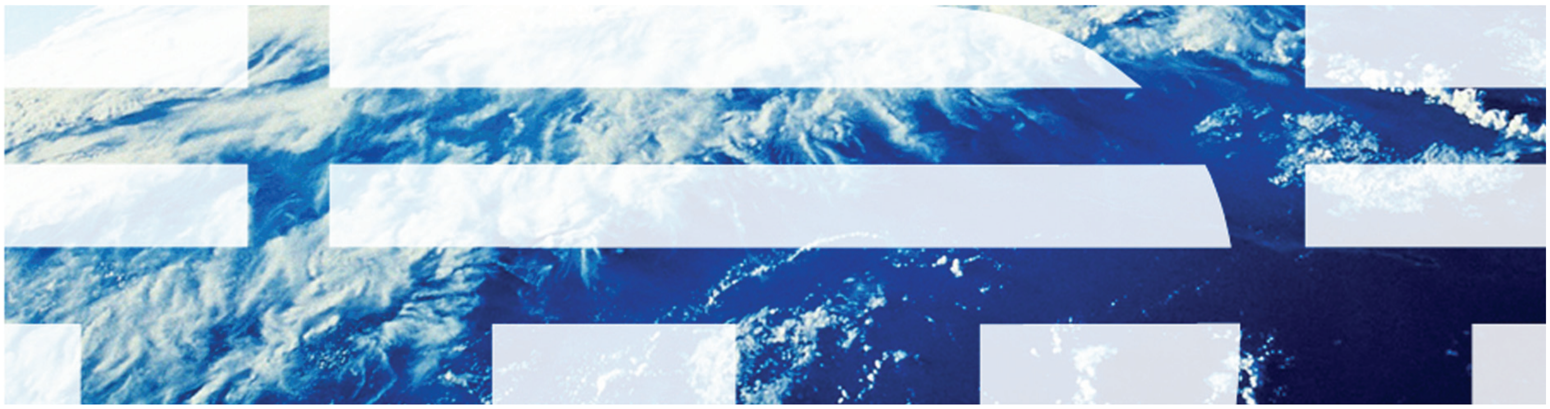


IBM Security Key Lifecycle Manager for z/OS 1.1: Operator Commands

White Paper

July 2011

Ori Pomerantz
orip@us.ibm.com

Table of contents

Introduction. 1

Audience 1

Acknowledgements 1

1 Running Java programs from JCL jobs 2

1.1 JVMLDM<version>, the z/OS program 2

1.2 JVMPRC<version>, the JCL procedure 3

1.2.1 Editing the JCL procedure 3

1.2.1.1 The library JCL variable 3

1.2.1.2 The steplib 3

1.3 JVMJCL<version>, JCL sample 4

1.3.1 Editing the JCL sample 4

1.4 Verifying the JCL works 5

1.4.1 Changing the Java class 5

2 Started tasks 6

2.1 Adding a data set to the started task data set list 6

2.2 Creating a started task 10

2.3 Test the started task 12

3 The ISKLM started task. 13

3.1 ISKLMENV, the environment setup script 13

3.2 ISKLM, the started task 13

3.3 Controlling ISKLM from a started task 14

Introduction

You learn how to implement operator commands to control IBM Security Key Lifecycle Manager (ISKLM) for z/OS from the console of a mainframe. This task requires two prerequisites that you can also learn from this paper:

1. How to run Java programs from jobs.
2. How to create a started task to be able to issue commands to ISKLM.

Audience

This paper is for implementers who install ISKLM for z/OS, and must create operator commands to simplify the operation and administration of the product.

Acknowledgements

Many thanks for the help from John Peck, Martien Balgooi Van, Mark S. Hahn, and Monique Conway.

1 Running Java programs from JCL jobs

Operator commands require a job or a started task that waits to receive operator commands. Started tasks are written in JCL, so the first step is to run a Java program from JCL.

The components that are required to run Java programs from JCL are already available as part of the Java virtual machine (JVM). They are typically stored in z/OS UNIX System Services (USS), under **$JAVA_HOME/mvstools**.

Note: In this white paper, **$JAVA_HOME** refers to the directory in z/OS UNIX System Services where Java is installed. Typically, this would be **/usr/lpp/java/J6.0** for Java 6.0.

