z/OS 3.1 IBM Education Assistant

Solution Name: JES2 Policy Enhancements

Solution Element(s): JES2

July 2023

Agenda

Trademarks

- See URL http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Objectives

- In this presentation, we will describe 3.1 enhancements to the JES2 policy function

Overview

- Who (Audience)
  - Z/OS system administrators new to JES2
  - JES3 administrators overseeing JES3 to JES2 conversion
  - JES2 administrators who wish to phase out JES2 exit programs
- What (Solution)
  - Enhance capabilities to customize JES2 processing in a way that does not require:
    - Low-level programming of JES2 exit programs (assembler)
    - Knowledge of the details of JES2 internal control structures
    - Understanding of the JES2 internal processing logic
- Wow (Benefit / Value, Need Addressed)
  - Reduce the need for specific JES2 skills
  - Improve JES2 reliability by isolating JES2 from bugs in JES2 exit programs

Usage & Invocation

- JES2 policy function was introduced in V2R4.
- At a very high level, JES2 policy defines in user-level terms what JES2 must do in certain strategic points in JES2 processing.
- Externally, a JES2 policy definition is a JSON object residing in a human-readable editable z/OS data set.
- After a system administrator creates a JES2 policy definition, it is imported into JES2 by a JES2 operator command, and the policy becomes available for JES2 processing.
- Note that JES2 policies do not completely replace JES2 exits – the exits are still supported.

Usage & Invocation – policy variables

- Variables are a new part of JES2 policy infrastructure that can be used in policies of any type
  - Variable support requires policy version 2, so the "policyVersion" property must be set to 2, for example:

```
{   "policyName":      "PCONV1",
    "policyVersion":   2,
    "policyType":      " PreConversion ",
        . . .
```

Usage & Invocation – declaring variables

- Variables must be declared within the new "variables" property of the policy definition source, for example:

```
"variables":
       [ { "name"   : "$MySys          ",
           "type"   : "character list ",
           "scope"  : "local          "
         },
         { "name"   : " MySysIns      ",
           "type"   : "character      ",
           "scope"  : "instance       "
         }
       ],
```

- “variables” property is optional; it is not required if policy definition does not use variables

Usage & Invocation – declaring variables (2)

- Variables have:
  - "name" - any 16-character string starting with alpha character and followed by alphanumeric characters
  - "type" - "character" (or "char"), "numeric" (or "num") or "logical" (or "lgl")
    - Numeric and character variable can also be a list which is defined by adding "list" to the “type” specification, for example

```
            “type” : “ character list “
```

  - "scope" - "local" or "instance".
    - Variables with **local** scope are only visible to the policy definition where they are declared
    - Variables with **instance** scope are visible to all policies of all types that are applied to the same JES2 object in the same processing phase
      - E.g., policies of all types that are applied during input phase to the same job will see the same set of instance variables, so that policies of types JobCreate and JobInput can share information through the instance variables
      - Variable with **instance** scope must be declared in each policy that uses it and the declared type of the variable must be the same – otherwise an error will be detected at runtime and the involved policies will fail

Usage & Invocation – assigning variable values

- Value of a variable is changed by "assign" action, for example:

```
{ "action"  : "assign",
  "variable": "$MySys",
  "value"   : " SYSAFF "
},
{ "action"  : "assign",
  "variable": "MySysEnv",
  "value"   : " string(Sysaff) "
},
```

- "assign" action can be coded anywhere where any other action is allowed
- “value” property contains an expression that must result in a value of the type compatible with the variable definition

Usage & Invocation – using variables

- Variable name can be used anywhere in the policy definition where attribute name is allowed by syntax
- Variable name must be preceded by "$" character, unless variable use is unambiguous.
    - For example, "$" character in front of the variable name is optional in variable declaration or in the "variable" property of "assign" action.
- If variable is used before value is assigned to it, result is a default value of appropriate type:
    - 0 for numeric variables
    - FALSE for logical variables
    - Empty string (string with length 0) for character variables
    - Empty list of a proper data type for variables which are declared as lists

Usage & Invocation – InsVarHasValue()

- The order in which policies of the same type are applied is not guaranteed by the policy implementation
- In order to facilitate management of instance variables in the multi-policy environment, logical function InsVarHasValue() is provided

`InsVarHasValue(' varname ')` - TRUE/FALSE

- Function returns TRUE if instance variable with name “varname” exists and has been assigned a value by “assign” action
    - Variable name passed to the function does not have to be declared

Usage & Invocation – new policy types

- Two policy types are added in JES2 3.1
  - JobInput – policies of this type are applied at the end of the input phase (right before exits 20/50 are called). This policy type enables management of how jobs should proceed through the conversion phase and beyond.
  - JobCreate – policies of this type are applied very early in a job's life – after a job card is scanned and job class defaults are applied, and before resources are allocated. This policies allows for the management of basic job attributes, especially those that may be difficult to modify at later points in processing.

