JZOS Batch Launcher and Toolkit
function in IBM SDK for z/OS

Installation and User's Guide

SA23-2245-05

JZOS Batch Launcher and Toolkit
function in IBM SDK for z/OS

Installation and User's Guide

SA23-2245-05

Note:
Before using this information and the product it supports, read the information in “Notices” on page 61.

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

This edition replaces SA23–2245–04.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+433-2856
FAX (Other Countries):
    Your International Access Code +1+845+433-2856

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/systems/z/os/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

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

Contents

About This Book

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.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS® elements and features, z/VM®, VSE/ESA, and Clusters for AIX® and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at www.ibm.com/systems/z/os/zos/bkserv/lookat/index.html.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX System Services).
- Your Microsoft Windows workstation. You can install LookAt directly from the *z/OS Collection* (SK3T-4269) or the *z/OS and Software Products DVD Collection* (SK3T-4271) and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from www.ibm.com/systems/z/os/zos/bkserv/lookat/lookatm.html with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection* (SK3T-4269).
- The *z/OS and Software Products DVD Collection* (SK3T-4271).
- The LookAt Web site (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Summary of Changes

This document is a refresh of JZOS Batch Launcher and Toolkit Installation and User's Guide, SA23-2245-02, which supports JZOS Version 2.3.0.

SA23–2245–05, Updated Edition

It contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Editorial Updates

Added information about using STEPLIB with JZOS Batch launcher program, see "Configuring Environment Variables" on page 8 in Chapter 3, "IBM JZOS Batch Launcher User's Guide."

Enhancements

These enhancements are all included in JZOS Version 2.4.2. Those products are IBM 31-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R31) and IBM 64-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R32).

- The JZOS Batch Launcher will now invoke the default UncaughtExceptionHandler set by the program if the main thread of the program encounters an uncaught Exception.

SA23–2245–04, Updated Edition

It contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Enhancements

These enhancements are all included in JZOS Version 2.4.2. Those products are IBM 31-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R31) and IBM 64-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R32).

- JZOS record mode data set performance has been significantly enhanced in release 2.4.2. See "MVS Data Set I/O" on page 17, for details.
- ZFile.obtainDSN() is updated to return null if the specified dsn does not have a DSCB in the VTOC. (In releases earlier than JZOS 2.4.0, the same method returns a pseudo DSCB.)

SA23–2245–03, Updated Edition

It contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Enhancements

These enhancements are all included in JZOS Version 2.4.0, which is part of the z/OS Java SDK 6.0.1 products. Those products are IBM 31-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R31) and IBM 64-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R32).

- Changes to ZFile to support proper ENQUEUE semantics
- The class com.ibm.jzos.MvsJobSubmitter was added to enable direct job submission from Java
- Added new package com.ibm.jzos.wlm with access to selected Workload Manager (WLM) APIs
- The com.ibm.jzos.ZLogstream class was enhanced to support IXGBRWSE (read) and IXGDELT (delete). In addition, InputStream/OutputStream wrappers were added
- Added the method ZFile.readDSCBChain() to support reading all of the DSCBs associated with a dsn / volume. This method also supports extended access volumes (Format-8 and Format-9 DSCBs)
- Changed the launcher to set the system property jzos.launcher=true. This property can be queried by a java class to determine if the JVM was started by the JZOS launcher
- Updated the ZFileException class to produce better informational messages
- Removed all JRIO code dependencies
- The PackedDecimalAsIntField, PackedDecimalAsLongField, PackedDecimalAsBigIntegerField, and PackedDecimalAsBigDecimalField classes in the com.ibm.jzos.fields package have been changed so "get" methods that encounter an invalid sign will throw an IllegalArgumentException.
- Some JZOS methods have been enhanced so that invalid arguments will cause IllegalArgumentExceptions.

SA23–2245–02, Updated Edition

It contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Enhancements

Many of the examples have been updated to SDK 6.0 product samples.

SA23–2245–01, Updated Edition

It contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Enhancements

These enhancements are all included in JZOS Version 2.3.0, which is part of the z/OS Java SDK 5.0 and SDK 6.0 products.

- 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)
- Use system property to the batch launcher to combine System.out and System.err to DD:SYSOUT
- Invoke z/OS clock functions STCK and STCKE
- Invoke ZFileRecordReader, ZFileFixedRecordReader, ZFileVariableRecordReader, ZFileVariableSpannedRecordReader for java deblocking of record mode files

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 z/OS Java SDK 1.4.2 (31 bit), z/OS Java SDK 5.0, SDK 6.0.0, (31 bit and 64 bit) and SDK 6.0.1, (31 bit and 64 bit).
- Simple yet flexible way to configure the Java execution environment.
- Access to data sets via JCL DD statements.
- Send output directly to JES SYSOUT data sets 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 data sets 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 2 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 JZOS batch launcher function is delivered as part of the z/OS Java product. 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 SDK for z/OS, Java 2 Technology Edition, V1.4 (5655-I56)	JVMLDM14	JVMPRC14	JVMJCL14
IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N98)	JVMLDM50	JVMPRC50	JVMJCL50
IBM 64-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N99)	JVMLDM56	JVMPRC56	JVMJCL56
IBM 31-bit SDK for z/OS, Java Technology Edition, V6.0.0 (5655-R31)	JVMLDM60	JVMPRC60	JVMJCL60
IBM 64-bit SDK for z/OS, Java Technology Edition, V6.0.0 (5655-R32)	JVMLDM66	JVMPRC66	JVMJCL66
IBM 31-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R31)	JVMLDM61	JVMPRC61	JVMJCL61
IBM 64-bit SDK for z/OS, Java Technology Edition, V6.0.1 (5655-R32)	JVMLDM67	JVMPRC67	JVMJCL67

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 V6.0.1 product, but are similar for the other z/OS SDK products. See “Appendix D. Non-SMP/E User Installation Instructions for SDK5 Users” on page 43 and “Appendix E. Non-SMP/E User Installation Instructions for SDK6 Users” on page 45, for specifics.