1.1 JVMLDM<version>, the z/OS program

The first component is the z/OS program that runs the JVM. This program is called **JVMLDM<*version*>**. It must be made available to JCL. Either use SMP/E, or allocate a new data set (by using the record format **U** and data set name type **LIBRARY**) to store the program. **JZOS.LOADLIB** is a reasonable name.

Open USS. Change to the **$JAVA_HOME/mvstools** folder and run the following command to copy the program:

```
cp JVMLDM60 "//'JZOS.LOADLIB(JVMLDM60)'"
```

Note: If your version of Java is not 6.0, or if the selected data set is not **JZOS.LOADLIB**, modify the command.

1.2 JVMPRC<*version*>, the JCL procedure

This JCL procedure contains configuration parameters that pertain to the JVM and are unlikely to change between jobs. It is stored in **$JAVA_HOME/mvstools/samples/jcl**.

To copy **JVMPRC<*version*>** to a procedure library, issue a command similar to the one as follows:

```
cp JVMPRC60 "//'USER.PROCLIB(JVMPRC60)'"
```

Alternatively, you can allocate a new data set (by using record format **FB**, record length **80**, and data set name type **PDS**) to store the program. In that case, you must specify the name of the new data set in the JCL job. Before the **EXEC** statement, place a line similar to the one as follows:

```
//JCL      JCLLIB ORDER=<name of new data set>
```

1.2.1 Editing the JCL procedure

Edit the JCL procedure as follows to access the JVM.

1.2.1.1 The library JCL variable

Note the following code:

```
//*  LIBRARY='<HLQ>.JZOS.LOADLIB', < STEPLIB FOR JVMLDM
```

By default, this line is commented out. Unless you used SMP/E to install the **JVMLDM<*version*>** into one of the linklib data sets, you must uncomment this line and specify the data set that contains the **JVMLDM<*version*>**. For example, if you used **JZOS.LOADLIB**, the correct line is as follows:

```
// LIBRARY='JZOS.LOADLIB',
```

1.2.1.2 The steplib

The second line that must be changed is as follows:

```
//*STEPLIB  DD DSN=&LIBRARY,DISP=SHR
```

Unless you used SMP/E to install the **JVMLDM<*version*>** into one of the linklib data sets, you must uncomment this line. Delete the asterisk.

```
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
```

1.3 JVMJCL<*version*>, JCL sample

This example JCL job uses **JVMLDM<*version*>** and **JVMPRC<*version*>** to run a Java class. Copy it to a data set that holds JCL jobs. You can also use a command similar to the following one to copy it to its own data set:

```
cp JVMJCL60 //JVMJCL60
```

This command copies the file to ***<your user name>*.JVMJCL60**.

1.3.1 Editing the JCL sample

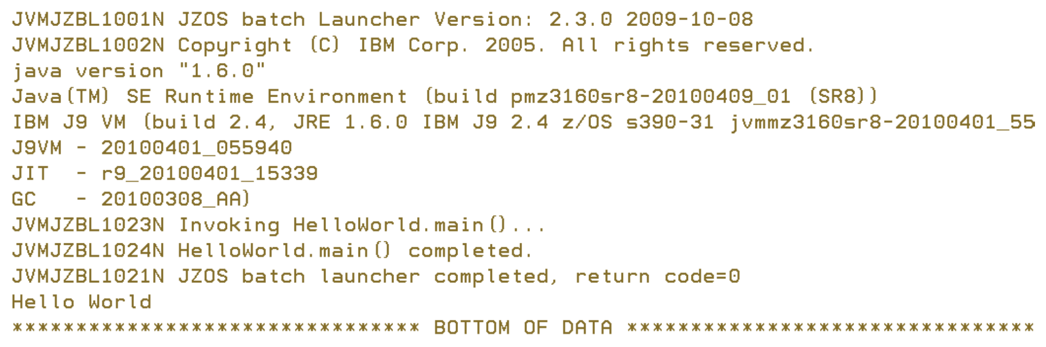
The JCL sample does not contain a job card (the first lines of JCL that specify the name of the job). Nor does it contain the name of the procedure library that contains the JCL procedure for Java. Add the two following lines in the top of the JCL sample, modifying as appropriate to your installation:

```
//RUNJVM JOB CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
//PROCLIB JCLLIB ORDER=JZOS.PROCLIB
```

One of the data definitions is **STDENV**. This file is a UNIX shell script that runs before the JVM. If Java is not installed in the default location, you must change the line that specifies **JAVA_HOME**.

