CheatSheet #34 zTidBits z10 Java Batch

- Did you know you can run Java as a **batch runtime** eligible for the zAAP CPU?
- This is an excellent way running a singleton JVM without using a container.
- There are many reasons for using Java in a batch or stand-alone mode on z/OS.
- The Java language can be used like any other programming language in a batch job, shell script, or from the command line, but the Java language is in most cases more powerful than any other language.
- With Java Batch (JB) you can:
  - Access: Databases (JDBC), VSAM, HFS files, MQ
  - Invoke: CICS transactions (CTG) Web Services (SOAP, TCP/IP sockets
  - Run Started-Tasks that wait for incoming work
  - Produce reports and PDF files
  - Pass datasets created by traditional job steps to Java programs
  - Call programs in CICS or IMS using J2CA APIs
  - Perform CPU-intensive calculations
  - Integrate with other programming languages using Java Native Interface (JNI) and zOS' Language Environment (LE).
- You can also use functions in the area of security and systems management.

  NOTE: Java on z/OS provides a number of security frameworks for security-related functionality, such as encryption, authentication, and authorization, and the JVMTI interface for systems management.
- Java batch programs can be short running or long running.
  - An example of a short-running Java program is a program performing a daily reconciliation between the contents of two data sets. An example of a long-running Java program is a continuously running server program that accepts requests from outside (through MQ or TCP/IP, for example).
- **JRIO is** an integral part of the Developer Kit, Java 2 Technology Edition and provides record-oriented applications (supporting multiple file systems) used on different file systems.
  - Provides a set of zOS native code drivers to access:
    - > Virtual Sequential Access Method (VSAM) data sets (KSDS only)
    - > Non-VSAM record-oriented data sets
    - > The zOS System Catalog
    - > Partitioned data set (PDS) directory

  NOTE: JRIO provides indexed I/O access to records within a VSAM KSDS. The VSAM support uses zOS native code to provide a set of classes that implement the KeyedAccessRecordFile class.
  This lets you access records: in entry sequence order, in primary unique key, by alternate unique key or non-unique key.
  - JRIO focuses on providing sequential and random I/O access to record-oriented data within non-VSAM sequential data sets and PDS.
  - The JRIO non-VSAM support uses zOS native code to provide a set of classes and new directory classes that use underlying non-VSAM physical record files.
- JRIO provides support for sequential and random I/O access to records within HFS files.
  - The HFS support uses pure Java code to provide a set of classes and directory classes that use the underlying **java.io**.
- There are two general batch Unix System Services (USS) processors:
  1. **BPXBATCH** : BPXBATCH is a utility that you can use to run shell commands or shell scripts and to run executable files through the zOS batch environment. BPXBATCH program sets up the stdin, stdout, and stderr and execs. BPXBATCH will run as a JES batch address space from JCL which can use WLM initiators. REXX execs can also use BPXBATCH to run shell scripts and executable files.
  2. **BPXBATSL** provides users with an alternate entry point into BPXBATCH, and forces a program to run using a local spawn instead of fork/exec as BPXBATCH does. This ultimately allows a program to *run faster*.

  NOTE: BPXBATSL is also useful when the user wants to perform a local spawn of their program but also needs subsequent child processes to be fork/exec'ed. Formerly, this could not be done since BPXBATCH and the requested program can share the environment variables. BPXBATSL is an alias of BPXBATCH.

  NOTE: BPXBATA2 and BPXBATA8 are provided as APF authorized alternatives to BPXBATSL. BPXBATA2 and BPXBATA8 provide the capability for a target APF authorized z/OS UNIX program to run in the same address space as the originating job, allowing it to share the same allocations, joblog, and so on. BPXBATA2 is specifically intended to provide the capability for APF Authorized z/OS UNIX program to be started in a PSW Key 2 .
  [PSW KEYS are used by zOS as a storage protection mechanism]

Running Java batch on Z takes advantage of the mainframe's powerful I/O subsystem.

- **JZOS** is superior to running Java under BPXBATCH and is offered on IBMs alphaworks.
- The JZOS job launcher and runtime APIs are the latest inclusion in the IBM Java SDK distribution for z/OS, which addresses BPXBATCH and BPXBATSL shortfalls.
- JZOS includes Java classes that make the console communication from applications easy instead of implementing the console communication via Java Native Interface (JNI) calls.
- JZOS provides a framework to register a *listener* for interaction with operators (WTO) { Write To Operator
  - It also provides a method to write messages to MVS system logs directly from Java applications having a means to interface with the system automation tools to monitor the status of the running Java batch programs.
- JZOS allows for Condition Code (CC) passing from System.exit() to subsequent steps.
  - JZOS enables Java programs to be integrated seamlessly with other job steps within a job.
  - Execution of job steps is typically controlled by *return codes* from the previous job steps.
  - JZOS reliably delivers the exit status of Java programs to following steps in a job run, which allows for inclusion of Java program steps in other zOS utilities and programs.
- Another big advantage of JZOS is its full support for DD statements.
  - Because Java programs launched via JZOS run in the same address space as any other steps, Java programs are able to access DD statements specified in a job.
    For example, it is common to allocate a temporary data set to store output from a step and be used by the following steps in a job. Using JZOS, Java programs are able to read and write from such data sets, which makes Java programs more appealing when considering replacing legacy applications.
- Thirdly, the JZOS batch launcher directs stdout and stderr input streams to the standard zOS data sets and JES' SYSOUT data set.
  - JZOS Java programs are able to read from stdin from the standard zOS data sets.

  NOTE: With BPXBATCH, it is only possible to write/read to files in USS.
- Operators and programmers are able to monitor the output from Java programs via System Display and Search Facility (SDSF), just like monitoring other job steps in zOS environments.

	BPXBATCH	BPXBATSL	JZOS
Flexible configuration of environment variables	YES*	YES*	YES
Route output directly to SYSOUT datasets	NO	NO	YES
Control output encoding separately from the default	YES**	YES**	YES
Condition-code passing between Java & non-Java steps	NO	NO	YES
Use zOS datasets and DD statements	NO	YES	YES
JVM runs in same address space	NO	YES	YES
Communicates with zOS Console	NO	YES***	YES

* *Variable Scripting and Substitution not allowed*
** *Must use ICONV in separate step*
*** *Must use C Console Function from a JNI Routine*

SAMPLES

```
//BPXBAT01 JOB (999,XXX),'JAVA BATCH',CLASS=A,MSGLEVEL=(1,1) .....
//JAVAJVM EXEC PGM=BPXBATCH,REGION=0M
//STDIN    DD PATH='/u/turbo/bin/myscript.in',PATHOPTS=(ORDONLY)
//STDOUT DD PATH='/u/turbo/bin/mystd.out',PATHOPTS=(OWRONLY,OCREAT),
//      PATHMODE=SIRWXU
//STDERR DD PATH='/u/turbo/bin/mystd.err',PATHOPTS=(OWRONLY,OCREAT),
//      PATHMODE=SIRWXU

//JZOSBAT JOB (999,XXX),'JAVA JZOS',CLASS=A,MSGLEVEL=(1,1),
//   MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//JAVAJVM EXEC PGM=JZOSVM14,PARM='MyClass arg1'
//STEPLIB DD DSN=JZOS.LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD *
   CLASSPATH=/proj/foo:/mainprog/prog1.jar
//
```