   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. For LOADLIB allocation recommendations refer to “Appendix C. SMP/E Install ++HOLD Information” on page 35, item "Define new data set using the following attributes".)
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 V6.0.1 SDK and using the default target libraries, issue the following commands under a USS shell:

   ```
   cd <JAVA_HOME>/mvstools
   cp samples/jcl/JVMJCL61 "//'SYS1.SAMPLIB(JVMJCL61)'"
   cp samples/jcl/JVMPRC61 "//'SYS1.PROCLIB(JVMPRC61)'"
   cp -X JVMLDM61 "//'SYS1.SIEALNKE(JVMLDM61)'"
   ```
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: 2.4.0 2010-11-16
   JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
   java version "1.6.0"
   Java(TM) SE Runtime Environment (build 20110126_73852)
   IBM J9 VM (build 2.6, JRE 1.6.0 z/OS s390-31 20110126_73852 (JIT enabled, AOT enabled)
   J9VM - R26_head_20110125_1328_B73788
   JIT - r11_20110125_18423
   GC - R26_head_20110124_1626_B73689
   J9CL - 20110126_73852)
   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 JVMLDM61,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

See “Appendix C. SMP/E Install ++HOLD Information” on page 35 for SDK1.4.2 and SDK5 users. The SMP/E installation of the ordered SDK6 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 SDK 6.0.1 product.

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//*     If the PDSE containing the JVMLDM61 module is not in your
//*     LNKLST, uncomment the STEPLIB statement and update the DSN to
//*     refer to the PDSE.
//*
//*********************************************************************
//JVMPRC61 PROC JAVACLS=,                < Fully Qfied Java class..RQD
//   ARGS=,                              < Args to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',       < STEPLIB FOR JVMLDM module
//   VERSION='61',                       < JVMLDM version: 61
//   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
```

See “Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products” on page 49 for SDK5 and SDK6 samples, which are very similar.

Sample JCL

The following sample is distributed in the 'mvstools/samples/jcl' directory. This sample JCL is for the 31-bit SDK 6.0.1 product.

```
//jobname JOB ...
//*********************************************************************
//*
//* 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=JVMPRC61,
// 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/J6.0.1

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 "
//
```

See “Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products” on page 49 for SDK5 and SDK6 samples, which are very similar.

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:

```
//JVMPRC61 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 13 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 13 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.

Table 4. JZOS Environment Variables (continued)

Environment Variable	Description
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.

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.
-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 13 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=JVMPRC60,
//  JAVACLS='com.ibm.jzos.sample.HelloWorld'
//STDENV DD *
APP_HOME=/usr/local/apps/myapp
. /etc/jzos_config.sh
//
```

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 13 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 13 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 13 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 17 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 20 and "Handling MVS START and MODIFY Commands" on page 20.

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 data sets 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 data sets, specify the encoding as `Zutil.getDefaultPlatformEncoding()`. The JZOS Toolkit portable file IO classes are already implemented to use this platform encoding for MVS data sets. See "Platform Independent Text File I/O with FileFactory" on page 18 for more information.

Messages and Return Codes

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. An error occurred in the launcher (see below).
4. 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 launcher itself fails, SYSOUT will contain the message:

  ```
  JVMJZBL1021E JZOS batch launcher failed, return code=nnn
  ```

  where nnn is one of the codes described in the table below.
- 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.

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=JVMPRC60,
//  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.

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/J6.0.1
  $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:

- 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 data sets.
- Methods for allocating, deleting, and renaming MVS data sets.
- 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 data set 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) 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
```

Note: another method, `MvsConsole.wto(String[]`, can be used to allow the caller to control where line breaks occur in multi-line WTOs.

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 alphaWorks, includes an example Eclipse project and guided instructions on using Eclipse as an IDE with z/OS Java: http://www.alphaworks.ibm.com/tech/zosjavabatchtk?
4. Beginning in z/OS V1R9, ISPF now supports ASCII editing of HFS or zFS files.

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.

Programmer response: No programmer response required.

System action: No system action is taken.

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

Explanation: Copyright notice issued with launcher is started.

Programmer response: No programmer response required.

System action: No system action is taken.

1003E JZOS_MSG_NO_STDENV "No //DD:STDENV found, using existing environment.".

Explanation: Issued when no STDENV is supplied by the user.

Programmer response: No programmer response required.

System action: The launcher terminates.

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.

Programmer response: No programmer response required.

System action: The launcher terminates.

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

Explanation: Debugging information from adoption of child environment.

Programmer response: No programmer response required.

System action: No system action is taken.

1006I JZOS_MSG_ENV_VAR "%s = %s".

Explanation: Informational message for environment variable and associated value.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: Correct the invalid environment variable if possible.

System action: The launcher terminates.

1008I JZOS_MSG_ENV_STOP_STRING "%s".

Explanation: Debugging info of environment stop string eyecatcher.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

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

System action: The launcher terminates.