1.4 Verifying the JCL works

Submit the JCL job. Go to SDSF and view the job results as shown in Figure 1.



```
JVMJZBL1001N JZOS batch Launcher Version: 2.3.0 2009-10-08
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.6.0"
Java(TM) SE Runtime Environment (build pmz3160sr8-20100409_01 (SR8))
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 z/OS s390-31 jvmmz3160sr8-20100401_55
J9VM - 20100401_055940
JIT  - r9_20100401_15339
GC   - 20100308_AA)
JVMJZBL1023N Invoking HelloWorld.main()...
JVMJZBL1024N HelloWorld.main() completed.
JVMJZBL1021N JZOS batch launcher completed, return code=0
Hello World
******************************** BOTTOM OF DATA ********************************
```

Figure 1: Results of a job that uses the JVM

The bottom displays a “Hello World” message, indicating that the Java class ran correctly.

1.4.1 Changing the Java class

The JCL job runs a class called **HelloWorld**, which is provided with the JVM for testing purposes. To run a different class, change the **JAVACLS** value in the line below the one that specifies the procedure name (**JVMPRC<*version*>**).

2 Started tasks

Started tasks are JCL jobs that are stored in specific data sets. The operator can control the commands from either the console or SDSF.

2.1 Adding a data set to the started task data set list

This procedure adds a data set to the list that is searched for started tasks:

Note: This procedure applies to installation that use JES2. If your installation uses JES3, ask a system programmer for help.

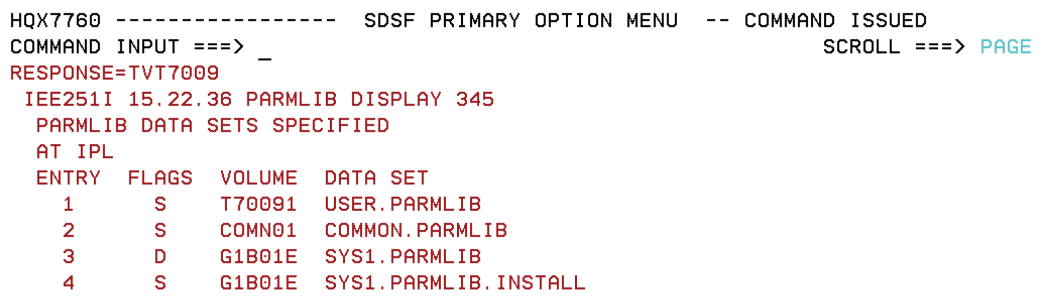
Perform the following steps to create a data set for the started task.

1. Run the following console command to get the list of parameter libraries:

```
D PARMLIB
```

Tip: There are two common ways to enter console command. One is to log on to the console. The other is to enter SDSF. Within SDSF, type slash (/), followed by the command in the command input field. To enter a long command in SDSF, type slash on its own in the command input field to open the system command extension panel.

The result is similar to Figure 2.



```
HQX7760 -----------------  SDSF PRIMARY OPTION MENU  -- COMMAND ISSUED
COMMAND INPUT ===> _                                     SCROLL ===> PAGE
RESPONSE=TVT7009
 IEE251I 15.22.36 PARMLIB DISPLAY 345
  PARMLIB DATA SETS SPECIFIED
  AT IPL
  ENTRY  FLAGS  VOLUME  DATA SET
    1      S    T70091  USER.PARMLIB
    2      S    COMN01  COMMON.PARMLIB
    3      D    G1B01E  SYS1.PARMLIB
    4      S    G1B01E  SYS1.PARMLIB.INSTALL
```

Figure 2: Results of D PARMLIB

The four data sets are as follows:

