JZOS Batch Launcher and Toolkit function in
IBM SDK for z/OS, Java Technology Edition,
Version 8

Installation and User's Guide

SC27-8418-07

Note:
Before you use this information and the product it supports, read the information in “Notices” on page 41.

This edition applies to IBM SDK, Java Technology Edition, Version 8 and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC27-8418-06.

Contents

Tables

About This Document

This document includes the instructions to install the JZOS batch launcher capabilities for both SMP/E and non-SMP/E installers of the z/OS® Java™ products.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to mhvrcfs@us.ibm.com
2. Visit the Contact z/OS web page at http://www.ibm.com/systems/z/os/zos/webqs.html

Note: The publication for this product is in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties using the PDF file and want to request a Web-based format for a publication, use one of the methods described or mail a request to the provided address.

Include the following information:

- Your name
- Your email address
- The publication title and order number:
  JZOS Batch Launcher and Toolkit function in IBM® SDK for z/OS, Java Technology Edition, Version 8 Installation and User's Guide
  SC27-8418-07
- The topic and page number related to your comment
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM zSeries support web page at the z/OS support page (http://www.ibm.com/systems/z/support/).

Summary of Changes

This document contains terminology, maintenance, and editorial changes. Technical changes are indicated by a vertical line to the left of the change.

SC27-8418-07, Java Technology Edition for z/OS, Version 8 SR7 (November 2021)

JZOS has been updated to add QueryVirtualServer class

SC27-8418-06, Java Technology Edition for z/OS, Version 8 SR6 FP35 (August 2021)

JZOS has been updated to add read and locate unlocked support to ZFile class

SC27-8418-05, Java Technology Edition for z/OS, Version 8 SR5 FP25 (October 2018)

JZOS has been updated to add Try-with-Resources to Enqueue class

SC27-8418-04, Java Technology Edition for z/OS, Version 8 SR5 FP20 (July 2018)

JZOS has been updated with the following new functions:

- Added ISGENQ RNL= option to JZOS Enqueue class
- Added java.time.* nanosecond support in ZUtil class
- Supported zOS batch Cobol/Java mix: redirect java stdout/stderr into COBOL SYSOUT
- Provided consistent behavior between CLASSPATH wildcarding in Java command and JZOS launcher
- Supported 8-character TSO User ID

SC27-8418-03, Java Technology Edition for z/OS, Version 8 SR5 (September 2017)

The JZOS Batch Launcher has been updated to increase CLASSPATH length to 150,528 bytes.

SC27-8418-02, Java Technology Edition for z/OS, Version 8 SR2 FP10 (January 2016)

The JZOS Batch Launcher has been updated with enhancements relating to setting the desired working directory. For more information, see “Setting the Desired Batch Launcher Working Directory” on page 12. In addition,“Java Runtime Statistics” on page 17 has been updated with a reference to a sample assembler DSECT.

SC27-8418-01, Java Technology Edition for z/OS, Version 8 SR2 (October 2015)

The JZOS Batch Launcher has been updated with a new environment variable, HJV_JZOS_JVM_SMF_THREADS_NATIVE_ID. For information on this environment variable, see Table 4 on page 10.

SC27-8418-00, Java Technology Edition for z/OS, Version 8

The JZOS Batch Launcher has been updated with additional enhancements relating to logging of MVS System Management Facilities (SMF) records containing Java runtime statistics. See “Java Runtime Statistics” on page 17, for more information.

Chapter 1. Overview

The z/OS Java products have been extended to include the JZOS Batch Launcher and Toolkit. JZOS is a set of tools that enhances the ability for z/OS Java applications to run in a traditional batch environment and/or access z/OS system services. The enhancements include a native launcher for running Java applications directly as batch jobs or started tasks, and a set of Java classes that make access to traditional z/OS data and key system services directly available from Java applications. Additional system services include console communication, multiline WTO (write to operator), and return code passing capability.

JZOS Batch Launcher and Toolkit Capabilities and Features

- Run Java applications on z/OS seamlessly in an MVS™ batch job step or started task.
- Supports IBM SDK for z/OS, Java Technology Edition, Version 8, (31 bit and 64 bit).
- Simple yet flexible way to configure the Java execution environment.
- Access to datasets via JCL DD statements.
- Send output directly to JES SYSOUT datasets with automatic codepage transcoding.
- Pass condition codes between Java and non-Java job steps.
- Communicate with the MVS system console.
- Read and write traditional MVS datasets from Java.
- Java interfaces to many z/OS specific APIs and features including SMF, Catalog Search and Logstreams.
- Java classes to convert COBOL and Assembler data type fields to Java objects.
- Invoke DFSORT and direct either input and output to the Java application.
- Invoke z/OS Access Method Services (IDCAMS).
- Serialize z/OS resources (ISGENQ).
- Access z/OS system symbols (ASASYMB system symbol service).
- Access z/OS Workload Manager (WLM) services.
- Submit z/OS batch jobs from Java.

What is in this Document?

The JZOS functionality now included in IBM SDK for z/OS, Java Technology Edition (and other z/OS Java products) consists of batch launcher capabilities, system services, and file I/O capabilities.

This document includes the instructions to install the JZOS batch launcher capabilities for both SMP/E and non-SMP/E installers of the z/OS Java products.

It also includes the Batch Launcher User's Guide and the Toolkit User's Guide. The z/OS Java web site contains JZOS javadoc and JZOS sample programs.

For more details about the z/OS Java products and the JZOS function, see www.ibm.com/systems/z/os/zos/tools/java/ and www.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html.

Chapter 2. Installation

This chapter contains install instructions.

Introduction to JZOS Batch Launcher Installation

The function consists of three pieces: a load module that must be put into a z/OS PDSE, a sample start proc that can be tailored and put into an appropriate PROCLIB, and sample JCL that can be tailored and put into an appropriate SAMPLIB. The names of the three files delivered with the product are as follows:

Product	Load module name	Sample PROC	Sample JCL
IBM 31-bit SDK for z/OS, Java Technology Edition, Version 8 (5655-DGG)	JVMLDM80	JVMPRC80	JVMJCL80
IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8 (5655-DGH)	JVMLDM86	JVMPRC86	JVMJCL86

Non-SMP/E Users

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions. The instructions are written here for the 31-bit SDK Version 8.

   Note: If the batch launcher function is not installed, steps 2 —6 are not followed and JZOS batch launcher function can not be used. However, all other JVM functions can still be used, including the JZOS system services and file I/O.
2. Allocate any needed MVS PDSE or PDS data sets. For your information, the SMP/E install will, by default, place a load module in SYS1.SIEALNKE, a sample PROC in SYS1.PROCLIB and sample JCL in SYS1.SAMPLIB. (For installation into private data sets, suggested allocation sizes are F/FB,80 5 tracks for SAMPJCL and for SAMPPROC.
3. Copy the load module to a PDSE data set. Copy the PROC and JCL to a PDS data set. The load module will be found in `<JAVA_HOME>/mvstools`. The sample JCL and PROC will be found in `<JAVA_HOME>/mvstools/samples/jcl`. For example, for the 31-bit Version 8 SDK and using the default target libraries, issue the following commands under a USS shell:

   ```
   cd <JAVA_HOME>/mvstools
   cp samples/jcl/JVMJCL80 "//'SYS1.SAMPLIB(JVMJCL80)'"
   cp samples/jcl/JVMPRC80 "//'SYS1.PROCLIB(JVMPRC80)'"
   cp -X JVMLDM80 "//'SYS1.SIEALNKE(JVMLDM80)'"
   ```
4. Change the sample JCL and PROC as appropriate for your environment. Specifically, update JCL with JOB card information, update the JCL and PROC with HLQ where the PROC and LOADLIB exist, and update the PROC with the location of JAVA_HOME. Make sure your sample JCL or PROC includes a STEPLIB to the load library unless that load library is included in your LNKLST member.

5. SUBMIT the modified JCL for the 31-bit version and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:

```
JVMJZBL1001N JZOS batch Launcher Version: 2.4.6 2014-10-31
JVMJZBL1002N Copyright (C) IBM Corp. 2005, 2015
java version "1.8.0"
Java(TM) SE Runtime Environment (build pmz3180-20150129_02)
IBM J9 VM (build 2.8, JRE 1.8.0 z/OS s390-31 20150116_231420 (JIT enabled, AOT enabled)
J9VM - R28_Java8_GA_20150116_2030_B231420
JIT  - tr.r14.java_20150109_82886.02
GC   - R28_Java8_GA_20150116_2030_B231420
J9CL - 20150116_231420)
JVMJZBL1023N Invoking HelloWorld.main()...
JVMJZBL1024N HelloWorld.main() completed.
JVMJZBL1021N JZOS batch launcher completed, return code=0
Hello World
```

6. To diagnose problems with the JZOS batch launcher, change the LOGLVL parameter to '+I' :

```
// EXEC JVMLDM80,LOGLVL='+I',
```

Note: Setting this logging level (+I) will dump the environment that is passed to the JVM. The trace level setting "+T" will produce many messages, some of which may be helpful in tracking down installation problems.

SMP/E Users

The SMP/E installation of the ordered IBM SDK for z/OS, Java Technology Edition, Version 8 products covers the JZOS installation.

Chapter 3. IBM JZOS Batch Launcher User's Guide

The IBM JZOS batch launcher is a MVS batch program which configures and launches the Java virtual machine (JVM). It can be run as a MVS batch job or started task. The following stored procedure to run the JZOS batch launcher is distributed with the product.