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.

Programmer response: No programmer response required.

System action: The launcher terminates.

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

Explanation: Issued when JVM creation fails.

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

System action: The launcher terminates.

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

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

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: No programmer response required.

System action: No system action is taken.

1015N JZOS_MSG_MVS_COMMANDS_DISABLED "MVS commands are DISABLED".

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

Programmer response: No programmer response required.

System action: No system action is taken.

1016I JZOS_MSG_MVS_COMMANDS_ENABLED "MVS commands are ENABLED".

Explanation: The launcher has established an operator console waiter.

Programmer response: No programmer response required.

System action: No system action is taken.

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

Explanation: The JNI_GetDefaultJavaVMInitArgs() failed.

Programmer response: No programmer response required.

System action: The launcher terminates.

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

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

Programmer response: Set the CLASSPATH environment variable.

System action: The launcher terminates.

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

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

Programmer response: Ensure enough storage is available.

System action: The launcher terminates.

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

Explanation: The specified java class could not be found.

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

System action: The launcher terminates.

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.

Programmer response: None.

System action: The launcher completes.

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

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

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

System action: The launcher terminates.

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

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

Programmer response: No programmer response required.

System action: No system action is taken.

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

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

Programmer response: No programmer response required.

System action: No system action is taken.

1025I JZOS_MSG_TRANSCODING_DISABLED "Automatic output transcoding is DISABLED".

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

Programmer response: No programmer response required.

System action: No system action is taken.

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

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

Programmer response: Select a supported encoding.

System action: No system action is taken.

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

Explanation: Issued to describe the output encoding in effect.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: No programmer response required.

System action: No system action is taken.

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.

Programmer response: Decrease the main argument length.

System action: The launcher terminates.

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.

Programmer response: Specify a DDNAME that exists.

System action: The launcher terminates.

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.

Programmer response: Fix the arguments.

System action: The launcher terminates.

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.

Programmer response: Specify a Java class.

System action: The launcher terminates.

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

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

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

System action: The launcher terminates.

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

Explanation: Could not spawn child environment process shell.

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

System action: The launcher terminates.

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

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

Programmer response: No programmer response required.

System action: No system action is taken.

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

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

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

System action: The launcher terminates.

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.

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

System action: The launcher terminates.

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

Explanation: The child environment process received an unexpected signal.

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

System action: The launcher terminates.

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

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

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

System action: The launcher terminates.

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.

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

System action: The launcher terminates.

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

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

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

System action: The launcher terminates.

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).

Programmer response: None.

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

1044E JZOS_EXIT_JVM_ABORTED "The Java virtual machine aborted".

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

Programmer response: None.

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

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.

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

System action: The launcher terminates.

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.

Programmer response: None.

System action: The launcher terminates.

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.

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

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

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.

Programmer response: None.

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

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.

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

System action: None.

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

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

Programmer response: None.

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

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

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

Programmer response: None.

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

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.

Programmer response: None.

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

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

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

Programmer response: None.

System action: 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.

Programmer response: Correct the invalid environment variable if possible.

System action: None.

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.

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

System action: The launcher terminates.

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.

Programmer response: Ensure enough storage is available.

System action: The launcher terminates.

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.

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

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

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

Explanation: Prints out the NLS environment variables.

Programmer response: None.

System action: 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.

Programmer response: None.

System action: 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.

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

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

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

Explanation: An exception occurred in a native method.

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

System action: None.

2007E JZOS_CMN_STACKTRACE_HEADER "Stack trace follows:".

Explanation: An exception occurred in a native method.

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

System action: None.

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.

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

System action: None.

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.

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

System action: None.

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

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

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

System action: None.

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.

Programmer response: None.

System action: None.

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

Versions of JZOS are also available on IBM alphaworks 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.

Appendix C. SMP/E Install ++HOLD Information

The following section may be of interest to the z/OS Java SDK customers who install via SMP/E.

++HOLD Information Delivered with the SDK 1.4.2 SR6 PTF

The ++HOLD information delivered with the SDK 1.4.2 SR6 PTF is reproduced below:

++HOLD DDDEF

Special Conditions:

This PTF includes JZOS batch launcher function. 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.

Customers using the defaults and installing into the z/OS SMP/E zone on z/OS V1.6 or above have to do nothing further for this ++HOLD.

Customers running on z/OS V1.4 or z/OS V1.5 or customers installing into a zone other than the z/OS SMP/E zone must perform the steps described below.

```
    DDDEF:
COMMENT(**************************************************************
        *                                                            *
        *  FUNCTION AFFECTED - JAVA SDK 1.4.2                        *
        *                      JZOS batch launcher function          *
        *                                                            *
        **************************************************************
        *                                                            *
        *  DESCRIPTION - Allocate new target/distribution library    *
        *                (SIEALNKE/AIEALNKE)                         *
        *                                                            *
        **************************************************************
        *                                                            *
        *  TIMING    -   Pre-Apply                                   *
        *                                                            *
        **************************************************************
```

In order to install this PTF, the following steps should be followed:

1. Define new DDDEF using the following UCLIN

   You could use your own system procedures or run a job using the sample below as a template. Fields that need filling in are csi name (#globalcsi), target zone (#tzone), volume (ttttt1 and dddddd). If you use a DATASET name other than "SYS1.SIEALNKE" or "ASYS1.SIEALNKE", then those names should be modified also. (There is text in the file AJVDDDEF provided with the this z/OS Java SDK product which describes the parameters in detail if additional information is desired.)