– USER.PARMLIB

– COMMON.PARMLIB

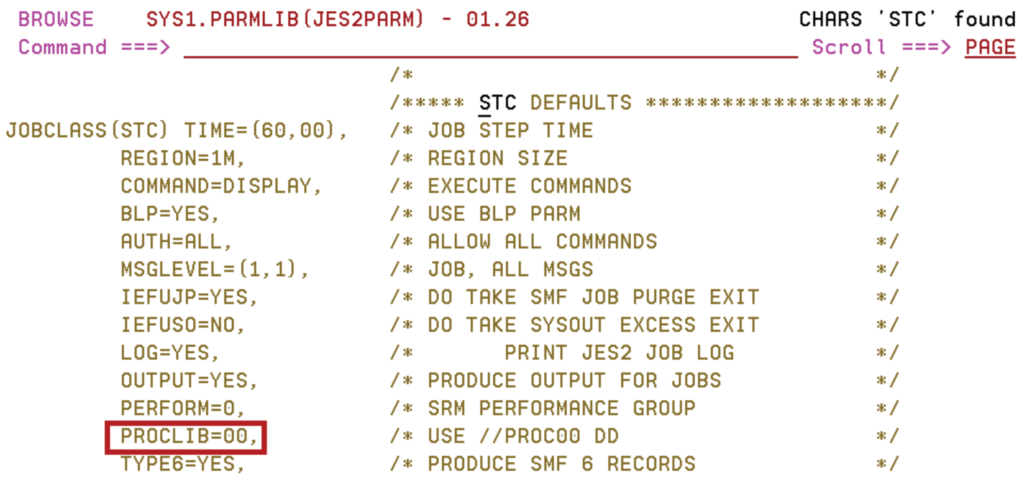
– SYS1.PARMLIB

– SYS1.PARMLIB.INSTALL

If the system is looking for a parameter member called, for example, **CLOCK00**, it looks first in **USER.PARMLIB(CLOCK00)**. If it does not find it, it looks in **COMMON.PARMLIB(CLOCK00)**, **SYS1.PARMLIB(CLOCK00)**, and finally, **SYS1.PARMLIB.INSTALL(CLOCK00)**.

2. In the parameter libraries, find the **JES2PARM** member that contains the definition of the **STC** (started task) job class. Find the **PROCLIB** parameter to identify the data set list that is used for started tasks.

   The name is **PROC** followed by the **PROCLIB** parameter value. For example, Figure 3 shows that the data set list is **PROC00**.



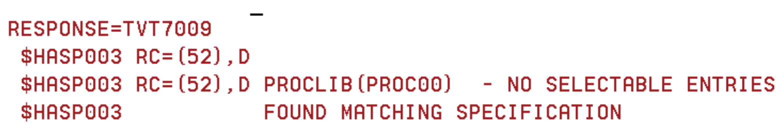
```
 BROWSE    SYS1.PARMLIB(JES2PARM) - 01.26                    CHARS 'STC' found
 Command ===> ________________________________________________ Scroll ===> PAGE
                               /*                                 */
                               /***** STC DEFAULTS ******************/
JOBCLASS(STC) TIME=(60,00),    /* JOB STEP TIME                   */
         REGION=1M,            /* REGION SIZE                     */
         COMMAND=DISPLAY,      /* EXECUTE COMMANDS                */
         BLP=YES,              /* USE BLP PARM                    */
         AUTH=ALL,             /* ALLOW ALL COMMANDS              */
         MSGLEVEL=(1,1),       /* JOB, ALL MSGS                   */
         IEFUJP=YES,           /* DO TAKE SMF JOB PURGE EXIT      */
         IEFUSO=NO,            /* DO TAKE SYSOUT EXCESS EXIT      */
         LOG=YES,              /*       PRINT JES2 JOB LOG        */
         OUTPUT=YES,           /* PRODUCE OUTPUT FOR JOBS         */
         PERFORM=0,            /* SRM PERFORMANCE GROUP           */
         PROCLIB=00,           /* USE //PROC00 DD                 */
         TYPE6=YES,            /* PRODUCE SMF 6 RECORDS           */
```

Figure 3: Definition of the data set list for started tasks

3. Run the following console command to see if there is already a dynamic data set list for started tasks. Substitute your own value for **PROC00** if necessary.

```
$D PROCLIB(PROC00)
```

4. If you receive a message similar to Figure 4, you must create the dynamic data set list. Otherwise, skip to the next step.



