

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



# Installation and User's Guide



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



# Installation and User's Guide

**Note:**

Before using this information and the product it supports, read the information in "Notices" on page 51.

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

This edition replaces SA23-2245-01.

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  
MHVRCFS, Mail Station P181  
2455 South Road Poughkeepsie, NY 12601-5400  
United States of America

FAX (United States & Canada): 1+845+432-9405  
FAX (Other Countries):  
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)  
Internet e-mail: mhvrcfs@us.ibm.com  
World Wide Web: <http://www.ibm.com/servers/eserver/zseries/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.

© Copyright International Business Machines Corporation 2006, 2009.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About This Book</b> . . . . .	v
Using LookAt to look up message explanations . . . . .	v
<b>Summary of Changes</b> . . . . .	vii
SA23–2245–02, Updated Edition . . . . .	vii
Enhancements . . . . .	vii
SA23–2245–01, Updated Edition . . . . .	vii
Enhancements . . . . .	vii
<b>Chapter 1. Overview</b> . . . . .	1
JZOS Batch Launcher and Toolkit Capabilities and Features . . . . .	1
What is in this Document? . . . . .	1
<b>Chapter 2. Installation</b> . . . . .	3
Introduction to JZOS Batch Launcher Installation . . . . .	3
Non-SMP/E Users . . . . .	3
SMP/E Users . . . . .	4
<b>Chapter 3. IBM JZOS Batch Launcher User’s Guide</b> . . . . .	5
Introduction . . . . .	5
Sample PROC . . . . .	5
Sample JCL . . . . .	6
Specifying the Java main class and its arguments . . . . .	7
Example: supplying arguments to a Java class . . . . .	7
Setting Batch Launcher Logging Levels . . . . .	8
Configuring Environment Variables . . . . .	8
Environment Variables . . . . .	8
System Environment Variables . . . . .	9
Java SDK Environment Variables . . . . .	9
JZOS Environment Variables . . . . .	9
Java SDK Options and System Properties . . . . .	10
Customizing a Reusable Configuration Script for Your Installation . . . . .	11
Files Used by the Batch Launcher . . . . .	11
MVS Console Interface . . . . .	12
Controlling Output Encoding . . . . .	12
Recommendations . . . . .	13
Messages and Return Codes . . . . .	13
Language Environment Runtime Options . . . . .	14
Troubleshooting . . . . .	15
<b>Chapter 4. Toolkit User’s Guide</b> . . . . .	17
Introduction . . . . .	17
MVS Dataset I/O Using the ZFile Classes . . . . .	17
Platform Independent Text File I/O. . . . .	18
Writing Messages to the System Console or Log . . . . .	19
Handling MVS START and MODIFY Commands . . . . .	19
JZOS Sample Programs . . . . .	19
Hints and Tips . . . . .	20
Editing ASCII Files under z/OS . . . . .	20
<b>Appendix A. Messages</b> . . . . .	21
Messages Issued by the JZOS Batch Launcher . . . . .	21
Messages Common/Shared by the JZOS Batch Launcher and Toolkit. . . . .	28

Non-Messages Strings Used in some Java Exceptions . . . . .	30
<b>Appendix B. Migration from alphaWorks IBM JZOS Batch Toolkit for z/OS</b>	
<b>SDKs</b> . . . . .	31
JZOS Batch Launcher Changes . . . . .	31
JZOS Toolkit Changes . . . . .	31
<b>Appendix C. SMP/E Install ++HOLD Information</b> . . . . .	33
++HOLD Information Delivered with the SDK 1.4.2 SR6 PTF . . . . .	33
++HOLD Information Delivered with the SDK 5 SR3 PTF . . . . .	36
<b>Appendix D. Non-SMP/E User Installation Instructions for SDK5 Users</b>	41
For the 31-bit SDK5 Product . . . . .	41
For the 64-bit SDK5 Product . . . . .	42
<b>Appendix E. Non-SMP/E User Installation Instructions for SDK6 Users</b>	43
For the 31-bit SDK6 Product . . . . .	43
For the 64-bit SDK6 Product . . . . .	44
<b>Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products</b> . . . . .	47
Sample PROC . . . . .	47
Sample JCL . . . . .	49
<b>Notices</b> . . . . .	51
Trademarks . . . . .	53
<b>Bibliography</b> . . . . .	55
<b>Index</b> . . . . .	57