   Note: If you are installing into a zone other than the MVS zone, you may need to define a new DDDEF for PROCLIB, SAMPLIB, APROCLIB and ASAMPLIB also.

   ```
   //DDDEFN1 EXEC PGM=GIMSMP,REGION=4096K
     //SMPCSI  DD   DSN=#globalcsi,
     //             DISP=SHR
     //SMPCNTL DD *
   ```

```
      SET   BDY(#tzone) .
       UCLIN.
          ADD DDDEF (SIEALNKE)
             DATASET(SYS1.SIEALNKE)
            UNIT(SYSALLDA)
             VOLUME(ttttt1)
             WAITFORDSN
                SHR.
        ADD DDDEF (AIEALNKE)
              DATASET(SYS1.AIEALNKE)
              UNIT(SYSALLDA)
               VOLUME(dddddd)
               WAITFORDSN
              SHR.
       ENDUCL.
  /*
  //
```

2. Define new data set using the following attributes

   Here is an example data set with the correct parameters for SYS1.SIEALNKE and SYS1.AIEALNKE. Fields that need filling in are volume (ttttt2 and dddddd). If you use a DATASET name other than "SYS1.SIEALNKE" or "ASYS1.SIEALNKE", then those names should be modified also.

```
//*  1) Add the job parameters to meet your system requirements      *
//*                                                                  *
//*  2) Change DSP and DDSP variable to the appropriate final        *
//*     disposition of the data set ("CATLG" is the default) on the  *
//*     exec steps if you choose not to use the default.             *
//*                                                                  *
//*  3) This job uses the recommended data set placement for         *
//*     the target libraries:                                        *
//*     Change ttttt1 to the volser for first volume for the         *
//*            target libraries (tvol1).                             *
//*                                                                  *
//*  4) Change dddddd to the volser for the distribution             *
//*     libraries.                                                   *
//*                                                                  *
//*  If you specify a volume for any data set in this job, you       *
//*  must also specify the same volume in the corresponding          *
//*  DDDEF entry in the DDDEF job, AJVDDDEF.                         *
//*  5) This PTF contains PDSE data set(s). You must verify your     *
//*     ACS routines to ensure that the data set will be             *
//*     allocated where appropriate for your environment. The        *
//*     default allocation of the PDSE(s) in this job is             *
//*     non-SMS managed.  If you want a non-SMS managed PDSE         *
//*     ensure that your ACS routines will not override the          *
//*     PDSE allocation parameters and create an SMS managed         *
//*     data set. If you want SMS managed PDSE(s), then verify       *
//*     that the allocation will result in SMS managed data sets.    *
//*                                                                  *
//*  Notes:                                                          *
//*                                                                  *
//*  1. This job should complete with a return code 0. You must      *
//*     check allocation messages to verify the data sets are        *
//*     allocated and cataloged as expected.                         *
//*                                                                  *
//*  2. If you are installing into a into a zone other than the MVS  *
//*     zone, you may need to allocate PROCLIB, SAMPLIB, APROCLIB    *
//*     and ASAMPLIB also.                                           *
//*                                                                  *
//********************************************************************
//**********************************************************
//*   Allocate Target Libraries                            *
//**********************************************************
//AJVALLOC  PROC  HLQ=,DHLQ=,
//            DSP=,DDSP=,
//            TVOL1=,DVOL1=
//*
//ALLOC1  EXEC  PGM=IEFBR14
//*
//SIEALNKE DD  DSN=&HLQ..SIEALNKE,
//         DISP=(NEW,&DSP),
//         RECFM=U,
//         LRECL=0,
//         BLKSIZE=32760,
//         DSNTYPE=LIBRARY,
//         UNIT=SYSALLDA,
//         VOL=SER=&TVOL1,
//         SPACE=(8800,(10300,130,210))
//*
//**********************************************************
//*   Allocate Distribution Libraries                      *
//**********************************************************
//*
//AIEALNKE DD  DSN=&DHLQ..AIEALNKE,
//         DISP=(NEW,&DDSP),
//         RECFM=U,
//         LRECL=0,
//         BLKSIZE=32760,
//         DSNTYPE=LIBRARY,
//         UNIT=SYSALLDA,
//         VOL=SER=&DVOL1,
//         SPACE=(8800,(10300,130,210))
//*
//*
//EALLOC      PEND
//*
//***************************************************************
//*  The following steps execute the PROCs to allocate the      *
//*  datasets from prior releases.  Remove these steps if       *
//*  the prior release datasets already exist with proper space. *
//***************************************************************
//*
//ALLOCT EXEC AJVALLOC,
//            HLQ=SYS1,     * SYS1 is the default for the target library
//            DHLQ=SYS1,    * SYS1 is the default for the dist. library
//            DSP=CATLG,    * CATLG is the default for the target library
//            DDSP=CATLG,   * CATLG is the default for the dist. library
//            TVOL1=ttttt1, * no default; volume for target library
//            DVOL1=dddddd  * no default; volume for dist. library
```

Also note that the following should be a ++HOLD(ACTION)

If you allocate a separate SIEALNKE library on z/OS R4 or R5, define the DDDEF and install the PTF, the library should be added to the LNKLST. More information about the LNKLST can be found in the *z/OS MVS Initialization and Tuning Guide*, SA22-7591.

After installation, see more jZOS documentation to learn about the JZOS functionality at www.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html.