Sample PROC

The following sample is distributed in the "mvstools/samples/jcl" directory. This sample PROC is for the 31-bit IBM SDK for z/OS, Java Technology Edition, Version 8 product.

```
//*********************************************************************
//* Licensed Materials - Property of IBM
//* 5655-DGG
//* Copyright IBM Corp. 1997, 2015
//* STATUS = HJVA800
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc for your installation:
//* If the PDSE containing the JVMLDMxx module is not in your
//* LNKLST, uncomment the STEPLIB statement and update the DSN to
//* refer to the PDSE
//*
//*********************************************************************
//JVMPRC80 PROC JAVACLS=,               < Fully Qfied Java class..RQD
//   ARGS=,                             < Args to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',      < STEPLIB FOR JVMLDM module
//   VERSION='80',                      < JVMLDM version: 80
//   LOGLVL='',                         < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                      < EXECUTION REGION SIZE
//   LEPARM=''
//JAVAJVM  EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//   PARM='&LEPARM/&LOGLVL &JAVACLS &ARGS'
//* STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD SYSOUT=*          < System stderr
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.err
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*
//*The following DDs can/should be present in the calling JCL
//*
//*STDIN   DD                   < OPTIONAL - Java System.in
//*STDENV  DD                   < REQUIRED - JVM Environment script
//*MAINARGS DD                  < OPTIONAL - Alt. method to supply args
// PEND
```

Sample JCL

The following sample is distributed in the 'mvstools/samples/jcl' directory. This sample JCL is for the 31-bit IBM SDK for z/OS, Java Technology Edition, Version 8 product.

```
//*********************************************************************
//* Licensed Materials - Property of IBM
//* 5655-DGG
//* Copyright IBM Corp. 1997, 2015
//* STATUS = HJVA800
//*
//* Batch job to run the Java VM
//*
//* Tailor the proc and job for your installation:
//* 1.) Modify the Job card per your installation's requirements
//* 2.) Modify the PROCLIB card to point to this PDS
//* 3.) edit JAVA_HOME to point the location of the SDK
//* 4.) edit APP_HOME to point the location of your app (if any)
//* 5.) Modify the CLASSPATH as required to point to your Java code
//* 6.) Modify JAVACLS and ARGS to launch desired Java class
//*
//*********************************************************************
//JAVA EXEC PROC=JVMPRC80,
// JAVACLS='HelloWorld'
//STDENV DD *
# This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.

. /etc/profile
export JAVA_HOME=/usr/lpp/java/J8.0

export PATH=/bin:"${JAVA_HOME}"/bin

LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390/j9vm
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic
export LIBPATH="$LIBPATH":

# Customize your CLASSPATH here
APP_HOME=$JAVA_HOME
CLASSPATH=$APP_HOME:"${JAVA_HOME}"/lib:"${JAVA_HOME}"/lib/ext

# Add Application required jars to end of CLASSPATH
for i in "${APP_HOME}"/*.jar; do
    CLASSPATH="$CLASSPATH":"$i"
    done
export CLASSPATH="$CLASSPATH":

# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional arguments to main
#export JZOS_MAIN_ARGS=""

# Configure JVM options
IJO="-Xms16m -Xmx128m"
# Uncomment the following to aid in debugging "Class Not Found" problems
#IJO="$IJO -verbose:class"
# Uncomment the following if you want to run with Ascii file encoding..
#IJO="$IJO -Dfile.encoding=ISO8859-1"
export IBM_JAVA_OPTIONS="$IJO "

//
```

Specifying the Java main class and its arguments

The goal of any Java launcher is to run the main() method of some Java class and possibly pass it some arguments. The Java class name and its arguments may be supplied to the Java batch launcher in the following ways:

- The fully qualified main class name and any arguments can be specified as the PARM= string to the batch launcher program. The JVMPRCxx stored procedure defines keyword parameters 'JAVACLS=' and 'ARGS=' which can be used to set the the program's PARM= string.
- The JZOS_MAIN_ARGS environment variable can contain the main class name and arguments.
- The contents of the file pointed to by //MAINARGS can contain the Java class name and arguments. This DD name can be changed from //MAINARGS to some other name by setting the environment variable JZOS_MAINARGS_DD.

These three mechanisms can be used individually or in combination to specify the class name and its arguments. If used in combination, they are read in the following order:

1. PARM=
2. the contents of the environment variable JZOS_MAIN_ARGS
3. the contents of the file pointed to by JZOS_MAIN_ARGS_DD (by default MAINARGS)

The main class name and its arguments are read from one or more of these sources as strings separated by white space characters (space, tab, newline). Single quotes may be used to enclose arguments that include white space characters. When enclosed in single quotes, an argument may include a newline character if the token spans multiple input lines, unless the line ends in a backslash character, in which case the newline character is not included in the quoted argument. When reading input from //MAINARGS, trailing spaces are automatically removed, but the input must not contain line numbers.

An executable JAR file may be launched by specifying "-jar <jar file name>" in place of a main class name. This behaves the same as the "-jar" option on the java shell command launcher - the MANIFEST entry is read from named jar file to find the main class name.

Example: Supplying arguments to a Java class

This example supplies arguments to a Java class.

```
// EXEC PROC=JVMPRCxx,JAVACLS='com.package.MyClass',
// ARGS='argument1 -arg2'
//STDENV *
...
//MAINARGS DD *
arg.number.3 'argument4 with embedded spaces
and newline' 'argument5 with embedded spaces \
but no newline'
//
```

The above example would result in the following:

- Java main class name = 'com.package.MyClass'
- arg[1] = 'argument1'
- arg[2] = '-arg2'

- arg[3] = 'arg.number.3'
- arg[4] = 'argument4 with embedded spaces and newline'
- arg[5] = 'argument5 with embedded spaces but no newline'

Setting Batch Launcher Logging Levels

The PARM= parameter to the batch launcher can accept an optional argument to control logging messages written by the launcher to //SYSOUT. If present, it must be the first argument, and can be one of the following:

Table 1. Optional arguments

Level	Description
+E	Only error level messages are emitted.
+W	Adds warning level messages.
+N	Adds notice level messages (the default).
+I	Adds informational messages. This level includes a dump of the environment variables (including CLASSPATH) prior to creating the Java VM.
+D	Adds debugging level messages. This level will print input to and output from the //STDENV configuration script process.
+T	Adds trace level messages. This level should be used when reporting a launcher problem to IBM.

The sample JCL procedure “JVMPRCxx” has a keyword parameter LOGLVL= which can be used to set this option.

Configuring Environment Variables

The //STDENV file is required by the batch launcher to contain a shell script which is used to set the environment variables used to configure the Java runtime environment. This file is used as input to the UNIX System Services shell (/bin/sh) and has the following requirements:

- It must export the environment variables that it wishes to set using the 'export' shell command.
- The input must not contain line numbers.
- The script must not issue the 'exit' shell command.
- The script is run under a regular shell, not a 'login' shell, so the /etc/profile script and user .profile script are not automatically executed. These scripts can be explicitly executed ('dotted in') if they are needed.

For an example STDENV DD, see “Sample JCL” on page 6.

In order to use private program libraries with your JZOS Batch Launcher application, you must specify either a JOBLIB DD statement or a STEPLIB DD statement. Exporting the STEPLIB environment variable using the 'export' shell command has no effect.

For example:

```
//JVMPRC80 PROC JAVACLS=, < Fully Qualified Java class..RQD
// ARGS=, < Args to Java class
// LIBRARY='<HLQ>.JZOS.LOADLIB', < STEPLIB FOR JVMLDM module
...
//JAVAJVM EXEC PGM=JVMLDM&VERSION;,REGION=&REGSIZ;,
// PARM='&LEPARM/&LOGLVL;&JAVALS; &ARGS;'
// STEPLIB DD DSN=&LIBRARY,DISP=SHR
//         DD DSN=<HLQ>.<DATASET>,DISP=SHR
...
```

In the above example, we defined a STEPLIB DD <HLQ>.JZOS.LOADLIB for locating the JZOS batch launcher and a STEPLIB DD <HLQ>.<DATASET> for locating other programs that may be executed.

Environment Variables

The following table shows the environment variables that are required or are commonly used with the Java Batch launcher. Additional environment variables may be required by your Java application or libraries that it uses.

System Environment Variables

Refer to the z/OS UNIX System Services Planning, GA22-7800 for more information.

Table 2. System Environment Variables

Environment Variable	Description
PATH	A colon-separated list of directories used search for executable files. Must include at least /bin and $JAVA_HOME/bin.
JAVA_HOME	Should point to the base Java SDK directory
LIBPATH	A colon-separated list of directories used to search for dynamic shared libraries. Must include at least /lib, /usr/lib, and the $JAVA_HOME/bin/classic
LANG	Specifies the language that system messages are displayed in. The system default is 'C'
TZ	The timezone name; must be set in order for Java to display local times.
NLSPATH	A colon-separated list of directories that the system searches for message catalogs.

Java SDK Environment Variables

Refer to the IBM Java SDK Diagnostic Guides for the specific Java SDK that you are using for more information.

Table 3. Java SDK Environment Variables

Environment Variable	Description
CLASSPATH	A colon-separated list of directories and jar names used to search for Java classes.

Table 3. Java SDK Environment Variables (continued)

Environment Variable	Description
IBM_JAVA_OPTIONS	This variable is used to set Java SDK options. These can include -X, -D or -verbose:gc style options; for example, `-Xms256m -Djava.compiler=NONE -verbose:gc` See "Java SDK Options and System Properties" on page 11 for more information. See "JZOS Environment Variables" - JZOS_JVM_OPTIONS for more information.

JZOS Environment Variables

This table shows the JZOS environment variables and a description.