---

## 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/servers/eserver/zseries/zos/bkserv/lookat/](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/).
- 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/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html](http://www.ibm.com/servers/eserver/zseries/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, SA23-2245-00, which supports JZOS Version 2.3.0.

---

### 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 and SDK 6.0, (31 bit and 64 bit).
- Simple yet flexible way to configure the Java execution environment.
- Access to datasets via JCL DD statements.
- Send output directly to JES SYSOUT datasets with automatic codepage transcoding.
- Pass condition codes between Java and non-Java job steps.
- Communicate with the MVS system console.
- Read and write traditional MVS datasets from Java.
- Java interfaces to many z/OS specific APIs and features including SMF, Catalog Search and Logstreams.
- Java classes to convert COBOL and Assembler data type fields to Java objects.
- Invoke DFSORT and direct either input and output to the Java application.
- Invoke z/OS Access Method Services (IDCAMS).
- Serialize z/OS resources (ISGENQ).
- Access z/OS system symbols (ASASYMB system symbol service).

---

### 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 also contains javadoc and the JZOS File I/O APIs.

For more details about the z/OS Java products and the JZOS function, see [www.ibm.com/servers/eserver/zseries/software/java](http://www.ibm.com/servers/eserver/zseries/software/java) and [www.ibm.com/servers/eserver/zseries/software/java/jzos](http://www.ibm.com/servers/eserver/zseries/software/java/jzos).



---

## Chapter 2. Installation

---

### 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	JVMPROC14	JVMJCL14
IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N98)	JVMLDM50	JVMPROC50	JVMJCL50
IBM 64-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N99)	JVMLDM56	JVMPROC56	JVMJCL56
IBM 31-bit SDK for z/OS, Java Technology Edition, V6 (5655-R31)	JVMLDM60	JVMPROC60	JVMJCL60
IBM 64-bit SDK for z/OS, Java Technology Edition, V6 (5655-R32)	JVMLDM66	JVMPROC66	JVMJCL66

---

### 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 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 41, 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 33, item “Define new data set using the following attributes”.)
  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 31-bit V6 SDK and using the default target libraries, issue the following commands under a USS shell:
- |
- |
- |
- |
- |

```

cd <JAVA_HOME>/mvstools
cp samples/jcl/JVMJCL60 "'/SYS1.SAMPLIB(JVMJCL60)'"
cp samples/jcl/JVMPROC60 "'/SYS1.PROCLIB(JVMPROC60)'"
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: 2.3.0 2008-05-12
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.6.0"
Java(TM) SE Runtime Environment (build pmz3160sr6-20090903_04(SR6))
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 z/OS s390-31 jvmmz3160sr6-20090902_41646 (JIT enabled, AOT enabled)
J9VM - 20090902_041646
JIT - r9_20090902_1330
GC - 20090817_AA)
JCL - 20090901_01
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 '+' :

```
// 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.

---

## SMP/E Users

See "Appendix C. SMP/E Install ++HOLD Information" on page 33.

---

## Chapter 3. IBM JZOS Batch Launcher User's Guide

---

### Introduction

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. See “Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products” on page 47 for SDK5 samples, which are very similar.

```

//*****
//*
//* 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
//  ARGVS=,                         < 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
//  LEPRM=''
//JAVAJVM EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//  PARM='&LEPRM/&LOGLVL &JAVACLS &ARGVS'
//* 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 is for the SDK 6.0 product. See “Appendix F. Sample PROC and JCL for the SDK5 and SDK6 Products” on page 47 for SDK5 samples, which are very similar.