Usage & Invocation – new built-in functions

- FailJob() takes one argument, "option", that determines how the job on which the policy is being applied should fail. Supported values are:
  - 'Purge' – the job will be purged immediately after the job input phase
  - 'Conversion' – the job will be allowed to proceed to the job conversion phase, but will fail immediately after the conversion phase
  - 'ValidationFailed' – the job will fail with a security error
  - Can be used in JobInput and JobCreate type policies
- JESSymbol(*symbol-name*) allows the extraction of JES Symbol values
  - Takes a character string argument and returns the value for the matching JES Symbol
    - If there is no matching JES Symbol, then an empty string is returned
    - *Symbol-name* must be 16 characters or less
    - Can be used by JobInput policy type
- SysSymbolSub(string) allows the utilization of *System Symbol Substitution Service* (ASASYMBF)
  - Takes an input string and returns a new string with system symbols substituted for their values
    - Input string can have a maximum length of 1k, and the substituted output string can have a maximum length of 2k.
      - If any maximum is exceeded, the original string is returned
  - Can be used by all policy types

Usage & Invocation – new job attributes

- New job attributes have been added:
  - DeviceType
  - EstByteNum
  - EstByteOpt
  - EstLineNum
  - EstLineOpt
  - EstPageNum
  - EstPageOpt
  - EstPunchNum
  - EstPunchOpt
  - EstTimeNum
  - ExecNode
  - InputType
  - IsSYSLOGJob
  - JobInputFailed
  - NextPhase
  - ProgrammerName
  - ResGroup
  - SubmitterJobID
  - SubmitterJobName
  - ValidationFailed

- These attributes can be referenced by any policy type that has access to necessary job control blocks. The list of attributes available for each policy type is documented in the JES2 Installation Exits manual.

Usage & Invocation – new job attributes (2)

Attribute Name	Description	Data Type
DeviceType	Type of input device of the job: • 'NET.JR': NJE Job Receiver • 'NET.RR': Job Route Receiver • 'OFF.JR': Offload Job Receiver • 'INTRDR': Internal Reader • 'RDR': Reader • 'RMT.JR': Remote Job Receiver	Character
EstByteNum	The number (1-999999), in thousands of bytes of spool space, used by a job's output, after which the $HASP375 message informs the operator that the job's estimated output has been exceeded.	Numeric
EstByteOpt	The action JES2 takes when a job exceeds the estimated spool utilization value: • 'Warning': Job is allowed to continue execution. • 'Cancel': Job is cancelled without a dump. • 'Dump': Job is cancelled with a dump.	Character

Usage & Invocation – new job attributes (3)

Attribute Name	Description	Data Type
EstLineNum	The default print line count (1-999999), in thousands of lines, after which the $HASP375 message informs the operator that the job's estimated output has been exceeded.	Numeric
EstLineOpt	The action JES2 takes when a job exceeds the estimated number of output lines: • 'Warning': Job is allowed to continue execution. • 'Cancel': Job is cancelled without a dump. • 'Dump': Job is cancelled with a dump.	Character
EstPageNum	The number of pages (1-99999999) at which the $HASP375 message informs the operator that the job's estimated output has been exceeded.	Numeric
EstPageOpt	The action JES2 takes when a job exceeds the estimated page output: • 'Warning': Job is allowed to continue execution. • 'Cancel': Job is cancelled without a dump. • 'Dump': Job is cancelled with a dump.	Character
EstPunchNum	The default punched output (0-99999999), in cards, for a job after which the $HASP375 message informs the operator that the job's estimated output has been exceeded.	Numeric

Usage & Invocation – new job attributes (4)

Attribute Name	Description	Data Type
EstPunchOpt	The action JES2 takes when a job exceeds the estimated card output: • 'Warning': Job is allowed to continue execution. • 'Cancel': Job is cancelled without a dump. • 'Dump': Job is cancelled with a dump.	Character
EstTimeNum	The default execution time (1-9999), in minutes, after which the $HASP308 message informs the operator that the job has exceeded its estimated execution time.	Numeric
ExecNode	The node on which execution is to take place for this job	Character
InputType	The type of input for this job: • 'RDR': Reader • 'INTRDR': Internal Reader • '$SUBMIT': $SUBMIT	Character
IsSYSLOGJob	Whether the policy is being applied to a SYSLOG job	Logical
JobInputFailed	Whether the job has failed in the input phase	Logical

Usage & Invocation – new job attributes (5)

Attribute Name	Description	Data Type
NextPhase	The next phase this job is to enter: • 'XMIT': Transmission • 'CONV': Conversion • 'OUTPUT': Output • 'PURGE': Purge	Character
ProgrammerName	The programmer name for the job	Character
ResGroup	Resource group to which the job belongs	Character
SubmitterJobID	The job ID of the submitter of the job	Character
SubmitterJobName	The name of the job that submitted this job	Character
ValidationFailed	Whether job failed security validation	Logical

Usage & Invocation – JobInput Policies

- Policies of type JobInput are applied at the end of the input phase for a job.
  - They run immediately before exits 20/50 are invoked
- The main purpose of this policy type is to evaluate how jobs are submitted and manipulate how they are to proceed through the conversion phase and beyond.
- The syntax for JobInput policies is similar to that of existing policy types.
- JobInput policies support existing policy actions HoldJob and ModifyJob, in addition to FailJob.