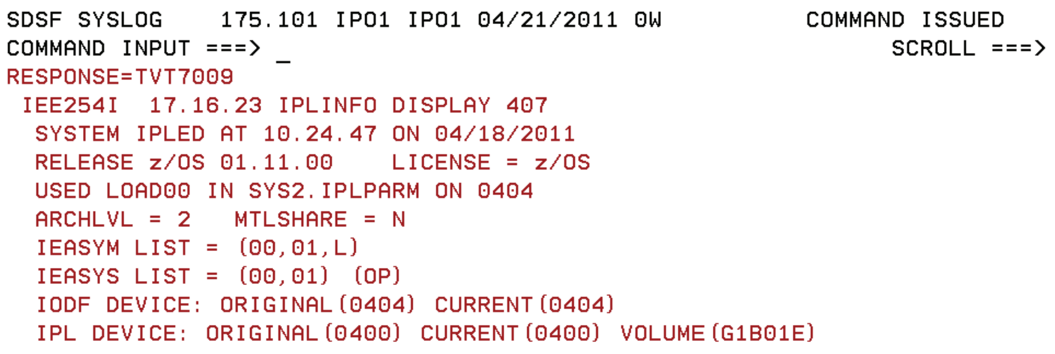
```
RESPONSE=TVT7009
 $HASP003 RC=(52),D
 $HASP003 RC=(52),D PROCLIB(PROC00)  - NO SELECTABLE ENTRIES
 $HASP003           FOUND MATCHING SPECIFICATION
```

Figure 4: Definition of the PROCLIB for started tasks

If there is no dynamic data set list, you must create it, and add the existing data sets to it in addition to the new one. To do so, perform the following steps:

a. Identify the data set member that runs the JES2 subsystem. To do so, run the console command to get information about the parameters that are used during the reboot. The command is called in mainframe terminology initial program load, or IPL:

```
D IPLINFO
```

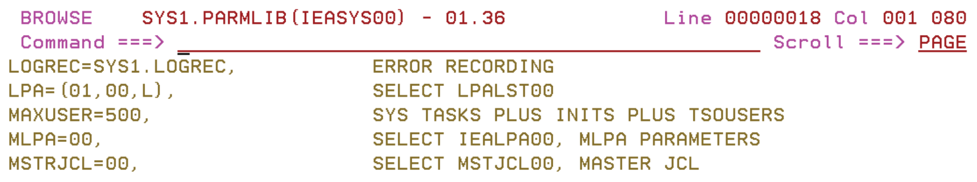


```
SDSF SYSLOG    175.101 IP01 IP01 04/21/2011 0W          COMMAND ISSUED
COMMAND INPUT ===> _                                          SCROLL ===>
RESPONSE=TVT7009
 IEE254I   17.16.23 IPLINFO DISPLAY 407
  SYSTEM IPLED AT 10.24.47 ON 04/18/2011
  RELEASE z/OS 01.11.00    LICENSE = z/OS
  USED LOAD00 IN SYS2.IPLPARM ON 0404
  ARCHLVL = 2   MTLSHARE = N
  IEASYM LIST = (00,01,L)
  IEASYS LIST = (00,01) (OP)
  IODF DEVICE: ORIGINAL(0404) CURRENT(0404)
  IPL DEVICE: ORIGINAL(0400) CURRENT(0400) VOLUME(G1B01E)
```

Figure 5: Results of D IPLINFO

The relevant field is the **IEASYS LIST**. The value shown in Figure 5 points to the configuration parameters in the **IEASYS00** and **IEASYS01** members of one of the parameter data sets.

b. Open those members in the parameter data set list. Find the **MSTRJCL** parameter, which defines the MASTER-jcl, the base MVS system that runs all other functions.