```
//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} "
//
```

---

## 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

```
// EXEC PROC=JVMPCxx,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\nand 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 “JVMPCxx” 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.

---

## 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.
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 10 for more information.  See “JZOS Environment Variables” - JZOS_JVM_OPTIONS for more information.

---

## JZOS Environment Variables

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 12 for more information.

Table 4. JZOS Environment Variables (continued)

Environment Variable	Description
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 12 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 12 for more information.
JZOS_GENERATE_SYSTEM_EXIT = {true   false }	If set to true, JZOS will generate a System.exit() call upon completion of main(). This will cause JZOS to complete, even if there are active non-daemon threads. The default if not specified is 'false', which means JZOS will wait for non-daemon threads to complete before exiting.
JZOS_MAIN_ARGS = {classname and arguments} JZOS_MAIN_ARGS_DD = {ddname   MAINARGS }	Allows for additional arguments to specified to the main method that JZOS invokes. See "Specifying the Java main class and its arguments" on page 7 for more information.
JZOS_JVM_OPTIONS	Some Java SDK options and System properties are not recognized by the JVM when specified in the "IBM_JAVA_OPTIONS" environment variable. The JZOS_JVM_OPTIONS is an alternative space to specify these settings, and it used to specify options that are specified when JVM is created.

## 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 includes most options found on the standard 'java' command line launcher, including -X and -D options. The table below includes options commonly used with the batch launcher; refer to the complete list provided by 'java -help' and 'java -X' commands or to the IBM Java SDK Diagnostic Guides for more information.

Table 5. Java SDK Options and System Properties

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

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

Option	Description
-Dfile.encoding=<encoding>	Used to set the default file encoding. The default is usually IBM-1047, an EBCDIC encoding, but it is common to set this to ISO-8859-1, an ASCII encoding. See “Controlling Output Encoding” on page 12 for more information.
-Djzos.logging={E W I D}	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 distributed file `mvstools/sample/jzos_config.sh` is a sample shell script that you can customize 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 12 for more information.

Table 6. Files Used by the Batch Launcher (continued)

DD Name	Description
STDERR	( Output, Required )  The output from Java System.err. This data is translated to the JZOS_OUTPUT_ENCODING codepage. See “Controlling Output Encoding” 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” 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 Dataset I/O Using the ZFile Classes” 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 19 and “Handling MVS START and MODIFY Commands” on page 19.

---

## 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 10 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 as Apache Tomcat, include code that requires this.
- When running with an ASCII default `file.encoding`, applications must specifically use an EBCDIC encoding when using MVS datasets encoded in EBCDIC. The `Zutil.getDefaultPlatformEncoding()` method should be used to obtain the current “platform” encoding for this purpose.

---

## Recommendations

1. Avoid writing code that assumes a default `file.encoding`, but if you need to run code that does, run with `-Dfile.encoding=ISO-8859-1`. There is really no penalty for doing this, since internal Unicode must be translated to something anyway.
2. When accessing MVS datasets, specify the encoding as `Zutil.getDefaultPlatformEncoding()`. The JZOS Toolkit portable file IO classes are already implemented to use this platform encoding for MVS datasets. See “Platform Independent Text File I/O” 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	<b>RC_OK</b>	The Java <code>main()</code> method invoked by the launcher returned normally, or a <code>System.exit()</code> or <code>System.exit(0)</code> message was used to shutdown the JVM.
100	<b>RC_MAIN_EXCEPTION</b>	The Java main class not found or main method threw an exception.
101	<b>RC_CONFIG_ERR</b>	A configuration or setup error occurred. Check SYSOUT messages for more diagnostic information.
102	<b>RC_SYSTEM_ERR</b>	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 as their Java command-line launcher counterparts.