Table 4. JZOS Environment Variables

Environment Variable	Description
JZOS_ENABLE_MVS_COMMANDS = {true | false}	This environment variable determines whether or not JZOS will allow processing of the MVS operator commands START (S), MODIFY (F) and STOP (P). If set to 'false', the JZOS batch launcher will not respond to MVS operator commands. The default if not specified is 'true'. See "MVS Console Interface" on page 13 for more information.
JZOS_OUTPUT_ENCODING = {codepage}	This environment variable specifies the codepage used by JZOS for its output to STDOUT and STDERR. If not specified, the default codepage for the current locale is used. If LANG=C is set, this is then this default is normally 'IBM-1047', which is an EBCDIC codepage. See "Controlling Output Encoding" on page 14 for more information.
JZOS_ENABLE_OUTPUT_TRANSCODING = {true | false}	If set to false, raw bytes written to System.out and System.err are not transcoded to the JZOS_OUTPUT_ENCODING codepage. See "Controlling Output Encoding" on page 14 for more information.
JZOS_GENERATE_SYSTEM_EXIT = {true | false }	If set to true, JZOS will generate a System.exit() call upon completion of main(). This will cause JZOS to complete, even if there are active non-daemon threads. The default if not specified is 'false', which means JZOS will wait for non-daemon threads to complete before exiting.
JZOS_MAIN_ARGS = {classname and arguments} JZOS_MAIN_ARGS_DD = {ddname | MAINARGS }	Allows for additional arguments to specified to the main method that JZOS invokes. See "Specifying the Java main class and its arguments" on page 7 for more information.
JZOS_JVM_OPTIONS	Some Java SDK options and System properties are not recognized by the JVM when specified in the "IBM_JAVA_OPTIONS" environment variable. JZOS_JVM_OPTIONS can be used to specify these options. The default property settings must be specified along with the desired extensions when using JZOS_JVM_OPTIONS.
JZOS_ABEND_EXIT = n Where n is a positive integer between 0 and 99, inclusive	If this environment variable is set and the JZOS Batch Launcher has a non-zero exit/condition code greater than the value specified in JZOS_ABEND_EXIT, then the JZOS Batch Launcher will force an abnormal termination with a user ABEND. See"Messages, Return Codes, and Abnormal Termination" on page 15, for more information.

Table 4. JZOS Environment Variables (continued)

Environment Variable	Description
HJV_JZOS_JVM_SMF_LOGGING = { true | false }	If set to 'true', the JZOS batch launcher will register a Java Virtual Machine (JVM) shutdown hook to log a SMF record containing Java runtime performance statistics right before the JVM shuts down. The use of this environment variable is preferred over JZOS_JVM_SMF_LOGGING, which is deprecated and will be eventually removed in the next version of Java. For more information on the format of the SMF record, see "Java Runtime Statistics" on page 17. The default if not specified is 'false'.
HJV_JZOS_JVM_SMF_LOGGING_INTERVAL = n Where n is a positive integer between 1 and Long.MAX_VALUE, inclusive	This environment variable only takes effect if HJV_JZOS_JVM_SMF_LOGGING is set to 'true'. This environment variable enables the JZOS batch launcher to schedule a periodic task where a single SMF record containing Java runtime performance statistics will be logged every n seconds. If n is outside the supported range of values or if this environment variable is not set, periodic logging of SMF records will be disabled. For more information on the format of the SMF record, see "Java Runtime Statistics" on page 17.
HJV_JZOS_JVM_SMF_THREADS = { true | false }	This environment variable only takes effect if HJV_JZOS_JVM_SMF_LOGGING is set to 'true'. This environment variable controls whether individual per thread detail will be included in the SMF record. If this environment variable is not set, is set to 'false', or is set to any value other than 'true', individual per thread detail will be omitted from the SMF record. For more information on the format of the SMF record, see "Java Runtime Statistics" on page 17.
HJV_JZOS_JVM_SMF_THREADS_NATIVE_ID = { true | false }	This environment variable only takes effect if both HJV_JZOS_JVM_SMF_LOGGING and HJV_JZOS_JVM_SMF_THREADS are set to 'true'. If this environment variable is set to 'true', then the native OS thread ID field in the individual per thread detail section of the SMF record will be assigned the native OS thread ID of the corresponding Java thread. If this environment variable is not set, set to 'false', or any value other than 'true', then a value of -1 will be assigned. For more information on the format of the SMF record, see "Java Runtime Statistics" on page 17.

Java SDK Options and System Properties

Java options and system properties are configured in the batch launcher via the IBM_JAVA_OPTIONS environment variable. These options may include most options found on the standard 'java' command line launcher, including -X and -D options. The table below includes options commonly used with the batch launcher; refer to the complete list provided by 'java -help' and 'java -X' commands or to the IBM Java SDK Diagnostic Guides for more information.

Table 5. Java SDK Options and System Properties

Option	Description
-verbose:class	This option will print classloader activity to //SYSOUT, which is option helpful for resolving "ClassNotFound" errors.
-D<name>=<value>	Used to set any Java system property, which are then accessible within Java by using the System.getProperty() method.

Table 5. Java SDK Options and System Properties (continued)

Option	Description
-Dfile.encoding=<encoding>	Used to set the default file encoding. The default is usually IBM-1047, an EBCDIC encoding, but it is common to set this to ISO-8859-1, an ASCII encoding. See "Controlling Output Encoding" on page 14 for more information.
-Djzos.logging={E|W|N|I|D|T}	This optional property can be used to control logging in the JZOS toolkit native library to //SYSOUT. This level is independent from the launcher logging level. See "Setting Batch Launcher Logging Levels" on page 8 for more information.
jzos.merge.sysout={true|false}	If set to true when running under the batch launcher, Java's System.out and System.err are redirected to DD:SYSOUT and DD:STDOUT/DD:STDERR are ignored. Default: false.
-Xms<size>	Sets the initial Java heap size.
-Xmx<size>	Sets the maximum Java heap size.

Customizing a Reusable Configuration Script for Your Installation

It's often a good idea to create a shell script that handles most, if not all, of the environment variable configuration for your installation's batch Java jobs. The sample shell script jzos_config.sh (available from the website) can be customized to meet your installation requirements. The following example assumes that you have customized this script and placed it in the /etc directory:

```
//jobname JOB ...
//stepname EXEC PROC=JVMPRC70,
//  JAVACLS='com.ibm.jzos.sample.HelloWorld'
//STDENV DD *
APP_HOME=/usr/local/apps/myapp
. /etc/jzos_config.sh
//
```

Setting the Desired Batch Launcher Working Directory

To set the desired working directory of your Java application, include a "cd" command for that directory in STDENV. If multiple "cd" commands are found in the STDENV, the directory referenced by the last "cd" command will be used as the desired working directory.

```
//jobname JOB ...
//stepname EXEC PROC=JVMPRC80,
// JAVACLS='com.ibm.jzos.sample.HelloWorld'
//STDENV DD *
cd /tmp
export JAVA_HOME=/usr/lpp/java/J8.0
export PATH=/bin:"${JAVA_HOME}"/bin
//
```

In the above example, "/tmp" is set as the desired working directory.

Files Used by the Batch Launcher

The following DD names are used by the Java batch launcher:

Table 6. Files Used by the Batch Launcher

DD Name	Description
SYSOUT	(Output, Required) Messages from the batch launcher and any system messages that are written to the UNIX stderr file descriptor.
SYSPRINT	(Output, Optional) Any system messages which are written to the UNIX stdout file descriptor. This is not normally used.
STDOUT	(Output, Required) The output from Java System.out. This data is translated to the JZOS_OUTPUT_ENCODING codepage. See “Controlling Output Encoding “Controlling Output Encoding” on page 14 for more information.
STDERR	(Output, Required) The output from Java System.err. This data is translated to the JZOS_OUTPUT_ENCODING codepage. See “Controlling Output Encoding” on page 14 for more information.
STDENV	(Input, Required) A UNIX shell script used to configure environment variables. See “Configuring Environment Variables” on page 8 for more information.
STDIN	(Input, Optional) The input to Java System.in. This data is translated from the JZOS_OUTPUT_ENCODING codepage to the default Java file.encoding codepage. See “Controlling Output Encoding” on page 14 for more information.
MAINARGS	(Input, Optional) Can be used to supply arguments to the main Java class. See “Specifying the Java main class and its arguments” on page 7 for more information.

Since the Java Virtual machine is executed under the same address space as the parent batch job step, additional DD names can be provided which can be accessed by the Java program. See “MVS Data Set I/O” on page 23 for more information.

MVS Console Interface

By default, the batch launcher establishes an environment for receiving the following MVS operator commands:

- START – If running under an MVS started task, a callback interface allows a Java application to access the parameters on the START command.
- STOP (P) – By default, runs the method System.exit(0), which shuts down the JVM. A callback interface allows an application to customize this behavior.

- MODIFY (F) – A callback interface allows a Java application to receive and process these commands.

The JZOS toolkit also includes an API for issuing single or multiline WTO messages to the system console or log.

For more information on the Java APIs for WTOs and MVS operator commands, see: "Writing Messages to the System Console or Log" on page 26 and "Handling MVS START and MODIFY Commands" on page 26.

Controlling Output Encoding

In a Java VM, regardless of the platform, Strings and characters are represented in Unicode. Since different platforms use different character set encoding natively, the following mechanisms are used to control encoding:

- The Java `file.encoding` system property. This property is used as the default charset any time characters need to be converted to/from bytes. On z/OS, the default `file.encoding` is some variant of the EBCDIC character sets (IBM-1047, IBM-273, etc...). On Windows and most UNIX platforms the default `file.encoding` is some variant of the ASCII character set (Cp1242, ISO8859-1, etc...) To change this default, the java option `Dfile.encoding=` can be supplied to the VM on startup (See "Java SDK Options and System Properties" on page 11 for more information).
- The JZOS batch launcher redirects the JVM's System.out and System.err to DDs //STDOUT and //STDERR respectively. When these PrintStreams are redirected, JZOS modifies them to use the encoding returned by the method `Zutil.getDefaultPlatformEncoding()`. By default, this is the encoding of the current locale, which for many installations is "IBM-1047" (assuming that the LANG=C system environment variable is set). This default can be modified by exporting the environment variable `JZOS_OUTPUT_ENCODING` in the //STDENV configuration script.

  Since `java.io.PrintStream` has the unfortunate history of also supporting interfaces for writing raw bytes, the batch launcher will also transcode raw bytes from the current Java `file.encoding` to the JZOS default platform encoding. This transcoding is only available if both codesets are single-byte encodings, and may be disabled by setting the environment variable `JZOS_ENABLE_OUTPUT_TRANSCODING=false`.
- Java coding best practices are to not assume a particular default `file.encoding`, but it is not uncommon for Java code to assume an ASCII `file.encoding`. This can happen in subtle ways, such as in generating or parsing XML without specifying an encoding. It is often necessary to run these applications with an ASCII `file.encoding` of ISO-8859-1. Some widely used Java applications, such Apache Tomcat, include code that requires this.
- When running with an ASCII default `file.encoding`, applications must specifically use an EBCDIC encoding when using MVS datasets encoded in EBCDIC. The `Zutil.getDefaultPlatformEncoding()` method should be used to obtain the current "platform" encoding for this purpose.

