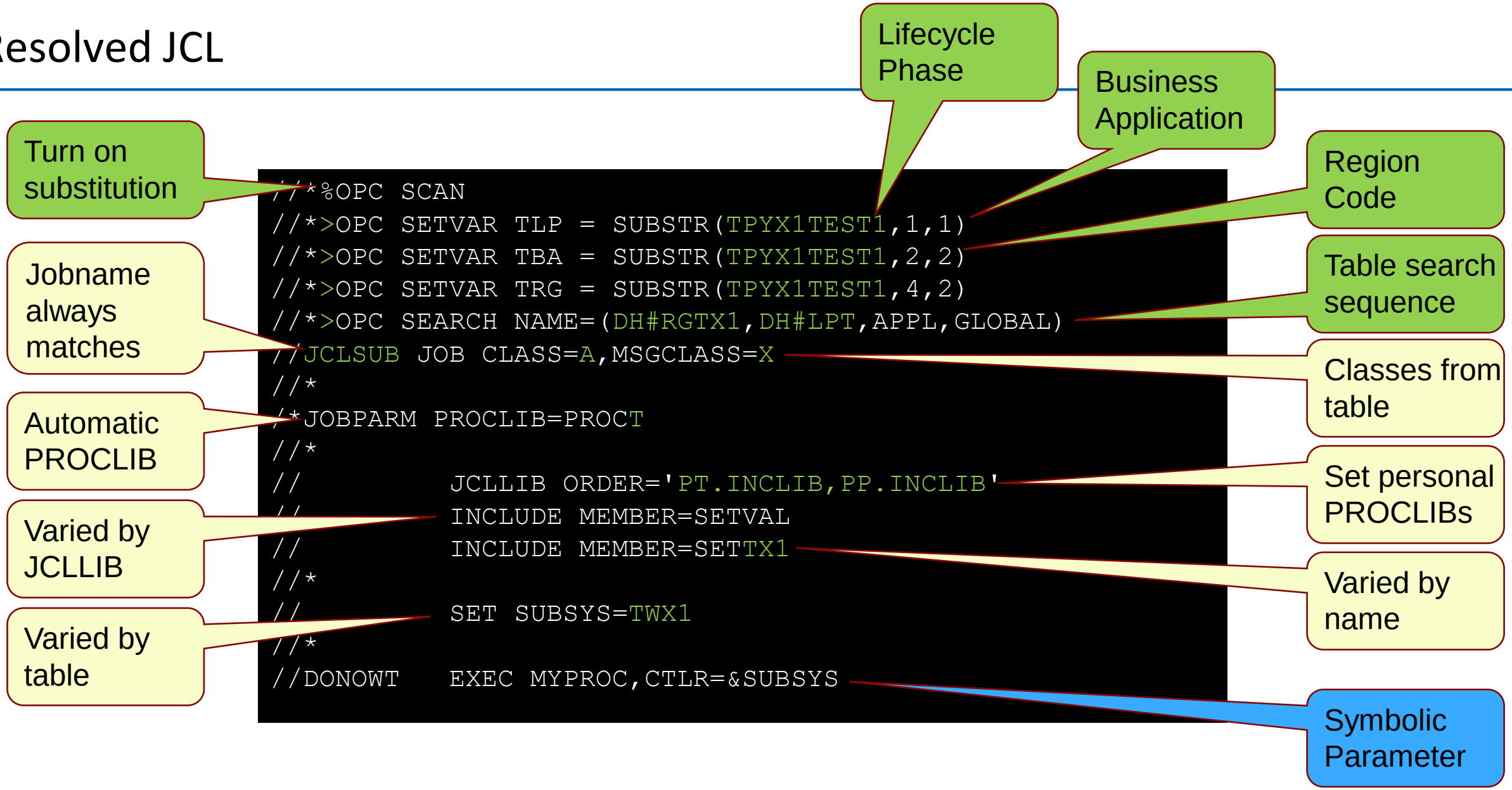
- The IBM Migration team have a sample Job-Library-Read exit
  - It can be configured by a parameter file – assembler skills not needed
  - You decide which fields in your application drive the exit
  - Available on z/Glue (Files page) https://www.ibm.com/developerworks/community/groups/community/zGlue
- For concatenations and the same member name throughout the life cycle
  - Use character 1 of jobname to select concatenation
  - Use characters 2-8 of jobname to select member name
  - No extra fields need to be set
  - Happens automatically for all jobs
- For explicitly selected concatenations and alternate member names
  - Use Operation User Fields
  - Only operations with the relevant user fields will “go elsewhere” for the JCL



Neutral JCL



Resolved JCL



Saddle up the horses

Getting things running in the Wild West

Adding workload to the plan

- As test runs aren’t planned, you need some easy ways to get things running
- WAPL can be used to add applications to the plan in a simple way

```
ADD ADID(ADHOC) HOLD(START)
```

- Or how about a group

```
ADD GROUPDEF(CROSSGROUP)
```

- Or how about a group and an application combined

```
ADD GROUPDEF(CROSSGROUP)
ADD ADID(CROSSGROUPEXTRA)
```