Changes to these environment variables 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 you 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  
| $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

---

### Introduction

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 datasets.
- Methods for allocating, deleting, and renaming MVS datasets.
- Methods for issuing single and multi-line WTOs.
- A callback interface for handling MVS Start, Modify, and Stop commands.
- An interface to z/OS Catalog Search (IGGCSI00).
- Class `com.ibm.jzos.ZUtil`, which contains APIs for many z/OS functions:
  - Obtaining job/step/user names, process ids, etc.
  - Writing SMF records
  - Reading environment variables
  - Reading OS Memory
- Class `com.ibm.jzos.PdsDirectory` for reading partitioned dataset directories.
- Class `com.ibm.jzos.ZLogstream` for writing to z/OS logger streams.
- 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/servers/eserver/zseries/software/java/jzos](http://www.ibm.com/servers/eserver/zseries/software/java/jzos) for more information.

---

### MVS Dataset I/O Using the ZFile Classes

The standard `java.io` package can only be used to access files in the HFS or zFS filesystem. The `com.ibm.jzos.ZFile` class complements `java.io` by providing low-level thin wrappers for the z/OS C/C++ Library IO routines – `fopen()`, `fread()`, `fwrite()`, etc., for accessing MVS datasets in stream (text or binary) and record mode. Javadoc for the `ZFile` class specifies each C library IO routine that is called, so that the Java programmer may reuse the excellent and well documented C/C++ IO library routines. Refer to the z/OS XL C/C++ Programming Guide, SC09–4765 for more information.

**Example: Reading an MVS Dataset in binary record mode using ZFile:**

```
ZFile zFile = new ZFile("//DD:INPUT", "rb,type=record,noseek");
try {
    byte[] recBuf = new byte[zFile.getLrecl()];
    String enc = Zutil.getDefaultPlatformEncoding();
    int nRead;
    while((nRead = zFile.read(recBuf)) > 0) {
        String line = new String(recBuf, 0, nRead, enc);
        System.out.println(line)
    };
} finally {
    zFile.close();
}
```

The C/C++ IO routines also allows datasets to be accessed in “text stream” mode, where records are converted to a stream containing “newline” characters. In stream mode, trailing spaces are removed from a record when reading and added if necessary to a record when writing.

**Example: Reading an MVS Dataset in text stream mode using ZFile:**

```
ZFile zFile = new ZFile("//DD:INPUT", "rt");
try {
    String enc = Zutil.getDefaultPlatformEncoding();
    InputStream is = zFile.getInputStream();
    BufferedReader rdr = new BufferedReader(
        new InputStreamReader(is, enc));

    String line;
    while ((line = rdr.readLine()) != null) {
        System.out.println(line);
    };
} finally {
    zFile.close();
}
```

## Platform Independent Text File I/O

The toolkit also provides the `com.ibm.jzos.FileFactory` class, which allows for portable creation of streams, readers and writers on Unix/Dos files (using the `java.io` package), or MVS datasets (using the `ZFile` class). The `FileFactory` class determines which underlying file system to use by the file name, so that a portable application can be configured at run time to use the appropriate filename / filesystem.

**Example: Copying files and/or datasets using the portable FileFactory class**

```
BufferedReader brdr = FileFactory.newBufferedReader(inFileName);
BufferedWriter bwtr = FileFactory.newBufferedWriter(outFileName);
try {
    String line;
    while ((line = brdr.readLine()) != null) {
        bwtr.write(line);
        bwtr.newLine();
    }
} finally {
    brdr.close();
    bwtr.close();
}
```

The `ZFile` class also includes the following methods:

- `ZFile.bpxwdyn(String cmd)`

This method is an interface to the flexible “BPXWDYN” text-based interface to MVS dynamic allocation. Refer to the z/OS Using REXX and z/OS UNIX System Services, SA22–7806 for more information.

- `ZFile.remove(String dsname)`
- `Zfile.rename(String oldname, String newname)`

These methods can be used to delete or rename an existing MVS dataset.

## 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("F001233E processing complete. Since this is a rather"
+ "long message, it will broken into a multi-line WTO on
+ "a word boundary as required." ,
0x0020, // routecode
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.ibm.com/servers/eserver/zseries/software/java/jzos/overview.html>.