Recommendations

1. Avoid writing code that assumes a default `file.encoding`, but if you need to run code that does, run with `-Dfile.encoding=ISO-8859-1`. There is really no penalty for doing this, since internal Unicode must be translated to something anyway.

2. When accessing MVS datasets, specify the encoding as `Zutil.getDefaultPlatformEncoding()`. The JZOS Toolkit portable file IO classes are already implemented to use this platform encoding for MVS datasets. See "Platform Independent Text File I/O with FileFactory" on page 24 for more information.

Messages, Return Codes, and Abnormal Termination

The batch launcher will complete under one of the following circumstances:

1. The Java main() method invoked by the launcher returns, and all of the non-daemon Java threads have completed.
2. A `System.exit(rc)` message was issued directly by the application, or in response to a MVS console STOP(P) command.
3. The Java main() thread terminates due to an uncaught exception (see below).
4. An error occurred in the launcher (see below).
5. An abend occurred in the launcher or JVM. For information on abends in the JVM, refer to the IBM Java SDK Diagnostic Guides.

When batch launcher completes normally, it will emit a return code.

If the batch launcher terminates due to an uncaught exception in the Java main method, SYSOUT will contain the message:

```
JVMJZBL1047W JZOS batch launcher completed with Java exception, return code=100
```

If the launcher itself fails, SYSOUT will contain the message:

```
JVMJZBL1042E JZOS batch launcher failed, return code=nnn
where nnn is one of the codes described in Table 7.
```

If the launcher completes without an internal error, the return code set by Java -- via `System.exit(rc)` will be returned and SYSOUT will contain the message:

```
JVMJZBL1021N JZOS batch launcher completed, return code=0
```

To prevent a Java exit code from matching a JZOS exit code, avoid the range 100 - 102.

Table 7. Exit Codes

RC	Name	Notes®
0	**RC_OK**	The Java main() method invoked by the launcher returned normally, or a System.exit() or System.exit(0) message was used to shutdown the JVM.
100	**RC_MAIN_EXCEPTION**	The Java main class not found or main method threw an exception.
101	**RC_CONFIG_ERR**	A configuration or setup error occurred. Check SYSOUT messages for more diagnostic information.
102	**RC_SYSTEM_ERR**	A system or internal error occurred. Check SYSOUT messages for more diagnostic information.

In version 2.4.5 of JZOS (available in IBM SDK, Java Technology Edition, Version 7 Release 1) a new optional environment variable, JZOS_ABEND_EXIT, may be used to force the batch launcher to ABEND under certain error conditions.

The environment variable may be specified as: JZOS_ABEND_EXIT=n
Where n is a positive integer between 0 and 99, inclusive. Use of any other values will be ignored.

If this environment variable is not set, then there will be no change in the current functionality.

If this environment variable is set, and the exit code from Java is either negative or greater than the value specified in JZOS_ABEND_EXIT=n, then the JZOS Batch Launcher will terminate with: ABEND U3333-rc Where rc is the non-zero exit/condition code that would have otherwise been used to terminate the job step.

Note: The rc value is an unsigned hexadecimal value 0-4095, whereas the exit code from Java is a signed int. This value is mapped using the normal system conventions. The rc will be the unsigned representation of the lower order 12 bits of the exit code.

The ABEND will be issued along with an option to suppress LE and system dumps. See clean-up option 3 for Language Environment® Callable Service CEE3AB2 in z/OS Language Environment Programming Reference, for more information.

The following example sets the environment variable JZOS_ABEND_EXIT=99:

```
// EXEC PROC=JVMPRCxx,JAVACLS='com.package.MyClass',
//CEEOPTS DD *
ENVAR("JZOS_ABEND_EXIT=99")
//STDENV *
...
//
```

Note:

1. In the example above, any System.exit(n) value 0-99 will result in normal step completion with CC=0-99. An exit code of 100 or greater, such as uncaught exceptions or System.exit(1000), will result in an ABEND.
2. The environment variable should be specified in LE options to the JZOS batch launcher as in the example above rather than in the STDENV script, in case there is a failure processing the script.

Language Environment Runtime Options

The JZOS batch launchers are built to specify LE runtime options that are designed to match their Java command-line launcher counterparts.

Changes to these LE runtime options must be made prior to executing the batch launcher. The following example demonstrates how to generate a storage report when the application exits:

```
//jobname JOB ...
//stepname EXEC PROC=JVMPRC80,
//  JAVACLS='com.ibm.jzos.sample.HelloWorld',
//  LEPARM='RPTOPTS(ON),RPTSTG(ON)'
//STDENV DD *
...
//
```

Refer to the IBM Java SDK Diagnostic Guides for information on tuning the Java virtual machine.

Note: Under z/OS 1.7 or later, the "CEEOPTS" DD may be used to specify a input file containing LE runtime options.

Java Runtime Statistics

A JZOS environment variable, HJV_JZOS_JVM_SMF_LOGGING, may be used to enable the logging of SMF records containing Java runtime statistics. By default, logging of Java runtime statistics is disabled. If this environment variable is set to true, the JZOS batch launcher will register a JVM shutdown hook to log a SMF record containing Java runtime performance statistics right before the JVM shuts down.

In addition to using HJV_JZOS_JVM_SMF_LOGGING to enable logging of a SMF record containing Java runtime statistics before JVM shutdown, HJV_JZOS_JVM_SMF_LOGGING_INTERVAL may be used to enable periodic logging of SMF records. Another environment variable, HJV_JZOS_JVM_SMF_THREADS, controls whether individual per thread detail will be included in these SMF records. Finally, HJV_JZOS_JVM_SMF_THREADS_NATIVE_ID controls whether the correct native OS thread ID information is assigned in the individual per thread detail section. For more information on how to configure these JZOS environment variables, refer to Table 4 on page 10.

SMF Record Type 121 Subtype 1

JZOS Java Runtime Performance Statistics
Record Mapping

Header/Self-defining Section

This section contains the common SMF record header fields and triplet fields (offset to section/length of section/number of sections) that locate the other sections on the record.

Table 8. SMF 121 Header/Self-defining Section

Offsets		Name	Length	Format	Description
0	0	SMF121LEN	2	binary	Record length (maximum size of 32,756). This field and the next field (total of four bytes) form the record descriptor word (RDW). The first two bytes (this field) must contain the logical record length including the RDW.
2	2	SMF121SEG	2	binary	Segment descriptor provided by SMF. Initialize with zeros.
4	4	SMF121FLG	1	binary	System indicator **Bit** **Meaning When Set** **0** Reserved. **1** Subtypes are valid. **2** Reserved. **3** MVS/SP Version 4 and above. Bits 3, 4, 5, and 6 are on.* **4** MVS/SP Version 3. Bits 4, 5, and 6 are on. **5** MVS/SP Version 2. Bits 5 and 6 are on. **6** VS2. Bit 6 is on. **7** Reserved. *IBM recommends that you use record type 30 to obtain the MVS product level.
5	5	SMF121RTY	1	binary	Record type. This should be 121 decimal.
6	6	SMF121TME	4	binary	Time since midnight, in hundredths of a second, that the record was moved into the SMF buffer.

Table 8. SMF 121 Header/Self-defining Section (continued)

Offsets		Name	Length	Format	Description
10	A	SMF121DTE	4	packed	Date when the record was moved into the SMF buffer, in the form 00*yydddF* or 0*cyydddF* (where c is 0 for 19xx and 1 for 20xx, *yy* is the current year (0-99), *ddd* is the current day (1-366), and *F* is the sign).
14	E	SMF121SID	4	EBCDIC	System identification.
18	12	SMF121SSI	4	EBCDIC	Subsystem identification.
22	16	SMF121STY	2	binary	Record subtype. This should be 1 decimal.
24	18	SMF121SDS_TRIPLETS	2	binary	Number of triplets (Offset/Length/Number combos). In this case, we have 3 triplets. One for the Java Runtime section, one for the Garbage Collector section, and one for the Thread section.
26	1A	SMF121SDS_RSERVD	2	binary	Reserved to account for fullword alignment of next field.
28	1C	SMF121SDS_OFFJRS	4	binary	Offset to the Java Runtime section.
32	20	SMF121SDS_LENJRS	2	binary	Length of each Java Runtime section.
34	22	SMF121SDS_NUMJRS	2	binary	Number of Java Runtime sections. We should only have 1 Java Runtime section.
36	24	SMF121SDS_OFFGCS	4	binary	Offset to the Garbage Collector section.
40	28	SMF121SDS_LENGCS	2	binary	Length of each Garbage Collector section.
42	2A	SMF121SDS_NUMGCS	2	binary	Number of Garbage Collector sections. This depends on how many Garbage Collectors are active in the JVM.
44	2C	SMF121SDS_OFFTS	4	binary	Offset to the Thread section.
48	30	SMF121SDS_LENTS	2	binary	Length of each Thread section.
50	32	SMF121SDS_NUMTS	2	binary	Number of Thread sections. This depends on the number of active Java threads.