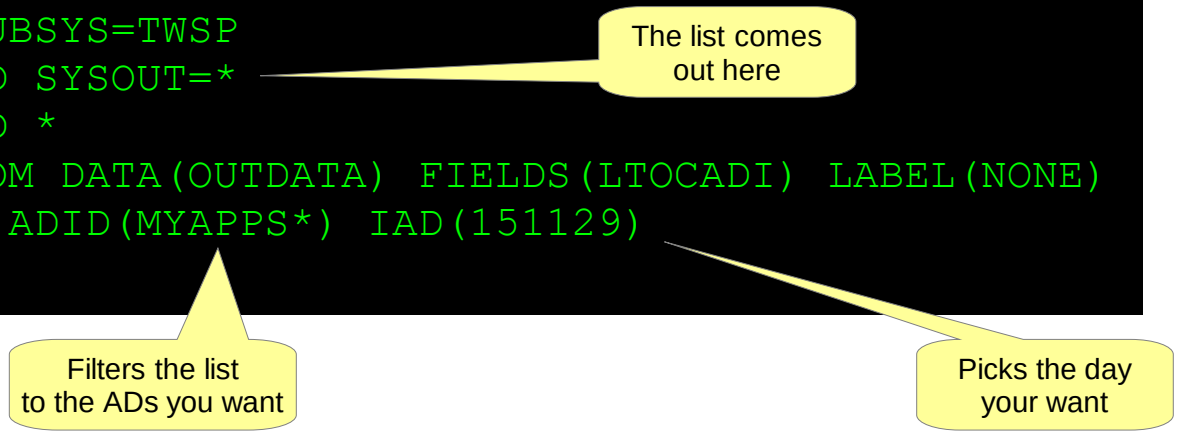
- All inter-dependencies will be honoured just as if they're planned
- For easy testing developers can keep members to assemble each test run

So what if you don’t know what you need?

- Again good naming conventions are your friend
- You can extract a list from the Production Long Term Plan
  - This will show you all of the Applications for any day for a particular prefix
  - You can then choose from that list which applications you want to add

WARNING: If you remove applications from the list, ensure that they don’t cause dependency holes

```
//RUNWAPL  EXEC EQQYXJCL,
//         SUBSYS=TWSP
//OUTDATA  DD SYSOUT=*
//SYSIN    DD *
OUTPUT LTOCCOM DATA(OUTDATA) FIELDS(LTOCADI) LABEL(NONE)
LIST LTOCCOM ADID(MYAPPS*) IAD(151129)
```



The New Sheriff in Town

Keeping things under control

The Law – Bring some security to the Wild West

- DO NOT
  - Use the same userids as production to run the batch
  - This risks accidental, or deliberate damage to production
  - Create clear access delineation between environments
  - The tracker/controller should not be given ANY direct access to business data
  - The tracker/controller should not be given SURROGAT access to anything
- Authority groups are your best form of security posse
  - You can control exactly what each user has access to
  - A user cannot change application or job to use another userid
  - Even if it is a valid userid in that controller
  - Each phase, business application and environment should have its own authority group

You can’t come in here without a tie

- Introduce the concept of control early in the life cycle
  - Allow direct user access to your lowest level (e.g. T)
  - Prevent JCL and IWSz elements being edited by users above that (e.g. Q, A, P)
- Only give your source management product update access above the first stage
  - This ensures that system and acceptance test phases have been through source management
  - Validation and enforcement can then happen early in the process
  - It also ensures consistency across the phases of the lifecycle