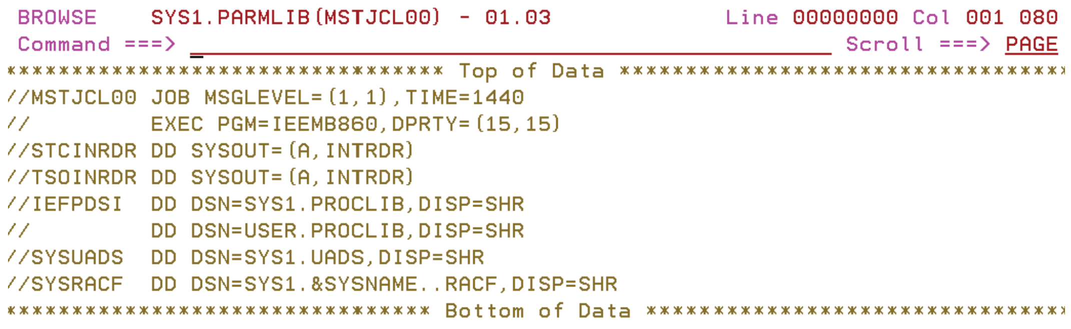


```
 BROWSE    SYS1.PARMLIB(IEASYS00) - 01.36             Line 00000018 Col 001 080
 Command ===> ________________________________________________ Scroll ===> PAGE
LOGREC=SYS1.LOGREC,           ERROR RECORDING
LPA=(01,00,L),                SELECT LPALST00
MAXUSER=500,                  SYS TASKS PLUS INITS PLUS TSOUSERS
MLPA=00,                      SELECT IEALPA00, MLPA PARAMETERS
MSTRJCL=00,                   SELECT MSTJCL00, MASTER JCL
```

Figure 6: IEASYS00 member with the MSTRJCL parameter

For example, the value here is **MSTRJCL=00**, which means that the member of the parameter library that defines the MASTER-jcl is **MSTJCL00**.

c. Open that member in the parameter data set list.



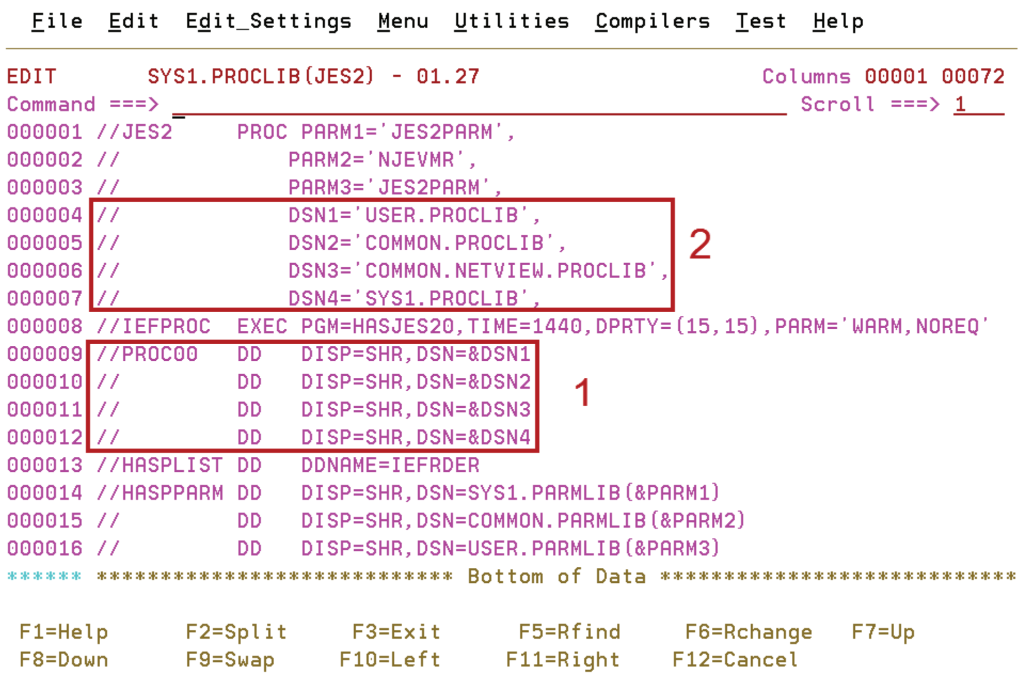
```
 BROWSE    SYS1.PARMLIB(MSTJCL00) - 01.03            Line 00000000 Col 001 080
 Command ===> _                                                Scroll ===> PAGE
********************************* Top of Data **********************************
//MSTJCL00 JOB MSGLEVEL=(1,1),TIME=1440
//         EXEC PGM=IEEMB860,DPRTY=(15,15)
//STCINRDR DD SYSOUT=(A,INTRDR)
//TSOINRDR DD SYSOUT=(A,INTRDR)
//IEFPDSI  DD DSN=SYS1.PROCLIB,DISP=SHR
//         DD DSN=USER.PROCLIB,DISP=SHR
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSRACF  DD DSN=SYS1.&SYSNAME..RACF,DISP=SHR
******************************** Bottom of Data ********************************
```