Java Runtime Section

Table 9. SMF 121 Java Runtime section

Offsets		Name	Length	Format	Description
0	0	SMF121JRS_FDFLAGS	4	binary	Field flags to indicate the addition of new fields. Flag byte 1 **Bit** **Meaning When Set** **0** Contains CPU usage summary fields. Fields include SMF121JRS_APPCPU, SMF121JRS_SYSCPU, SMF121JRS_GCCPU, and SMF121JRS_JITCPU. **1-7** Reserved. Flag bytes 2, 3, and 4 are reserved.
4	4	SMF121JRS_NAME	80	EBCDIC	Formatted name representing the running Java virtual machine, in the format pid@hostname. Retrieved from java.lang.management.RuntimeMXBean::getName(). If longer than 80 characters, it will be truncated.
84	54	SMF121JRS_STRTTME	8	binary	The approximate time when the Java virtual machine started, in milliseconds. Retrieved from java.lang.management.RuntimeMXBean::getStartTime().
92	5C	SMF121JRS_UPTIME	8	binary	Uptime of the Java virtual machine in milliseconds. Retrieved from java.lang.management.RuntimeMXBean::getUptime().
100	64	SMF121JRS_GCMODE	40	EBCDIC	The current Garbage Collection mode as a human-readable string. Retrieved from com.ibm.lang.management.MemoryMXBean::getGCMode(). If longer than 40 characters, it will be truncated.
140	8C	SMF121JRS_PEAKTHRD	4	binary	The peak live thread count since the Java virtual machine started or peak was reset. Retrieved from java.lang.management.ThreadMXBean::getPeakThreadCount().
144	90	SMF121JRS_CURRTHRD	4	binary	The current number of live threads including both daemon and non-daemon threads. Retrieved from java.lang.management.ThreadMXBean::getThreadCount().

Table 9. SMF 121 Java Runtime section (continued)

Offsets		Name	Length	Format	Description
148	94	SMF121JRS_APPCPU	8	binary	Total CPU usage for all application threads in microseconds. Retrieved from com.ibm.lang.management.JvmCpuMonitorInfo::getApplicationCpuTime(). If CPU usage information is not available, this field will contain -1.
156	9C	SMF121JRS_SYSCPU	8	binary	Total CPU usage of all system threads in microseconds, which includes GC, JIT and other JVM daemon threads. Retrieved from com.ibm.lang.management.JvmCpuMonitorInfo::getSystemJvmCpuTime(). If CPU usage information is not available, this field will contain -1.
164	A4	SMF121JRS_GCCPU	8	binary	Total CPU usage of all GC threads in microseconds. Retrieved from com.ibm.lang.management.JvmCpuMonitorInfo::getGcCpuTime(). If CPU usage information is not available, this field will contain -1.
172	AC	SMF121JRS_JITCPU	8	binary	Total CPU usage of all JIT threads in microseconds. Retrieved from com.ibm.lang.management.JvmCpuMonitorInfo::getJitCpuTime(). If CPU usage information is not available, this field will contain -1.

Garbage Collector Section

Table 10. SMF 121 Garbage Collector section

Offsets		Name	Length	Format	Description
0	0	SMF121GCS_FDFLAGS	4	binary	Field flags to indicate the addition of new fields, currently should be all zeros.
4	4	SMF121GCS_NAME	40	EBCDIC	Garbage Collector name. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getName(). If longer than 40 characters, it will be truncated.
44	2C	SMF121GCS_COLLCNT	8	binary	Total number of collections that have occurred. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getCollectionCount().
52	34	SMF121GCS_COLLTME	8	binary	The approximate accumulated collection elapsed time in milliseconds. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getCollectionTime().
60	3C	SMF121GCS_TMEMFREED	8	binary	The cumulative total amount of memory freed, in bytes, by the garbage collector. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getTotalMemoryFreed().
68	44	SMF121GCS_TCOMPACTS	8	binary	The cumulative total number of compacts that was performed by the garbage collector. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getTotalCompacts().
76	4C	SMF121GCS_MEMUSED	8	binary	A snapshot of the amount of heap memory used by objects that are managed by the garbage collector right before this SMF record is recorded. Retrieved from com.ibm.lang.management.GarbageCollectorMXBean::getMemoryUsed().

Thread Section

The number of thread sections may be less than the number of Java threads because the number of thread sections is limited by the maximum length allowed for a SMF record.

Table 11. SMF 121 Thread section

Offsets		Name	Length	Format	Description
0	0	SMF121TS_FDFLAGS	4	binary	Field flags to indicate the addition of new fields, currently should be all zeros.
4	4	SMF121TS_ID	8	binary	Java thread ID. Retrieved from java.lang.management.ThreadInfo::getThreadId().
12	C	SMF121TS_NAME	24	EBCDIC	Java thread name. Retrieved from java.lang.management.ThreadInfo::getThreadName(). If longer than 24 characters, it will be truncated.

Table 11. SMF 121 Thread section (continued)

Offsets		Name	Length	Format	Description
36	24	SMF121TS_CAT	8	EBCDIC	Thread category. Retrieved from com.ibm.lang.management.JvmCpuMonitorMXBean::getThreadCategory(long). Possible thread categories include: APP APP-U1 APP-U2 APP-U3 APP-U4 APP-U5 SYS GC JIT OTHER RM If thread category information is not available, this field will contain an empty string.
44	2C	SMF121TS_CPU	8	binary	Total CPU usage time. This field has nanosecond precision but may not have nanosecond accuracy. Retrieved from java.lang.management.ThreadMXBean::getThreadCpuTime(long). If CPU usage information is not available, this field will contain -1.
52	34	SMF121TS_NATIVEID	8	binary	The corresponding native OS thread ID. If this information is not available, the field will contain -1.

A sample assembler DSECT for SMF record type 121 subtype 1 can be found in <JAVA_HOME>/mvstools/samples/smf.

For more information related to SMF, refer to z/OS MVS System Management Facility, SA38-0667.

Troubleshooting

Classpath problems:

- Run the batch launcher with LOGLVL='+I' to display the CLASSPATH and other environment variables prior to starting the JVM.
- Try running your job with verbose: class added to your IBM_JAVA_OPTIONS environment variable. This will write system classloader messages to //SYSOUT, which can be useful in determining which class is really missing.

Batch launcher problems:

- Run the batch launcher with LOGLVL='+T'. This will trace execution of the batch launcher to //SYSOUT.
- Check that the batch job's userid is properly configured to use UNIX System Services. Test the userid by using it to login into the UNIX shell. Refer to the z/OS UNIX System Services Planning for more information.
- Verify that the version of Java that your are trying to use has been properly installed and configured. Check that the latest maintenance has been installed. Login to a UNIX shell with the userid you are trying to use and issue the following commands (substituting your installation's Java home directory):

```
export JAVA_HOME=/usr/lpp/java/J8.0
$JAVA_HOME/bin/java -version
$JAVA_HOME/bin/java -cp $JAVA_HOME HelloWorld
```

Environment variable / STDENV shell script problems:

- Add a line 'set -x' to the beginning of your shell script, which will trace the shell script execution to //STDOUT. LOGLVL must be set to +D or +T for this output to be displayed.

- If after setting LOGLVL='+T' you find that the configuration child process is hanging or failing, check that you are properly setting required system environment variables. See “Configuring Environment Variables” on page 8 for more information.

Chapter 4. Toolkit User's Guide

The JZOS Toolkit is a Java Native Interface (JNI) library, consisting of a Java archive (JAR) file and native dynamic library. This toolkit includes:

- A high performance interface for reading and writing sequential data sets in record mode.
- Low-level wrappers for the z/OS C library I/O functions.
- A factory class for creating portable Readers and Writers to text files, including MVS datasets.
- Methods for allocating, deleting, and renaming MVS datasets.
- Methods for issuing single and multi-line WTOs.
- A callback interface for handling MVS Start, Modify, and Stop commands.
- An interface to z/OS Catalog Search (IGGCSI00).
- Class com.ibm.jzos.ZUtil, which contains APIs for many z/OS functions:
  - Obtaining job/step/user names, process ids, etc.
  - Writing SMF records
  - Reading environment variables
  - Reading OS Memory
- Class com.ibm.jzos.PdsDirectory for reading partitioned dataset directories.
- Class com.ibm.jzos.ZLogstream for reading, writing and deleting z/OS log streams.
- Classes in package com.ibm.jzos.wlm for access to z/OS Workload Manager (WLM) services.
- Class com.ibm.jzos.MvsJobSubmitter for submitting z/OS batch jobs from Java.
- Package com.ibm.jzos.fields containing classes for converting Cobol and Assembler datatypes to Java objects.
- Class com.ibm.jzos.DfSort, an interface for invoking DFSORT and directing I/O into/from Java applications.
- Class com.ibm.jzos.AccessMethodServices, which is an interface for invoking IDCAMS.
- Class com.ibm.jzos.Enqueue for serializing and releasing z/OS resources using the ISGENQ service.

The following sections provide an overview of the JZOS toolkit classes, refer to the HTML javadoc documentation provided at www.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html for more information.

MVS Data Set I/O

The standard java.io package can only be used to access files in the HFS or zFS filesystem. The JZOS toolkit complements this package by providing classes that allow Java applications to interact with MVS data sets. Java programs can use JZOS to access any MVS data set supported by the C/C++ library, including:

- Partitioned Data Set (PDS)
- Partitioned Data Set Extended (PDSE)
- Sequential Files
- Virtual Sequential Access File (VSAM) of the type KSDS, RRDS, or ESDS

The JZOS I/O classes support several models of I/O when using MVS data sets.

Record mode: Each read or write processes a single record of a data set.

Stream mode: Data set records are presented as a stream of bytes. Each read or write reads some portion of those bytes, irrespective of record boundaries. Stream mode is further distinguished by two types:

- Text (stream) mode - Data set records are converted to a stream of bytes and a "new line" record delimiter is placed in the stream between records after trailing blanks are removed.
- Binary (stream) mode - Data set records are placed in the stream as is, without record separators.

General Data Set Access with ZFile

The JZOS class com.ibm.jzos.Zfile is a general purpose class that wraps the z/OS C/C++ Library I/O routines – fopen(), fread(), fwrite(), etc., for accessing MVS data sets in stream (text or binary) and in record mode. Javadoc for the ZFile class refers to each C/C++ library I/O routine that is called, so that the Java programmer may consult the documentation for these routines directly. See the z/OS XL C/C++ Run-Time Library Reference, SA22-7821 and z/OS XL C/C++ Programming Guide, SC09–4765 for more information.

Platform Independent Text File I/O with FileFactory

The JZOS class com.ibm.jzos.FileFactory allows for portable creation of streams, readers and writers on POSIX/DOS files (using the java.io package), or MVS data sets (using the ZFile class in text mode). The FileFactory class determines which underlying file system to use based on the file name, so that a portable application can be configured at run time to use the appropriate filename / filesystem.