---

## Hints and Tips

### 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.

---

---

<b>1007E</b>	<b>JZOS_MSG_ENV_VAR_SET "Error setting env var: %s = %s - %s".</b>
<b>Explanation:</b> Error message when environment variable cannot be set in launcher environment.	
<b>Programmer response:</b> Correct the invalid environment variable if possible.	
<b>System action:</b> The launcher terminates.	

---

<b>1008I</b>	<b>JZOS_MSG_ENV_STOP_STRING "%s".</b>
<b>Explanation:</b> Debugging info of environment stop string eyecatcher.	
<b>Programmer response:</b> No programmer response required.	
<b>System action:</b> No system action is taken.	

---

<b>1009E</b>	<b>JZOS_MSG_STDENV_HAS_EXIT "Child shell process exited without printing environment; //STDENV should not contain 'exit'".</b>
<b>Explanation:</b> Error message emitted when //STDENV contains an explicit exit call.	
<b>Programmer response:</b> Remove the exit call from //STDENV.	
<b>System action:</b> The launcher terminates.	

---

<b>1010E</b>	<b>JZOS_MSG_SIGPIPE_SIG_DFL "Error setting SIGPIPE to SIG_DFL - %s".</b>
<b>Explanation:</b> Issued when attempting to restore default signals for child process pipes.	
<b>Programmer response:</b> No programmer response required.	
<b>System action:</b> The launcher terminates.	

---

<b>1011E</b>	<b>JZOS_MSG_CREATE_JVM_ERROR "JNI_CreateJavaVM error, rc = %d".</b>
<b>Explanation:</b> Issued when JVM creation fails.	
<b>Programmer response:</b> Look up the return code in the JNI documentation and fix the condition if possible.	
<b>System action:</b> The launcher terminates.	

---

<b>1012I</b>	<b>JZOS_MSG_JAVA_VERSION_HEADER "Java Virtual Machine created. Version information follows:".</b>
<b>Explanation:</b> Header line emitted prior to Java SDK version information.	
<b>Programmer response:</b> No programmer response required.	
<b>System action:</b> No system action is taken.	

---

<b>1013I</b>	<b>JZOS_MSG_SYSTEM_EXIT "Calling System.exit()".</b>
<b>Explanation:</b> 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.	
<b>Programmer response:</b> No programmer response required.	
<b>System action:</b> No system action is taken.	

---

<b>1014I</b>	<b>JZOS_MSG_WAITING_FOR_THREADS "Waiting for non-daemon Java threads to finish before exiting...".</b>
<b>Explanation:</b> The launcher is preparing to terminate, but will wait for non-daemon threads to complete.	
<b>Programmer response:</b> No programmer response required.	
<b>System action:</b> 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.