++HOLD Information Delivered with the SDK 5 SR3 PTF

The ++HOLD information delivered with the SDK 5 SR3 PTF is reproduced below:

++HOLD DDDEF

Special Conditions:

This PTF includes JZOS batch launcher function. 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.

Customers using the defaults and installing into the z/OS SMP/E zone on z/OS V1.6 or above have to do nothing further for this ++HOLD.

Customers installing into a zone other than the z/OS SMP/E zone must perform the steps described below.

```
*****************************************************************
*
*  FUNCTION AFFECTED - JAVA SDK 5
*                      JZOS batch launcher function
*
*****************************************************************
*
*  DESCRIPTION - Allocate new target/distribution libraries
*  (SIEALNKE/AIEALNKE), (PROCLIB,APROCLIB), and
*  (SAMPLIB,ASAMPLIB)
*
*****************************************************************
*
*  TIMING    -   Pre-Apply
*
*****************************************************************
```

In order to install this PTF, the following steps should be followed:

1. Define new DDDEF using the following UCLIN

   You could use your own system procedures or run a job using the sample below as a template. Fields that need filling in are csi name (#globalcsi), target zone (#tzone), high level qualifer (hlq), and volume (ttttt1, ttttt2 and dddddd).

   Note: If you are installing into a zone other than the MVS zone, you may need to define a new DDDEF for PROCLIB, SAMPLIB, APROCLIB and ASAMPLIB also.

```
//DDDEFN1 EXEC PGM=GIMSMP,REGION=4096K
//SMPCSI  DD   DSN=#globalcsi,
//             DISP=SHR
//SMPCNTL DD *
 SET   BDY(#tzone) .
     UCLIN.
      ADD DDDEF (SIEALNKE)
          DATASET(hlq.SIEALNKE)
          UNIT(SYSALLDA)
          VOLUME(ttttt1)
          WAITFORDSN
          SHR.
      ADD DDDEF (AIEALNKE)
          DATASET(hlq.AIEALNKE)
          UNIT(SYSALLDA)
          VOLUME(dddddd)
          WAITFORDSN
```

```
        SHR.
    ADD DDDEF (PROCLIB)
        DATASET(hlq.PROCLIB)
        UNIT(SYSALLDA)
        VOLUME(ttttt1)
        WAITFORDSN
        SHR.
    ADD DDDEF (APROCLIB)
        DATASET(hlq.APROCLIB)
        UNIT(SYSALLDA)
        VOLUME(dddddd)
        WAITFORDSN
        SHR.
    ADD DDDEF (SAMPLIB)
        DATASET(hlq.SAMPLIB)
        UNIT(SYSALLDA)
        VOLUME(ttttt2)
        WAITFORDSN
        SHR.
    ADD DDDEF (ASAMPLIB)
        DATASET(hlq.ASAMPLIB)
        UNIT(SYSALLDA)
        VOLUME(dddddd)
        WAITFORDSN
        SHR.
   ENDUCL.
/*
//
```