Usage & Invocation – JobCreate Policies

- Policies of type JobCreate are applied during the input phase, right after the job card is scanned and defaults from the job class are applied
  - Since both are in the same phase, JobInput & JobCreate policies share the same instance variable scope
  - Variables can be assigned in JobCreate policies and used by JobInput policies for the same job
- The main purpose of this policy type is to provide access to basic job attributes, especially those that may be difficult to change later, such as the job name.
- As of now, this is the only policy type that can modify the new RESGROUP attribute

Usage & Invocation – JobInput & JobCreate Policies

- The following table lists policy attributes (both existing and new) and how they may be used by the new policy types

Attribute Name	**JobInput can read**	**JobInput can modify**	**JobCreate can read**	**JobCreate can modify**
CompletionCode	Yes	No	Yes	No
CreatedLocally	Yes	No	Yes	No
DeviceType	Yes	No	Yes	No
EstByteNum	Yes	Yes	Yes	Yes
EstByteOpt	Yes	Yes	Yes	Yes
EstLineNum	Yes	Yes	Yes	Yes
EstLineOpt	Yes	Yes	Yes	Yes
EstPageNum	Yes	Yes	Yes	Yes
EstPunchNum	Yes	Yes	Yes	Yes
EstPunchOpt	Yes	Yes	Yes	Yes

Usage & Invocation – JobInput & JobCreate Policies (2)

Attribute Name	JobInput can read	JobInput can modify	JobCreate can read	JobCreate can modify
EstTimeNum	Yes	Yes	Yes	Yes
ExecNode	Yes	Yes	No	No
InputMember	Yes	No	Yes	No
InputType	Yes	No	Yes	No
JobAcct	Yes	No	Yes	No
JobClass	Yes	Yes	Yes	Yes
JobInputFailed	Yes	No	Yes	No
JobHasAffinity	Yes	No	Yes	No
JobIsDupl_Exempt	Yes	Yes	Yes	Yes
JobHasFailed	Yes	No	Yes	No
JobIsHeld	Yes	No	Yes	No
JobIsPrivileged	Yes	No	Yes	No
JobIsProtected	Yes	No	Yes	No

Usage & Invocation – JobInput & JobCreate Policies (4)

Attribute Name	JobInput can read	JobInput can modify	JobCreate can read	JobCreate can modify
JobIsSYSLOG	Yes	No	Yes	No
JobName	Yes	No	Yes	Yes
JobOwner	Yes	No	Yes	No
JobPrty	Yes	Yes	Yes	Yes
JobSecLabel	Yes	No	Yes	No
JobSubmitter	Yes	No	Yes	No
JobType	Yes	No	Yes	No
JOEResAction	Yes	Yes	Yes	Yes
JOEResLimit	Yes	Yes	Yes	Yes
MsgClass	Yes	No	Yes	Yes
NextPhase	Yes	Yes	Yes	No
ProgrammerName	Yes	No	Yes	No
ResGroup	Yes	No	Yes	Yes
SchEnv	Yes	Yes	Yes	Yes
SrvClass	Yes	Yes	Yes	Yes

Usage & Invocation – JobInput Policy Example

```
{ "policyName": " JINPUT1 ",
  "policyVersion": 1,
  "policyType": " JobInput ",
  "definitions":
   [
    { "condition" : "  EstByteNum > 9999 ",
      "actions" :
        [
           {   "action" : " modifyJob ",
               "attribute" : " EstByteOpt ",
               "value" : "'CANCEL'"
           },
           {   "action" : " SendMessage ",
               "message" : " 'Job ' || JOBNAME || ' will be canceled' "
           }
         ]
    }
  ]
}
```

Usage & Invocation – JobCreate Policy Example

```
{  "policyName": " JCREATE ",
   "policyVersion": 1,
   "policyType": " JobCreate ",
   "definitions":
     [
      {  "condition" : "  PROGRAMMERNAME = 'FRED' ",
         "actions" :
           [
              {   "action" : " modifyJob ",
                  "attribute" : " RESGROUP ",
                  "value" : "'FREDGRP'"
              }
            ]
      }
   ]
}
```

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - Any installation that has a need to customize JES2 processing.

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level:

  No

- JobInput and JobCreate policy types are not visible to down-level members; they can still be used by 3.1 members, but down-level members will ignore them
- Attempting to use a policy type, attribute, or action unsupported by a given member's version will result in the policy being rejected

Installation & Configuration

- No special installation is required.
- Planning considerations for using JES2 policies are documented in JES2 Installation Exits publication.

Summary

- In this presentation we described 3.1 enhancements to the JES2 policy function.

Appendix

- Publications
    - z/OS 3.1 JES2 Commands
    - z/OS 3.1 JES2 Initialization and Tuning Guide
    - z/OS 3.1 JES2 Initialization and Tuning Reference
    - z/OS 3.1 JES2 Installation Exits
    - z/OS 3.1 JES2 Messages
    - z/OS 3.1 MVS Using the Subsystem Interface