Like JCL checking, but for applications

- WAPL OBJECT variables allow detailed programatic access to TWS objects
- IF/THEN and DO loops allows code to navigate the full structure
- WAPL DISPLAY statements allow your code to issue your own messages
- It is now possible to write your own “checker” for IWSz objects
  - This can be included in your source management processing
  - So objects failing your site standards can be prevented from progressing

```
INFO:    Application: DAILYPLANNING
WARNING: D not valid lifecycle phase
WARNING: L not valid frequency indicator
INFO:    Operation: 001
INFO:    Operation: 005
INFO:    Resource: MY.RESOURCE
INFO:    User field: HELLO
INFO:    Operation: 010
INFO:    Operation: 255
```

```
VARSUB SCAN
VARSET MAXCC   = 0
VARSET ADID    = "DAILYPLANNING"
VARSET ADSTAT  = "A"
VARSET VALID   = "="
VARSET PHASES  = "T Q A P"
VARSET PROJECTS = "PY GL NH DB WS"
VARSET FREQS   = "D W M A"

SELECT AD ADID(!ADID) STATUS(!ADSTAT) VALID(!VALID) OBJECT(AD)

/*-------------------------------------------------------------------+
 | Start validation                                                  |
 +-------------------------------------------------------------------*/
DISPLAY "INFO:    Application: !@AD-ADID"

/*-------------------------------------------------------------------+
 | Validate the name to site standards                               |
 +-------------------------------------------------------------------*/
VARSET PHASE   = SUBSTR(!@AD-ADID,1,1)
VARSET PROJECT = SUBSTR(!@AD-ADID,2,2)
VARSET FREQ    = SUBSTR(!@AD-ADID,4,1)

IF WORDPOS("!PHASE","!PHASES") = 0 THEN DO
   DISPLAY "WARNING: !PHASE not valid lifecycle phase"
   VARSET MAXCC = MAX(!MAXCC,4)
END

IF WORDPOS("!PROJECT","!PROJECT") = 0 THEN DO
   DISPLAY "WARNING: !PROJECT not valid project code"
   VARSET MAXCC = MAX(!MAXCC,4)
END

IF WORDPOS("!FREQ","!FREQS") = 0 THEN DO
   DISPLAY "WARNING: !FREQ not valid frequency indicator"
   VARSET MAXCC = MAX(!MAXCC,4)
END

DO X = 1 TO !@AD-#ADRUN
   /* Run cycle validation goes here */
END

DO X = 1 TO !@AD-#ADOP
   /* Operation validation goes here */
   VARSET OP = "@AD-ADOP-!X."
   DISPLAY "INFO:    Operation:" @V(!OP.-ADOPNO)

   /* Single subsegment validation like ADEXT and ADSAI can go here */
   VARSET EX        = "!OP.-ADEXT-1"
   VARSET SA        = "!OP.-ADSAI-1"
   VARSET RE        = "!OP.-ADRE-1"

   IF STRIP(@V(!OP.-#ADDEP)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADDEP)
         VARSET DP        = "!OP.-ADDEP-!Y."
         /* Dependency validation goes here */
      END
   END

   IF STRIP(@V(!OP.-#ADXIV)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADXIV)
         VARSET XI        = "!OP.-ADXIV-!Y."
         /* Dependency interval validation goes here */
      END
   END

   IF STRIP(@V(!OP.-#ADCNC)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADCNC)
         VARSET CC        = "!OP.-ADCNC-!Y."
         /* Condition validation goes here */
      END
   END

   IF STRIP(@V(!OP.-#ADCIV)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADCIV)
         VARSET CI        = "!OP.-ADCIV-!Y."
         /* Condition interval validation goes here */
      END
   END

   IF STRIP(@V(!OP.-#ADCNS)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADCNS)
         VARSET CS        = "!OP.-ADCNS-!Y."
         /* Conditional dependency validation goes here */
      END
   END

   IF STRIP(@V(!OP.-#ADSR)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADSR)
         /* Special resource validation goes here */
         VARSET SR        = "!OP.-ADSR-!Y."
         VARSET ADSRN     = @V(!SR-ADSRN)     /* Name */
         VARSET ADSRT     = @V(!SR-ADSRT)     /* Type S/X */
         VARSET ADSRONER  = @V(!SR-ADSRONER)  /* On Error */
         VARSET ADSRAMNT  = @V(!SR-ADSRAMNT)  /* Quantity */
         VARSET ADSRAVACO = @V(!SR-ADSRAVACO) /* On Complete */
         DISPLAY "INFO:    Resource:" !ADSRN
      END
   END

   IF STRIP(@V(!OP.-#ADUSF)) <> "" THEN DO
      DO Y = 1 TO @V(!OP.-#ADUSF)
         /* User field validation goes here */
         VARSET UF        = "!OP.-ADUSF-!Y."
         VARSET ADUSFNAME  = @V(!UF-ADUSFNAME)  /* Name */
         VARSET ADUSFVALUE = @V(!UF-ADUSFVALUE) /* Value */
         DISPLAY "INFO:    User field:" !ADUSFNAME
      END
   END
END
```

Dead or Alive you're coming with me

- Of course with no operators, you've got nobody taking away the dead bodies
  - Time to bring in your own Robocop
  - Imagine a scenario of "Nothing is allowed in the plan past 10 days old"

- They can be found like this -

```
VARSUB SCAN
VARDATE LIMIT BASE(-10)
LIST CPOPCOM IA-LE(!LIMIT.2359) SAVELIST(SINBIN)
```

- This gives you a SINBIN containing all the offending jobs
  - You can then perform a list of actions against them
  - The sequence of actions will depend on the way you design your workload
  - You could then make those operations "harmless"
    ALTER * USELIST(SINBIN) NEW_WSNAME(NONR) DROPSR(*) TIMEDEP(N)
  - You'll then need to find and complete any operations in error

"Scattergun automation" may fail to make some updates, but iterative repeats will eventually clean up the city.

There’s more where that came from

- Most things you can do through ISPF can be automated by WAPL
  - WAPL can create dynamic applications entirely on the fly
  - It can HOLD jobs, RELEASE jobs
  - Add or remove dependencies, special resources or user fields
- Combine your WAPL automation with front ends
  - Enable all sorts of actions
  - ISPF front ends
  - Bespoke web front ends
  - You could even use Self Service Catalog to add WAPL processes with variables
- If there’s something you want to do – talk to us
  - We can provide technology and skills to help your business be a success

HCL

Relationship™

BEYOND THE CONTRACT

$7.6 BILLION ENTERPRISE | **119,000** IDEAPRENEURS | **32** COUNTRIES

Session NH: Taming the Wild West