2. Define new data set using the following attributes

   Here is an example data set with the correct parameters for hlq.SIEALNKE and hlq.AIEALNKE. Fields that need filling in are volume (ttttt1 and dddddd).

```
//*****************************************************************
//*  1. Add the job parameters to meet your system
//*     requirements.
//*
//*  2. Change DSP and DDSP variable to the appropriate final
//*     disposition of the data set ("CATLG" is the default) on
//*     the exec steps if you choose not to use the default.
//*
//*  3. This job uses the recommended data set placement for
//*     the target libraries:
//*     Change ttttt1 to the volser for first volume for the
//*            target libraries (tvol1).
//*     Change ttttt2 to the volser for second volume for the
//*            target libraries (tvol2).
//*
//*  4. Change dddddd to the volser for the distribution
//*     libraries.
//*
//*  If you specify a volume for any data set in this job, you
//*  must also specify the same volume in the corresponding
//*  DDDEF entry in the DDDEF job, AJVDDDEF.
//*
//*  5. This PTF contains PDSE data set(s). You must verify
//*     your ACS routines to ensure that the data set will be
//*     allocated where appropriate for your environment. The
//*     default allocation of the PDSE(s) in this job is
//*     non-SMS managed.  If you want a non-SMS managed PDSE
//*     ensure that your ACS routines will not override the
//*     PDSE allocation parameters and create an SMS managed
//*     data set. If you want SMS managed PDSE(s), then verify
//*     that the allocation will result in SMS managed
//*     data sets.
//*
//*  Notes:
//*
```

```
//*  1. This job should complete with a return code 0. You must
//*     check allocation messages to verify the data sets are
//*     allocated and cataloged as expected.
//*
//*  2. If you are installing into a into a zone other than the
//*     MVS zone, you may need to allocate PROCLIB, SAMPLIB,
//*     APROCLIB and ASAMPLIB also.
//*
//**************************************************************
//*   Allocate Target Libraries
//**************************************************************
//AJVALLOC  PROC  HLQ=,DHLQ=,
//            DSP=,DDSP=,
//            TVOL1=,DVOL1=
//*
//ALLOC1  EXEC  PGM=IEFBR14
//*
//SIEALNKE DD  DSN=&HLG..SIEALNKE,
//         DISP=(NEW,&DSP),
//         RECFM=U,
//         LRECL=0,
//         BLKSIZE=32760,
//         DSNTYPE=LIBRARY,
//         UNIT=SYSALLDA,
//         VOL=SER=&TVOL1,
//         SPACE=(8800,(5000,50,100))
//*
//PROCLIB DD  DSN=&HLQ..PROCLIB,
//         DISP=(NEW,&DSP),
//         RECFM=FB,
//         LRECL=80,
//         BLKSIZE=27920,
//         UNIT=SYSALLDA,
//         VOL=SER=&TVOL1,
//         SPACE=(8800,(40,4,2))
//*
//SAMPLIB DD  DSN=&HLQ..SAMPLIB,
//         DISP=(NEW,&DSP),
//         RECFM=FB,
//         LRECL=80,
//         BLKSIZE=27920,
//         UNIT=SYSALLDA,
//         VOL=SER=&TVOL2,
//         SPACE=(8800,(20,2,2))
//*
//**************************************************************
//*   Allocate Distribution Libraries
//**************************************************************
//*
//AIEALNKE DD  DSN=&DHLQ..AIEALNKE,
//         DISP=(NEW,&DDSP),
//         RECFM=U,
//         LRECL=0,
//         BLKSIZE=32760,
//         DSNTYPE=LIBRARY,
//         UNIT=SYSALLDA,
//         VOL=SER=&DVOL1,
//         SPACE=(8800,(5000,50,100))
//*
//APROCLIB DD  DSN=&HLQ..APROCLIB,
//         DISP=(NEW,&DSP),
//         RECFM=FB,
//         LRECL=80,
//         BLKSIZE=27920,
//         UNIT=SYSALLDA,
//         VOL=SER=&DVOL1,
//         SPACE=(8800,(40,4,2))
```

```
//*
//ASAMPLIB DD  DSN=&HLQ..ASAMPLIB,
//         DISP=(NEW,&DSP),
//         RECFM=FB,
//         LRECL=80,
//         BLKSIZE=27920,
//         UNIT=SYSALLDA,
//         VOL=SER=&DVOL1,
//         SPACE=(8800,(20,2,2))
//*
//*
//*
//EALLOC      PEND
//*
//****************************************************************
//*  The following steps execute the PROCs to allocate the
//*  datasets from prior releases.  Remove these steps if
//*  the prior release datasets already exist with proper
//*  space.
//****************************************************************
//*
//ALLOCT EXEC AJVALLOC,
//            HLQ=hlq,      * HLQ for the target library
//            DHLQ=hlq,     * HLQ for the dist. library
//            DSP=CATLG,    * default for the target library
//            DDSP=CATLG,   * default for the dist. library
//            TVOL1=ttttt1, * volume for target library
//            TVOL2=ttttt2, * volume for target library
//            DVOL1=dddddd  * volume for dist. library
```

If you allocate a separate SIEALNKE library or install into the z/OS zone on z/OS R6 or R7, and want to use JZOS, the library should be added to the LNKLST if not already there. More information about the LNKLST can be found in the *z/OS MVS Initialization and Tuning Guide*, SA22-7591.

After installation, see more jZOS documentation to learn about the JZOS functionality at www.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html.

Appendix D. Non-SMP/E User Installation Instructions for SDK5 Users

The instructions are very similar to those for current users of JZOS, including those using the IBM AlphaWorks version.

For the 31-bit SDK5 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions.

   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. For LOADLIB allocation recommendations refer to "Appendix C. SMP/E Install ++HOLD Information" on page 35, item "Define new data set using the following attributes". **Note:** This information can be sent if required.)
3. Copy loadmod to a PDSE, PROC, and JCL to a PDS from the unpaxed file. 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 5.0 SDK and using the default target libraries, issue the following commands under a USS shell:
   ```
   cp JVMJCL50 "//'SYS1.SAMPLIB(JVMJCL50)'"
   cp JVMPRC50 "//'SYS1.PROCLIB(JVMPRC50)'"
   cp -X JVMLDM50 "//'SYS1.SIEALNKE(JVMLDM50)'"
   ```
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 and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:
   ```
   JVMJZBL1001N JZOS batch Launcher Version: 2.0.0 2006-06-30
   JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
   java version "1.5.0"
   Java(TM) 2 Runtime Environment, Standard Edition (build pmz31devifx-20060524 (SR3))
   IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-20060505a
     (JIT enabled)
   J9VM - 20060501_06428_bHdSMr
   JIT  - 20060428_1800_r8
   GC   - 20060501_AA)
   JCL  - 20060524a
   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 JVMLDM50,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.

For the 64-bit SDK5 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions.

   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. For LOADLIB allocation recommendations refer to “Appendix C. SMP/E Install ++HOLD Information” on page 35, item "Define new data set using the following attributes". **Note:** This information can be sent if required.)
3. Copy loadmod to a PDSE, PROC, and JCL to a PDS from the unpaxed file. 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 5.0 SDK and using the default target libraries, issue the following commands under a USS shell:
   ```
   cp JVMJCL56 "//'SYS1.SAMPLIB(JVMJCL56)'"
   cp JVMPRC56 "//'SYS1.PROCLIB(JVMPRC56)'"
   cp -X JVMLDM56 "//'SYS1.SIEALNKE(JVMLDM56)'"
   ```
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 and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:
   ```
   JVMJZBL1001N JZOS batch Launcher Version: 2.0.0 2006-06-30
   JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
   java version "1.5.0"
   Java(TM) 2 Runtime Environment, Standard Edition (build pmz64devifx-20060524 (SR3))
   IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390x-64 j9vmmz6423-20060505a
     (JIT enabled)
   J9VM - 20060501_06428_bHdSMr
   JIT  - 20060428_1800_r8
   GC   - 20060501_AA)
   JCL  - 20060524a
   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 JVMLDM56,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.