Figure 7: MSTRJCL00 member with the IEFPDSI data definition

The IEFPDSI data definition specifies the proclib data sets where the **JES2** procedure is defined. In this example, look for **SYS1.PROCLIB(JES2)** and **USER.PROCLIB(JES2)**.

d. Open the JES2 member and identify the data sets in the static data set list that is used for started tasks now.

For example, Figure 8 shows that the data definition is **PROC00**, so the data sets are the variables **DSN1**, **DSN2**, and so on. Those variables were defined earlier as **USER.PROCLIB**, **COMMON.PROCLIB**, and so on.



```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------
EDIT       SYS1.PROCLIB(JES2) - 01.27                  Columns 00001 00072
Command ===> _                                              Scroll ===> 1
000001 //JES2     PROC PARM1='JES2PARM',
000002 //              PARM2='NJEVMR',
000003 //              PARM3='JES2PARM',
000004 //              DSN1='USER.PROCLIB',
000005 //              DSN2='COMMON.PROCLIB',                  2
000006 //              DSN3='COMMON.NETVIEW.PROCLIB',
000007 //              DSN4='SYS1.PROCLIB',
000008 //IEFPROC  EXEC PGM=HASJES20,TIME=1440,DPRTY=(15,15),PARM='WARM,NOREQ'
000009 //PROC00   DD   DISP=SHR,DSN=&DSN1
000010 //         DD   DISP=SHR,DSN=&DSN2     1
000011 //         DD   DISP=SHR,DSN=&DSN3
000012 //         DD   DISP=SHR,DSN=&DSN4
000013 //HASPLIST DD   DDNAME=IEFRDER
000014 //HASPPARM DD   DISP=SHR,DSN=SYS1.PARMLIB(&PARM1)
000015 //         DD   DISP=SHR,DSN=COMMON.PARMLIB(&PARM2)
000016 //         DD   DISP=SHR,DSN=USER.PARMLIB(&PARM3)
****** **************************** Bottom of Data ****************************

 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 8: The JES2 data set member

e. Run the following console command to create the dynamic data set list:

```
$add proclib(proc00),dd(1)=(dsn=<1st data set in the
list>)
```

f. For every other data set in the static list, issue the following command to append it to the end of the list:

```
$t proclib(proc00),dd(99)=(dsn=<data set name>)
```

Note: This command adds the data set in the 99th place. However, the operating system puts it at the lowest free place. Therefore, it is possible to reuse this location.

5. To add the new data set to the list, issue the following console command:

```
$t proclib(proc00),dd(99)=(dsn=<new data set>)
```

2.2 Creating a started task

Create a started task of your own by performing the following steps:

1. Specify the standard environment for Java in a data set member. On my system, I allocated a data set called **JZOS.PARMLIB** and used **JZOS.PARMLIB(STDENV)**.

```
. /etc/profile
```

Replace with the directory where Java is installed on your system. Often, this value is **/usr/lpp/java/J6.0** for Java version 6.0:

```
export JAVA_HOME=/u/usr/lpp/java/J6.0
export PATH=$JAVA_HOME/bin:/bin:$PATH
export CLASSPATH=$JAVA_HOME/lib:$JAVA_HOME/lib/
ext:$JAVA_HOME
LIBPATH=/lib:/usr/lib:$JAVA_HOME/bin:$JAVA_HOME/bin/j9vm
LIBPATH=$LIBPATH:$JAVA_HOME/bin/classic
export LIBPATH=$LIBPATH:$JAVA_HOME/bin/s390
env
```

2. Enter the following JCL code as a **JVMTEST** member in the started task data set that you added previously (**JZOS.PROCLIB**, for example):

```
//JVMTEST  PROC JAVACLS='HelloWorld',
```

Replace JZOS.LOADLIB with the name of the data set that contains the **JVMLDM<*version*>** member on your system.