High Speed Data Set Record I/O with RecordReader and RecordWriter

In version 2.4.2 of JZOS (available in z/OS Java SDK 6.0.1 SR1 and z/OS Java SDK 7.0.0 SR1) new classes have been added to provide a high performance interface for reading and writing sequential data sets in record mode. These classes (com.ibm.jzos.RecordReader and com.ibm.jzos.RecordWriter) use the native z/OS Basic Sequential Access Method (BSAM) for data set interaction and should be considered where very high performance record mode I/O is required by a Java application (in some cases providing a 60-70% reduction in CPU usage over ZFile). The best results from the RecordReader and RecordWriter classes will be seen on data sets with a relatively large blocking factor.

Usage Recommendations

- For POSIX file stream style access to MVS data sets, use ZFile (or the simpler and more platform independent FileFactory if only text mode access is required).
- For sequential record mode access to data sets (both text and binary), use the RecordReader and RecordWriter classes. Use ZFile in cases where applications require functionality not provided by these classes (e.g. positioning).
- For generalized access to VSAM data sets (KSDS, RRDS, or ESDS), use ZFile.

The following examples illustrate some common use cases.

Example: Reading a data set in text stream mode using FileFactory

```
BufferedReader rdr = FileFactory.newBufferedReader("//DD:INPUT");
try {
  String line;
  while ((line = rdr.readLine()) != null) {
    System.out.println(line);
  }
} finally {
  rdr.close();
}
```

Example: Processing a data set in binary stream mode using Zfile

```
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
MyDocumentHandler handler = new MyDocumentHandler();
ZFile zFile = new Zfile("//MY.XML.DATA", "rb");
try {
    parser.parse(zFile.getInputStream(), handler);
} finally {
    zFile.close();
}
```

Example: Read a data set in record mode using a RecordReader

```
RecordReader reader = null;
try {
  reader = RecordReader.newReader("//my.dataset", ZFileConstants.FLAG_DISP_SHR);
  byte[] recordBuf = new byte[reader.getLrecl()];
  while ((bytesRead = reader.read(recordBuf)) >= 0) {
    ...
  }
} finally {
  if (reader != null) {
    reader.close();
  }
}
```

Example: Create a new data set and write to it using a RecordWriter

```
String ddname = ZFile.allocDummyDDName();
String cmd = "alloc fi("+ddname+") da(HLQ.MYDATA) reuse new catalog msg(2)"
       + " recfm(f,b) space(100,50) cyl"
       + " lrecl(80)";
ZFile.bpxwdyn(cmd);  // might throw RcException
RecordWriter writer = null;
try {
  writer = RecordWriter.newWriterForDD(ddname);
  byte[] recordBuf = new byte[writer.getLrecl()];
  int bytesToWrite;
  while ((bytesToWrite = getNextAppRecord(recordBuf)) > 0) {
    writer.write(recordBuf, 0, bytesToWrite);
  }
} finally {
  if (writer != null) {
    try {
      writer.close();
    } catch (ZFileException zfe) {
      zfe.printStackTrace();  // but continue
    }
  }
  try {
    ZFile.bpxwdyn("free fi(" + ddname + ") msg(2)");
  } catch (RcException rce) {
    rce.printStackTrace();  // but continue
  }
}
```

Writing Messages to the System Console or Log

The toolkit contains a method for issuing single or multi-line write-to-operator (WTO) messages to the system console or log. The message String is automatically broken on word boundaries as required to fit on multiple lines and converted to the platform encoding.

Example: Issuing an MVS WTO

```
MvsConsole.wto("FOO1233E processing complete. Since this is a rather"
         + "long message, it will broken into a multi-line WTO on
         + "a word boundary as required." ,
         0x0020,  // routecde
         0x4000); // descriptor code
```

Handling MVS START and MODIFY Commands

A callback interface is provided for a Java application that needs to receive and process MVS MODIFY (F) commands. If the application was started as an MVS started task, then the parameters on the START command will be sent to the `handleStart()` callback method as soon as it is registered.

Example: Handling MVS START, STOP, and MODIFY commands

```
final String startCmd = null;

MvsConsole.registerMvsCommandCallback(new MvsCommandCallback() {
    public void handleModify(String s) {
        System.out.println("Received Modify command: " + s); }
    public void handleStart(String s) {
        startCmd = s; }
    public boolean handleStop() {
        return true; // so that System.exit(0) is done
    }
});
System.out.println("Start command options = " + s);
```

JZOS Sample Programs

Several sample programs are available for download at http://www-03.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html.

Hints and Tips for Editing ASCII Files under z/OS

XML and Java properties files are normally stored in the HFS (UNIX) filesystem in codepage ISO8859-1 (ASCII).

Here are several approaches for editing ASCII files under z/OS

1. Convert the file from ASCII to EBCDIC before editing and back again when done. For example:

   ```
   iconv -f ISO8859-1 -t IBM-1047 myfile.properties >
   myfile.properties.a
   vi myfile.properties.a (or oedit if under a 3270 OMVS shell)
   iconv -f IBM-1047 -t ISO8859-1 myfile.properties.a >
   myfile.properties
   ```

   The "atools" package, available from the IBM "UNIX Tools and Toys" download site provides small shell scripts that automate this process.
2. Tag the file as ASCII text and then enable automatic conversion.

```
chtag -tc ISO8859-1 myfile.properties (you only have to tag a given file once)
export _BPXK_AUTOCVT=ON (this environment variable enables
automatic conversion)
vi myfile.properties
```

However, be careful to not set _BPXK_AUTOCVT=ON in your actual JVM process, as Java will not expect automatic conversion of properties and XML files.

For more information, see z/OS enhanced ASCII Functionality.

3. Edit the file in an IDE, such as Eclipse, and then deploy it to z/OS using an Ant script by using the Ant FTP task.

   For more information on the Ant FTP task: http://ant.apache.org/manual/OptionalTasks/ftp.html

   The JZOS Cookbook, available on developerWorks®, includes an example Eclipse project and guided instructions on using Eclipse as an IDE with z/OS Java: https://www.ibm.com/developerworks/community/groups/service/

4. Beginning in z/OS V1R9, ISPF now supports ASCII editing of HFS or zFS files.

z/OS CMPSC Compression Algorithm

The JZOS toolkit provides a wrapper, com.ibm.jzos.ZCompressor, for the z/OS CMPSC static dictionary compression algorithm. Static dictionary compression algorithms, such as CMPSC, don't solve the same problems as adaptive compression algorithms. Static compression works well for compressing discrete chunks of data, whereas adaptive compression, such as (G)ZIP, “warms up” over the life of a stream of data to be compressed.

Any system where chunks of data are discretely processed (for example, DB2® pages, VSAM control intervals, SAM disk blocks) should benefit from a static compression algorithm.

Refer to z/Architecture Principles of Operation, SA22-7832 and Enterprise Systems Architecture/390 Data Compression, SA22-7208 for details on the compression algorithm and on the CMPSC machine instruction.

The SAMPLE REXX exec 'SYS1.SAMPLIB(CSRBDICT)' may be used to create a compression and expansion dictionary for a representative sample dataset. The sample JCL below may be used to create these dictionaries and then to concatenate them into a single dataset as required by ZCompressor. See the documentation included in the comments of the CSRBDICT or in the above references for more information.

```
//CSRBDICT JOB (),'USER'
//*
//* The following symbol defines the dataset containing the
//* text that you will scan.  This will also become the
//* prefix for other datasets that are created,
//* and assumes that you have created a ".SPECFILE"
//* dataset containing the specs for CSRBDICT.
//* In this example, we have downloaded the public domain text
//* of "Moby Dick" from http://www.gutenberg.org/files/2701/2701.txt
//* and we will generate dictionaries with 4K entries which implies
//* 12 bit symbol sizes, since 4096 = 2**(12+3) / 8.
//*
// SET DSNPREF=MYUSER.MOBYDICK
//*
//TSOTMP   EXEC PGM=IKJEFT1A,DYNAMNBR=20
//SYSPROC  DD  DSN=SYS1.SAMPLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 %CSRBDICT 4 1 EB 'MYUSER.MOBYDICK'
//*
//* Concatenate the dictionaries, selecting only
//* the leading (non-blank) 8 bytes from each entry record
//* This assumes that your specfile does NOT have the "asm"
//* option.  For this example, the total size in bytes of the
//* output dataset will be 64K (4K * 8 * 2).
//*
//CONCAT   EXEC PGM=SORT,COND=(4,LT)
//SORTIN   DD  DSN=&DSNPREF..ACDICT41,DISP=SHR
//         DD  DSN=&DSNPREF..AEDICT41,DISP=SHR
//SORTOUT  DD  DSN=&DSNPREF..CDICTS41,
//             DISP=(NEW,CATLG),SPACE=(CYL,(1)),
//             DCB=(LRECL=8,BLKSIZE=0,RECFM=FB)
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
 OPTION COPY
 INREC FIELDS=(1,8)
//
```

For the above sample JCL, here are the contents of the &DSNPREF..SPECFILE dataset:

```
** This is the TEXT FILE example from SYS1.SAMPLIB(CSRBDICT):
**    - removed "asm" option - output is 8-byte binary entries
**    - added   "opt"
**
**The following is with a 4K-entry dictionary.
**Provides 30.88% compression (output/input) for the source of
**Chapter 5 of the ESA/390 Principles of Operation (30.32% if all output
**bits are concatenated together).
**Optimization (change x under opt to opt) improves compression by 0.7%.
**results maxnodes maxlevels msglevel stepping prperiod dicts
  r       40000    60        3        f 7 2 7  1000     afd
**colaps opt treedisp treehex treenode dupccs
  aam    opt x        h       n        x
**FLD col type dcenmen              INT  intspec
  FLD 1   sa   dce 4                INT  aeis 1 (40)
                                    INT  a12b3s (40)
  FLD end
**Note: Some text will be compressed better if the INT aeis 1 (40) is
**omitted; i.e., try it with and without the INT aeis 1 (40).  Also, if
**the text is ASCII instead of EBCDIC, the 40 should be changed to 20.
```

Appendix A. Messages

Below are the messages issued by the JZOS batch launcher and / or toolkit. If you are using Notice level logging (the default), you will see the EWN messages. For more information, see “Setting Batch Launcher Logging Levels” on page 8. Message issued by the batch launcher are prefixed with "JVMJZBL" and those by the JZOS toolkit APIS are prefixed by "JVMJZTK".