---

## Non-Messages Strings Used in some Java Exceptions

JZOS_TLKT_BPXDYN_MALLOC_ERROR	"bpxwdyn __malloc31() failed"
JZOS_TLKT_INVALID_FILE_HANDLE	"Invalid file handle"
JZOS_TLKT_BPXDYN_FAILED	"bpxwdyn failed"
JZOS_TLKT_DDEXISTS_MALLOC_ERROR	"ddExists: __malloc31() error"
JZOS_TLKT_REMOVE_FAILED	"remove() failed"
JZOS_TLKT_RENAME_FAILED	"rename() failed"
JZOS_TLKT_FCLOSE_FAILED	"fclose() failed"
JZOS_TLKT_FDELREC_FAILED	"fdelrec() failed"
JZOS_TLKT_FFLUSH_FAILED	"fflush() failed"
JZOS_TLKT_FGETPOS_FAILED	"fgetpos() failed"
JZOS_TLKT_FLDATA_FAILED	"initialize: fldata() failed"
JZOS_TLKT_FOPEN_FAILED	"fopen() failed"
JZOS_TLKT_FLOCATE_FAILED	"flocate() failed"
JZOS_TLKT_FREOPEN_FAILED	"freopen() failed"
JZOS_TLKT_REWIND_FAILED	"rewind() failed"
JZOS_TLKT_FSEEKO_FAILED	"fseeko() failed"
JZOS_TLKT_FSETPOS_FAILED	"fsetpos() failed"
JZOS_TLKT_FREAD_FAILED	"fread() failed"
JZOS_TLKT_FWRITE_FAILED	"fwrite() failed"
JZOS_TLKT_FSETPOS_BUFSIZE_ERROR	"fsetpos() buffer is wrong size"
JZOS_TLKT_FTELLO_FAILED	"ftello() failed"
JZOS_TLKT_BUFFER_TOO_SMALL	"buffer too small for the request"
JZOS_TLKT_FUPDATE_FAILED	"fupdate() failed"
JZOS_TLKT_ALLOC_DD_MALLOC_ERROR	"allocDummyDDName: __malloc31() error"
JZOS_TLKT_ALLOC_DD_SVC99_ERROR	"allocDummyDDName: svc99() error"
JZOS_TLKT_PREVIOUS_EXCEPTION	"Previous Exception"
JZOS_TLKT_UNHANDLED_EXCEPTION	"Unhandled exception found"
JZOS_TLKT_PEEK_ACCESS_ERROR	"peekOSMemory() storage access exception"
JZOS_TLKT_PEEK_SYSTEM_ERROR	"peekOSMemory() system error"
JZOS_TLKT_GET_START_PARAMS_ERROR	"retrieveStartParameters - unexpected error from GSTRCMD"
JZOS_TLKT_OFFSET_LEN_LESS_0	"offset and length arguments may not be < 0"

---

## 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.

The Java SDK product version of JZOS differs from the original pre-2.0 alphaWorks version in several ways. This document describes the high level differences. Consult the source code in the migration package for detailed API differences on IBM alphaWorks at <http://www.alphaworks.ibm.com/tech/zosjavabatchtk?>

---

### JZOS Batch Launcher Changes

- The launcher messages have been rewritten and have a new message prefix: JVMJZBL
- The launcher now emits one of the following four return codes upon completion:
  - 0 RC\_OK
  - 100 RC\_MAIN\_EXCEPTION
  - 101 RC\_CONFIG\_ERR
  - 102 RC\_SYSTEM\_ERR
- A detailed message accompanies these return codes which provides additional information about the failure reason. These four return codes replace the many return codes emitted by the alphaWorks version.
- The launcher now uses local spawn() rather than fork() to adopt the environment specified via the STDENV DD.

---

### JZOS Toolkit Changes

- The toolkit messages have been rewritten and have a new message prefix: JVMJZTK
- The package prefix for all Toolkit classes have changed from com.dovetail.jzos to com.ibm.jzos. The jzos\_migration.jar can be used as a temporary adapter for existing code. This jar retains the com.dovetail.jzos prefix, but calls the associated com.ibm.jzos API. In general, most of the alphaWorks functionality is available in the SDK via this migration jar, but some exceptions are noted below:
  - There is no migration class available for com.ibm.jzos.PdsDirectory. Any existing code that uses com.dovetail.jzos.PdsDirectory will need to be recompiled to use the new class.
  - The com.dovetail.jzos.ZFile methods hfsExists(), getVolumnList(), locateDSN() and readDSCB() are no longer supported.
  - There are several new methods in com.ibm.jzos.ZUtil for getting additional information about the current job: getCurrentProcStepName(), getCurrentStepName(), getCurrentTsoPrefix().
  - There are several new methods in com.ibm.jzos.ZUtil for peeking OS memory. See the user's guide for details.
- The MVS Console support has been significantly reworked in the SDK version of the JZOS toolkit:
  - The MvsCommandCallback class now supports the STOP command in addition to START and MODIFY.

- The MODIFY command text must now be prefixed with the string "APPL=".
- The WTO and console interaction functionality has been moved from ZUtil to the new classes MvsConsole and WtoMessage. See the jzos\_migration.jar source code for details on how to use these new classes.