Appendix E. Non-SMP/E User Installation Instructions for SDK6 Users

The instructions are very similar to those for current users of JZOS, including those using the IBM AlphaWorks version.

For the 31-bit SDK6.0.0 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions.

   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. For LOADLIB allocation recommendations refer to “Appendix C. SMP/E Install ++HOLD Information” on page 35, item "Define new data set using the following attributes". **Note:** This information can be sent if required.)
3. Copy loadmod to a PDSE. Copy PROC and JCL to a PDS from the unpaxed file. 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 6.0.0 SDK and using the default target libraries, issue the following commands under a USS shell:
   ```
   cp JVMJCL60 "//'SYS1.SAMPLIB(JVMJCL60)'"
   cp JVMPRC60 "//'SYS1.PROCLIB(JVMPRC60)'"
   cp -X JVMLDM60 "//'SYS1.SIEALNKE(JVMLDM60)'"
   ```
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 and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:
   ```
   JVMJZBL1001N JZOS batch Launcher Version: 2.0.0 2007-02-12
   JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
   java version "1.6.0"
   JVMJZBL1012I Java Virtual Machine created. Version information follows:
   java version "1.6.0"
   Java(TM) SE Runtime Environment (build 2.4)
   IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 z/OS s390-31 jvmmz316020071121_15015
     (JIT enabled)
   J9VM - 20071121_015015_bHdSMr
   JIT  - r9_20071121_1300
   GC   - 20071031_AA)

   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 JVMLDM60,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.

For the 64-bit SDK6.0.0 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions.

   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. For LOADLIB allocation recommendations refer to “Appendix C. SMP/E Install ++HOLD Information” on page 35, item "Define new data set using the following attributes". **Note:** This information can be sent if required.)
3. Copy loadmod to a PDSE. Copy PROC and JCL to a PDS from the unpaxed file. 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 6.0.0 SDK and using the default target libraries, issue the following commands under a USS shell:
   ```
   cp JVMJCL66 "//'SYS1.SAMPLIB(JVMJCL66)'"
   cp JVMPRC66 "//'SYS1.PROCLIB(JVMPRC66)'"
   cp -X JVMLDM66 "//'SYS1.SIEALNKE(JVMLDM66)'"
   ```
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 and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:
   ```
   JVMJZBL1001N JZOS batch Launcher Version: 2.0.0 2007-02-12
   JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
   java version "1.6.0"
   Java(TM) SE Runtime Environment (build 2.4)
   IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 z/OS s390x-64 jvmmz6460-20071121_15015
     (JIT enabled)
   J9VM - 20071121_015015_bHdSMr
   JIT  - r9_20071121_1300
   GC   - 20071031_AA)

   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 JVMLDM66,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.

For the 64-bit SDK6.0.1 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unpaxed per the Java SDK product install instructions.

   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. For LOADLIB allocation recommendations refer to “Appendix C. SMP/E Install ++HOLD Information” on page 35, item "Define new data set using the following attributes". **Note:** This information can be sent if required.)
3. Copy loadmod to a PDSE. Copy PROC and JCL to a PDS from the unpaxed file. 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 6.0.1 SDK and using the default target libraries, issue the following commands under a USS shell:
```
cp JVMJCL67 "//'SYS1.SAMPLIB(JVMJCL67)'"
cp JVMPRC67 "//'SYS1.PROCLIB(JVMPRC67)'"
cp -X JVMLDM67 "//'SYS1.SIEALNKE(JVMLDM67)'"
```
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 and check the job log. If everything was set up properly, the SYSOUT DD should contain output like this:
```
JVMJZBL1001N JZOS batch Launcher Version: 2.4.0 2010-11-16
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.6.0"
Java(TM) SE Runtime Environment (build 20110122_73551)
IBM J9 VM (build 2.6, JRE 1.6.0 z/OS s390x-64 20110122_73551 (JIT enabled, AOT enabled)
J9VM - R26_head_20110122_0937_B73530
JIT - r11_20110121_18381
GC - R26_head_20110121_1603_B73484
J9CL - 20110122_73551)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 JVMLDM67,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.

Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products

These samples are for the SDK 5 products. The SDK 6 samples would be nearly identical. See the table in “Introduction to JZOS Batch Launcher Installation” on page 3, for more information on naming conventions.

Sample PROC

This sample PROC is for the 31-bit SDK5 product.

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//* 1.) Replace '<HLQ>.JZOS.LOADLIB' with the PDSE that contains the
//*     JVMLDM50 module that was installed during installation
//* 2.) The STEPLIB is commented out. Unless you are doing an SMP/E
//*     install into the MVS zone on z/OS 1.6 or above, you should
//*     UNCOMMENT the STEPLIB and point to the STEPLIB being used
//*
//*********************************************************************
//JVMPRC50 PROC JAVACLS=,               < Fully Qfied Java class..RQD
//   ARGS=,                            < Args to Java class
//   LIBRARY='<HLQ>.JZOS.LOADLIB',     < STEPLIB FOR JVMLDM module
//   VERSION='50',                     < JVMLDM version: 50
//   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
```

This sample PROC is for the 64-bit SDK5 product.