Messages Issued by the JZOS Batch Launcher

1001N JZOS_MSG_VERSION "JZOS batch Launcher Version: %s".

Explanation: Copyright notice issued with launcher is started.

System action: No system action is taken.

Programmer response: No programmer response required.

1002N JZOS_MSG_COPYRIGHT "Copyright (C) IBM Corp. 2005. All rights reserved.".

Explanation: Copyright notice issued with launcher is started.

System action: No system action is taken.

Programmer response: No programmer response required.

1003E JZOS_MSG_NO_STDENV "Error opening //DD:STDENV, using existing environment."

Explanation: Issued when there is an error opening the STDENV or when no STDENV is supplied by the user.

System action: The launcher terminates.

Programmer response: No programmer response required.

1004E JZOS_MSG_SIGPIPE_SIG_IGN "Error setting SIGPIPE to SIG_IGN - %s".

Explanation: Issued when attempting to ignore SIGPIPE signals for child process pipes.

System action: The launcher terminates.

Programmer response: No programmer response required.

1005I JZOS_MSG_STDENV_OUTPUT "Output from DD:STDENV config shell script:".

Explanation: Debugging information from adoption of child environment.

System action: No system action is taken.

Programmer response: No programmer response required.

1006I JZOS_MSG_ENV_VAR "%s = %s".

Explanation: Informational message for environment variable and associated value.

System action: No system action is taken.

Programmer response: No programmer response required.

1007E JZOS_MSG_ENV_VAR_SET "Error setting env var: %s = %s - %s".

Explanation: Error message when environment variable cannot be set in launcher environment.

System action: The launcher terminates.

Programmer response: Correct the invalid environment variable if possible.

1008I JZOS_MSG_ENV_STOP_STRING "%s".

Explanation: Debugging info of environment stop string eyecatcher.

System action: No system action is taken.

Programmer response: No programmer response required.

1009E JZOS_MSG_STDENV_HAS_EXIT "Child shell process exited without printing environment; //STDENV should not contain 'exit'".

Explanation: Error message emitted when //STDENV contains an explicit exit call.

System action: The launcher terminates.

Programmer response: Remove the exit call from //STDENV.

1010E JZOS_MSG_SIGPIPE_SIG_DFL "Error setting SIGPIPE to SIG_DFL - %s".

Explanation: Issued when attempting to restore default signals for child process pipes.

System action: The launcher terminates.

Programmer response: No programmer response required.

1011E JZOS_MSG_CREATE_JVM_ERROR "JNI_CreateJavaVM error, rc = %d".

Explanation: Issued when JVM creation fails.

System action: The launcher terminates.

Programmer response: Look up the return code in the JNI documentation and fix the condition if possible.

1012I JZOS_MSG_JAVA_VERSION_HEADER "Java Virtual Machine created. Version information follows:".

Explanation: Header line emitted prior to Java SDK version information.

System action: No system action is taken.

Programmer response: No programmer response required.

1013I JZOS_MSG_SYSTEM_EXIT "Calling System.exit()".

Explanation: The launcher is preparing to force a System.exit() call because JZOS_GENERATE_SYSTEM_EXIT is set TRUE or there was a launcher error after the JVM was created.

System action: No system action is taken.

Programmer response: No programmer response required.

1014I JZOS_MSG_WAITING_FOR_THREADS "Waiting for non-deamon Java threads to finish before exiting...".

Explanation: The launcher is preparing to terminate, but will wait for non-daemon threads to complete.

System action: No system action is taken.

Programmer response: No programmer response required.

1015N JZOS_MSG_MVS_COMMANDS_DISABLED "MVS commands are DISABLED".

Explanation: The launcher will not establish an operator console waiter.

System action: No system action is taken.

Programmer response: No programmer response required.

1016I JZOS_MSG_MVS_COMMANDS_ENABLED "MVS commands are ENABLED".

Explanation: The launcher has established an operator console waiter.

System action: No system action is taken.

Programmer response: No programmer response required.

1017E JZOS_MSG_JVM_ARGUMENTS "Could not get default JVM arguments".

Explanation: The JNI_GetDefaultJavaVMInitArgs() failed.

System action: The launcher terminates.

Programmer response: No programmer response required.

1018E JZOS_MSG_NO_CLASSPATH_ENV_VAR "CLASSPATH environment variable is not set".

Explanation: The CLASSPATH environment variable, which is required, was not set.

System action: The launcher terminates.

Programmer response: Set the CLASSPATH environment variable.

1019E JZOS_MSG_CLASSPATH_ALLOC_FAILURE "Storage for classpath could not be allocated".

Explanation: Heap storage for the JVM classpath could not be obtained.

System action: The launcher terminates.

Programmer response: Ensure enough storage is available.

1020E JZOS_MSG_CLASS_NOT_FOUND "Java class not found: %s".

Explanation: The specified java class could not be found.

System action: The launcher terminates.

Programmer response: Ensure that the required class is in the classpath.

1021 JZOS_EXIT_LAUNCHER_COMPLETED_RC0 "JZOS batch launcher completed, return code=0".

Explanation: The main method returned normally, batch launcher completed with return code = 0.

System action: The launcher completes.

Programmer response: None.

1022E JZOS_MSG_NEW_STRING_PLATFORM "NewStringPlatform failed with rc = %d".

Explanation: Issued when a call to the JNI function NewStringPlatform() fails.

System action: The launcher terminates.

Programmer response: Look up the JNI return code and fix the condition if possible.

1023I JZOS_MSG_MAIN_BEGIN "Invoking %s.main()...".

Explanation: Issued just prior to invocation of the java main class main().

System action: No system action is taken.

Programmer response: No programmer response required.

1024I JZOS_MSG_MAIN_END "%s.main() completed.".

Explanation: Issued just after java main class main() completes.

System action: No system action is taken.

Programmer response: No programmer response required.

1025I JZOS_MSG_TRANSCODING_DISABLED "Automatic output transcoding is DISABLED".

Explanation: The environment variable JZOS_ENABLE_OUTPUT_TRANSCODING has been set to false.

System action: No system action is taken.

Programmer response: No programmer response required.

1026W JZOS_MSG_INVALID_OUTPUT_ENCODING "Requested output encoding %s not valid. Using %s".

Explanation: Issued if the requested output encoding is not valid.

System action: No system action is taken.

Programmer response: Select a supported encoding.

1027I JZOS_MSG_OUTPUT_ENCODING "Using output encoding: %s".

Explanation: Issued to describe the output encoding in effect.

System action: No system action is taken.

Programmer response: No programmer response required.

1028I JZOS_MSG_REGION_REPORT_64 "Region requested = %s, Actual below/above limit = %s / %s, MEMLIMIT=%s".

Explanation: Issued to report the memory region in a 64 bit environment.

System action: No system action is taken.

Programmer response: No programmer response required.

1029I JZOS_MSG_REGION_REPORT_31 "Region requested = %s, Actual below/above limit = %s / %s".

Explanation: Issued to report the memory region in a 31 bit environment.

System action: No system action is taken.

Programmer response: No programmer response required.

1030E JZOS_MSG_MAIN_ARGS_TOO_LONG "Main arguments exceeded maximum length of %d".

Explanation: Issued when the length of main args exceeds the maximum limit.

System action: The launcher terminates.

Programmer response: Decrease the main argument length.

1031E JZOS_MSG_MAIN_ARGS_DD_MISSING "JZOS_MAIN_ARGS_DD %s does not refer to a valid DDNAME".

Explanation: Issued if the specified main args DD is not present in the job, or could not be opened.

System action: The launcher terminates.

Programmer response: Specify a DDNAME that exists.

1032E JZOS_MSG_MAIN_ARGS_NOT_TERMINATED "JZOS_MAIN_ARGS contains an unterminated string: %s".

Explanation: Issued if JZOS_MAIN_ARGS contains an unterminated string.

System action: The launcher terminates.

Programmer response: Fix the arguments.

1033E JZOS_MSG_JAVA_CLASS_MISSING "No Java class name argument supplied.".

Explanation: Issued if no Java class was specified for the launcher to call.

System action: The launcher terminates.

Programmer response: Specify a Java class.

1034E JZOS_MSG_PIPE_CREATION_FAILURE "Pipe creation failure: %s".

Explanation: Could not create a pipe for child environment process.

System action: The launcher terminates.

Programmer response: Refer to the reason error message for the cause; correct and retry.

1035E JZOS_MSG_SPAWN_FAILURE "Could not spawn: %s - %s".

Explanation: Could not spawn child environment process shell.

System action: The launcher terminates.

Programmer response: Refer to the reason error message for the cause; correct and retry.

1036D JZOS_MSG_SPAWN "Spawned child shell process with PID: %d".

Explanation: Debugging-level message containing spawned child's PID.

System action: No system action is taken.

Programmer response: No programmer response required.

1037E JZOS_MSG_CHILD_WAIT_ERROR "Error waiting for child shell process - %s".

Explanation: An error occurred waiting for the child environment process.

System action: The launcher terminates.

Programmer response: Refer to the reason error message for the cause; correct and retry.

1038E JZOS_MSG_CHILD_EXIT "Child shell process exited with exit code: %d".

Explanation: The child //STDENV script process exited with a non zero code.

System action: The launcher terminates.

Programmer response: Correct the //STDENV shell script and rerun the job.

1039E JZOS_MSG_CHILD_SIGNAL_RECEIVED "Child shell process received signal: %d".

Explanation: The child environment process received an unexpected signal.

System action: The launcher terminates.

Programmer response: Correct the //STDENV shell script and rerun the job.

1040E JZOS_MSG_CHILD_SIGNAL_STOPPED "Child shell process stopped with signal: %d".