---

## 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 (tttt1 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 dataset with the correct parameters for SYS1.SIEALNKE and SYS1.AIEALNKE. Fields that need filling in are volume (tttt2 and ddddd) and target libraries (tvoll). 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 dataset ("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 (tvoll).                                       *
/*                                                                    *
/* 4) Change ddddd to the volser for the distribution              *
/* libraries.                                                       *
/*                                                                    *
/* If you specify a volume for any dataset 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/servers/eserver/zseries/software/java/jzos](http://www.ibm.com/servers/eserver/zseries/software/java/jzos).

---

## ++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 qualifier (hlq), and volume (tttt1, tttt2 and ddddd).

**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(tttt1)
    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 dataset with the correct parameters for hlq.SIEALNKE and hlq.AIEALNKE. Fields that need filling in are volume (ttttt1 and ddddd).

```

//*****
/** 1. Add the job parameters to meet your system
/**    requirements.
/**
/** 2. Change DSP and DDSP variable to the appropriate final
/**    disposition of the dataset ("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 (tv01).
/**    Change ttttt2 to the volser for second volume for the
/**        target libraries (tv02).
/**
/** 4. Change ddddd to the volser for the distribution
/**    libraries.
/**
/** If you specify a volume for any dataset 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=h1q,      * HLQ for the target library
//          DHLQ=h1q,    * 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=ddddd  * 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 LNKLIST if not already there. More information about the LNKLIST 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/servers/eserver/zseries/software/java/jzos](http://www.ibm.com/servers/eserver/zseries/software/java/jzos).



---

## 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 unaxed 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 33, 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 unaxed 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 JMVMPC50 "'SYS1.PROCLIB(JMVMPC50)'"
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 33, 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 Product

1. These instructions assume that the non-SMP/E pax file has been uploaded and unaxed 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 33, 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 unaxed 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 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 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 33, 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 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 LNKLIST 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 (+l) 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

```

## 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} "  
  
//
```

---

## 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.

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

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie, New York 12601-5400  
U.S.A.  
Attention: Information Request

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

---

## Trademarks

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

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, other countries, or both.

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

Other company, product, and service names may be trademarks or service marks of others.



---

## 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

## C

configuring  
    configuring environment variables 8  
    environment variables 8  
    Java SDK environment 9  
    Jzos environment 9  
    system environment 9  
controlling output encoding 12  
customizing configuration script 11

## H

handling MVS MODIFY 19  
handling MVS START 19

## I

installation  
    upload files 3

## J

Java SDK options 10  
Jzos Batch Launcher  
    arguments 7  
    introduction 5  
    sample JCL 6  
    sample PROC 5  
    setting logging levels 8  
    specifying the Java main class 7

## L

LE runtime options 14  
LookAt message retrieval tool v

## M

message retrieval tool, LookAt v  
messages 21  
migration  
    JZOS Batch Launcher Changes 31  
    JZOS Toolkit Changes 31  
MVS console interface 12  
MVS dataset I/O  
    ZFile classes 17

## N

Non-SMP/E Install 3, 41

## O

overview  
    features 1

## S

SDK5  
    sample JCL 47  
    sample PROC 47  
SMP/E Install 33  
system properties 10

## T

Toolkit User's Guide  
    introduction 17

## W

writing messages to the system console or log 19



---

## 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. SA23-2245-02**

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 e-mail to: [mhvrcfs@us.ibm.com](mailto: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.

\_\_\_\_\_

E-mail address



Fold and Tape

Please do not staple

Fold and Tape



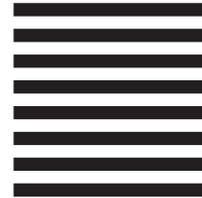
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
MHVRCFS, Mail Station P181  
2455 South Road  
Poughkeepsie, New York  
12601-5400



Fold and Tape

Please do not staple

Fold and Tape





Printed in USA

SA23-2245-02