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//* 1.) Replace '<HLQ>.JZOS.LOADLIB' with the PDSE that contains the
//*     JVMLDM56 module that was installed during installation
//* 2.) The STEPLIB is commented out. Unless you are doing an SMP/E
//*     install into the MVS zone on z/OS 1.6 or above, you should
//*     UNCOMMENT the STEPLIB and point to the STEPLIB being used
//*
//*********************************************************************
//JVMPRC56 PROC JAVACLS=,               < Fully Qfied Java class..RQD
//   ARGS=,                             < Args to Java class
//   LIBRARY='<HLQ>.JZOS.LOADLIB',      < STEPLIB FOR JVMLDM module
//   VERSION='56',                      < JVMLDM version: 56
//   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
```

This sample PROC is for the 31-bit SDK6.0.0 product.

```
//*******************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//* 1.) Replace '<HLQ>.JZOS.LOADLIB' with the PDSE that contains the
//*     JVMLDM60 module that was installed during installation
//* 2.) The STEPLIB is commented out. Unless you are doing an SMP/E
//*     install into the MVS zone on z/OS 1.6 or above, you should
//*     UNCOMMENT the STEPLIB and point to the STEPLIB being used
//*
//*******************************************************************
//JVMPRC60 PROC JAVACLS=,               < Fully Qfied Java class..RQD
//   ARGS=,                             < Args to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',      < STEPLIB FOR JVMLDM module
//   VERSION='60',                      < JVMLDM version: 60
//   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
```

This sample PROC is for the 64-bit SDK6.0.0 product.

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//* 1.) Replace '<HLQ>.JZOS.LOADLIB' with the PDSE that contains the
//*     JVMLDM66 module that was installed during installation
//* 2.) The STEPLIB is commented out. Unless you are doing an SMP/E
//*     install into the MVS zone on z/OS 1.6 or above, you should
//*     UNCOMMENT the STEPLIB and point to the STEPLIB being used
//*
//*********************************************************************
//JVMPRC66 PROC JAVACLS=,                < Fully Qfied Java class..RQD
//   ARGS=,                              < Args to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',       < STEPLIB FOR JVMLDM module
//   VERSION='66',                       < JVMLDM version: 66
//   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
```

This sample PROC is for the 31-bit SDK6.0.1 product.

```
//*******************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc your installation:
//*     If the PDSE containing the JVMLDM61 module is not in your
//*     LNKLST, uncomment the STEPLIB statement and update the DSN to
//*     refer to the PDSE.
//*
//*******************************************************************
//JVMPRC61 PROC JAVACLS=,                < Fully Qfied Java class..RQD
//   ARGS=,                              < Args to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',       < STEPLIB FOR JVMLDM module
//   VERSION='61',                       < JVMLDM version: 61
//   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
```

This sample PROC is for the 64-bit SDK6.0.1 product.

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the proc 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
//*
//********************************************************************
//JVMPRC67 PROC JAVACLS=,               < Fuly Qfied Java class..RQD
//   ARGS=,                             < Arg to Java class
//*  LIBRARY='<HLQ>.JZOS.LOADLIB',      < STPLIB FOR JVMLDM module
//   VERSION='67',                      < JVLDM version: 67
//   LOGLVL='',                         < Debg LVL: +I(info) +T(trc)
//   REGSIZE='0M',                      < XECUTION 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

This sample JCL is for the 31-bit SDK5 product.

```
********************************* Top of Data **************************
//**********************************************************************
//*
//* 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=JVMPRC50,
// 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/J5.0

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

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

# Customize your CLASSPATH here
APP_HOME=$JAVA_HOME
CLASSPATH=$APP_HOME

# 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 "

//
```

This sample JCL is for the 64-bit SDK5 product.

```
//*********************************************************************
//*
//* 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=JVMPRC56,
// 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/J5.0_64

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

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

# Customize your CLASSPATH here
APP_HOME=$JAVA_HOME
CLASSPATH=$APP_HOME

# 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 "

//
```

This sample JCL is for the 31-bit SDK6.0.0 product.

```
//jobname JOB ...
//*********************************************************************
//*
//* 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=JVMPRC60,
// 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/J6.0

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 "
//
```

This sample JCL is for the 64-bit SDK6.0.0 product.

```
//jobname JOB ...
//*******************************************************************
//*
//* 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=JVMPRC66,
// 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/J6.0_64

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 "
//
```

This sample JCL is for the 31-bit SDK6.0.1 product.

```
//jobname JOB ...
//*********************************************************************
//*
//* 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=JVMPRC61,
// 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/J6.0.1

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 "
//
```

This sample JCL is for the 64-bit SDK6.0.1 product.

```
//jobname JOB ...
//*******************************************************************
//*
//* 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=JVMPRC67,
// 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/J6.0.1_64

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

LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390x
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390x/j9vm
LIBPATH="$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 "
//
```

Notices

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

IBM may not offer the products, services, or features discussed in this document in all 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 grant you any license to these patents. You can send license inquiries, in writing, to:

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

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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.

Trademarks

Bibliography

See the following publications for additional information.

The JZOS Cookbook, available on alphaWorks: http://www.alphaworks.ibm.com/tech/zosjavabatchtk?

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

Index

Readers' Comments — We'd Like to Hear from You

JZOS Batch Launcher and Toolkit
function in IBM SDK for z/OS
Installation and User's Guide

Publication No. SA23-2245-05

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

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 state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send your comments via email to: mhvrcfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SA23-2245-05

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation, Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, New York
12601-5400

Fold and Tape **Please do not staple** Fold and Tape

SA23-2245-05

Cut or Fold
Along Line

Printed in USA

SA23-2245-05