Explanation: The child environment process stopped on an unexpected signal.

System action: The launcher terminates.

Programmer response: Correct the //STDENV shell script and rerun the job.

1041E JZOS_MSG_CHILD_WAIT_FAILURE "Child shell process exited with unexpected wait status code: %d".

Explanation: The child environment process exited with an unexpected wait status code.

System action: The launcher terminates.

Programmer response: Correct the //STDENV shell script and rerun the job.

1042E JZOS_EXIT_LAUNCHER_FAILED "JZOS batch launcher failed, return code=%d".

Explanation: The batch launcher failed with the given return code.

System action: The launcher terminates.

Programmer response: Refer to the previous message for the cause.

1043N JZOS_EXIT_JVM_SYSTEM_EXIT "The Java virtual machine completed with System.exit(%d)".

Explanation: The Java virtual machine completed with a Java System.exit(n).

System action: The launcher terminates with a return code equal to the JVM exit code.

Programmer response: None.

1044E JZOS_EXIT_JVM_ABORTED "The Java virtual machine aborted".

Explanation: The Java virtual machine completed by calling the launcher abort hook.

System action: The JVM generates a system abend, terminating the launcher job step.

Programmer response: None.

1045E JZOS_MSG_JVM_JAVA_VERSION_ERROR "Unable to retrieve Java version information".

Explanation: An error occurred when the launcher calls the JVM to retrieve Java SDK version information.

System action: The launcher terminates.

Programmer response: Ensure that the Java SDK is properly installed.

1046E JZOS_MSG_JVM_CONSOLE_LISTENER "Unable to establish MVS console listener".

Explanation: An error occurred when the launcher calls the JVM to establish the MVS console listener.

System action: The launcher terminates.

Programmer response: None.

1047W JZOS_EXIT_MAIN_EXCEPTION "JZOS batch launcher completed with Java exception, return code=%d".

Explanation: The batch launcher completed after an exception occurred when running the Java main method.

System action: The launcher completes with a return code = 100.

Programmer response: If possible correct the Java program and retry.

1048N JZOS_EXIT_JVM_NON_MAIN_SYSTEM_EXIT "Non main thread performed a System.exit(%d)".

Explanation: A non main Java thread performed a non zero system exit.

System action: The launcher terminates with a return code equal to the System.exit() value.

Programmer response: None.

1049W JZOS_BUILD_VERSION_MISMATCH "JZOS batch Launcher Version '%s' does not match jzos.jar Version '%s'".

Explanation: The version of the Batch Launcher doesn't match the version of JZOS in the SDK.

System action: None.

Programmer response: The versions should be brought together, possibly by installing appropriate PTFs.

1050E JZOS_ERROR_READING_JAR_FILE "Error reading jarfile \"%s\" for -jar option".

Explanation: There was an exception creating a java.util.JarFile object.

System action: The launcher terminates with a return code = 101.

Programmer response: None.

1051E JZOS_ERROR_READING_JAR_MANIFEST "Error reading jarfile \"%s\" manifest".

Explanation: There was an exception reading the Manifest for an executable jar file.

System action: The launcher terminates with a return code = 101.

Programmer response: None.

1052E JZOS_ERROR_READING_JAR_NO_MAIN "Error reading jarfile \"%s\" manifest - no Main-Class key".

Explanation: There was an exception reading the Manifest for an executable jar file.

System action: The launcher terminates with a return code = 101.

Programmer response: None.

1053I JZOS_MSG_OSNAME_REPORT "OS Release R%s Machine %s".

Explanation: Reports uname() data on OS and machine level.

System action: None.

Programmer response: None.

1054W JZOS_ENV_VAR_TOO_LONG "Environment variable %s exceeds maximum allowable length of %d".

Explanation: Warning message when an environment variable is read that exceeds the maximum length.

System action: None.

Programmer response: Correct the invalid environment variable if possible.

1055E JZOS_MAX_JVM_OPTIONS_EXCEEDED "JVM options exceed maximum length of %d".

Explanation: Issued when the number of JVM options exceeds the maximum limit.

System action: The launcher terminates.

Programmer response: Decrease the number of JZOS_JVM_OPTIONS and use IBM_JAVA_OPTIONS for more options.

1056I JZOS_MAIN_ARGS_FOLLOW "Arguments to main..."

Explanation: Displays the arguments to be supplied to the java Main class.

System action: None.

Programmer response: None.

1057I JZOS_MAIN_ARG "%s"

Explanation: Displays the value of one argument to the java Main class.

System action: None.

Programmer response: None.

1058E JZOS_ERROR_READING_DD_MAINARGS "Error reading MAINARGS file - \"%s\""

Explanation: There was an I/O error reading from DD:MAINARGS (or user supplied MAINARGS DD).

System action: The launcher terminates with a return code = 101.

Programmer response: None.

1059N JZOS_MSG_ISSUING_ABEND "Issuing ABEND U%d for exitCode=%d"

Explanation: The JZOS_ABEND_EXIT environment variable was set, causing an ABEND for exitCode < 0 or exitCode > JZOS_ABEND_EXIT.

System action: The launcher terminates with an ABEND U3333-exitCode.

Programmer response: None.

1060E JZOS_MSG_SMF_121_1_WRITER_REGISTRATION "Unable to register SMF 121.1 record writer with JVM hooks"

Explanation: An error occurred when the launcher calls the JVM to register the SMF 121.1 record writer with various JVM hooks.

System action: The launcher terminates.

Programmer response: None.

1061E JZOS_MSG_SMF_121_1_DUPLICATE_ENVAR "JZOS_JVM_SMF_LOGGING and HJV_JZOS_JVM_SMF_LOGGING environment variables were both set, only one should be set"

Explanation: The JZOS_JVM_SMF_LOGGING environment variable is deprecated and will be removed in a future release, use HJV_JZOS_JVM_SMF_LOGGING for improved forward compatibility.

System action: The launcher terminates.

Programmer response: Unset the JZOS_JVM_SMF_LOGGING environment variable and only use HJV_JZOS_JVM_SMF_LOGGING.

Messages Common/Shared by the JZOS Batch Launcher and Toolkit

2001E JZOS_CMN_MSG_MALLOC_ERROR "malloc() error in routine: %s - %s".

Explanation: A malloc() call failed to allocate heap storage.

System action: The launcher terminates.

Programmer response: Ensure enough storage is available.

2002D JZOS_CMN_CATOPEN_ERROR "Unable to open NLS catalog: \"%s\", using built-in English messages".

Explanation: The NLS message catalog could not be opened.

System action: The system continues with built-in English messages.

Programmer response: Ensure that the JZOS NLS catalog is installed properly.

2003D JZOS_CMN_CATENV_ERROR "Current NLS settings: LANG=%s, NLSPATH=%s".

Explanation: Prints out the NLS environment variables.

System action: None.

Programmer response: None.

2004I JZOS_CMN_LOGGING_LEVEL_CHANGED "Log level has been set to: %c".

Explanation: The logging level for the JZOS toolkit has been changed.

System action: None.

Programmer response: None.

2005I JZOS_CMN_LOGGING_LEVEL_INVALID "Invalid log level %c. Must be one of: %s".

Explanation: The logging level was configured with an invalid value.

System action: None; the previous/default level is retained.

Programmer response: Configure the logging level to a valid value.

2006E JZOS_CMN_THROWABLE_DESCRIPTION "An Exception occurred: %s".

Explanation: An exception occurred in a native method.

System action: None.

Programmer response: If possible correct the condition that caused the exception.

2007E JZOS_CMN_STACKTRACE_HEADER "Stack trace follows:".

Explanation: An exception occurred in a native method.

System action: None.

Programmer response: If possible correct the condition that caused the exception.

2008E JZOS_CMN_CLASS_NOT_FOUND "Could not find or load class: %s".

Explanation: A Java Native Interface (JNI) FindClass request failed, either because the class was not found, or failed class initialization.

System action: None.

Programmer response: Verify that the class is in the current classpath. Specifying the "verbose:class" JVM option can aid in diagnosing class loader problems.

2009E JZOS_CMN_METHOD_NOT_FOUND "Could not find method '%s' in class %s".

Explanation: A Java Native Interface (JNI) request to find a method failed.

System action: None.

Programmer response: If this error is for a launched java class, ensure that the class has a valid main method.

2010E JZOS_CMN_INVOKE_EXCEPTION "Exception occurred invoking %s.%s()".

Explanation: A method invoked via the Java Native Interface (JNI) threw an exception.

System action: None.

Programmer response: If possible correct the condition that caused the exception.

Messages not NLS Enabled

These diagnostic/trace messages are not NLS enabled.

Note: ZLog.h refers to this message by numeric set/message id.

2999T JZOS_CMN_DIAGNOSTIC_MSG "%s".

Explanation: Module diagnostic message. Normally "T"race level, but also other levels.

System action: None.

Programmer response: None.

Appendix B. Migration from developerWorks IBM Experimental JZOS Batch Toolkit for z/OS SDKs

Versions of experimental JZOS are also available on IBM developerWorks, which include new and additional functionality on an “as-is” basis for customer evaluation and feedback. Some of these new features may eventually be available in the SDK product version, but some features, such as the JZOS Cookbook, will not become part of the supported z/OS Java SDK products.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademark or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Online Privacy Statement Highlights at http://www.ibm.com/privacy and the IBM Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled “Cookies, Web Beacons and Other Technologies”, and the IBM Software Products and Software-as-a-Service Privacy Statement at http://www.ibm.com/software/info/product-privacy.

Bibliography

See the following publications for additional information.

The JZOS Cookbook, available on developerWorks: https://www.ibm.com/developerworks/community/groups/service/

IBM Java SDK Diagnostic Guides

z/OS UNIX System Services Planning, GA22-7800

z/OS Using REXX and z/OS UNIX System Services, SA22–7806

z/OS XL C/C++ Programming Guide, SC09–4765

z/OS Language Environment Programming Reference, SA38-0683

z/OS MVS System Management Facility, SA38-0667

Index

Printed in USA

SC27-8418-07