```
//   LIBRARY='JZOS.LOADLIB',
```

Replace with the Java version that you use, if different.

```
//   VERSION='60'
//*
//JVMTEST  EXEC PGM=JVMLDM&VERSION,
//   PARM='&JAVACLS',REGION=512M
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
```

Replace JZOS.PARMLIB(STDENV) with the name of the data set and member where you placed the standard environment.

```
//STDENV   DD DSN=JZOS.PARMLIB(STDENV),DISP=SHR
```

3. If RACF, the security subsystem, controls started tasks (the STARTED class being active), run the two following TSO commands to create a generic profile for this job:

```
RDEFINE STARTED JVMTEST.** UACC(ALTER)
STDATA(USER(userid))
SETROPTS RACLIST(STARTED) REFRESH
```

2.3 Test the started task

Perform the following steps:

1. Run the following console command:

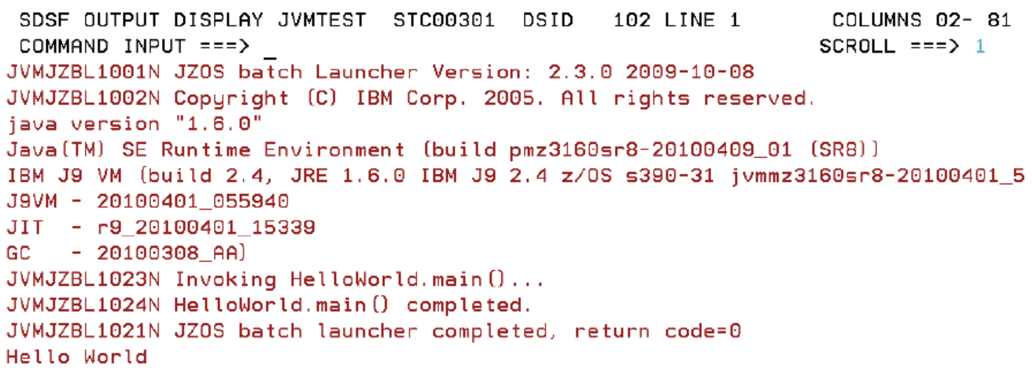
   ```
   S JVMTEST
   ```

2. In SDSF, view the status of the JVMTEST job as follows:

   a. Type **ST** to view job status.

   b. Find the JVMTEST job and type **S** next to it.

   c. If everything ran correctly, the bottom of the result resembles the following screen capture:



```
 SDSF OUTPUT DISPLAY JVMTEST  STC00301  DSID   102 LINE 1       COLUMNS 02- 81
 COMMAND INPUT ===> _                                          SCROLL ===> 1
JVMJZBL1001N JZOS batch Launcher Version: 2.3.0 2009-10-08
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.6.0"
Java(TM) SE Runtime Environment (build pmz3160sr8-20100409_01 (SR8))
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 z/OS s390-31 jvmmz3160sr8-20100401_5
J9VM - 20100401_055940
JIT  - r9_20100401_15339
GC   - 20100308_AA)
JVMJZBL1023N Invoking HelloWorld.main()...
JVMJZBL1024N HelloWorld.main() completed.
JVMJZBL1021N JZOS batch launcher completed, return code=0
Hello World
```

Figure 9: Expected result of the JVMTEST started task

3 The ISKLM started task

ISKLM contains two files that are used for the ISKLM started task. After using SMP/E to install the product, those files are located in the following data sets:

- *HLQ*.**CKL.SCKLSAMP(ISKLM)**: The started task to put in one of the started task data sets
- *HLQ*.**CKL.SCKLSAMP(ISKLMENV)**: The data set member that specifies the environment

Note: These files are also available in the SMP/E distribution library, *HLQ*.**CKL.ACKLSAMP**. However, administrators are not supposed to use the distribution library for any purpose except for SMP/E installation.

3.1 ISKLMENV, the environment setup script

This UNIX script sets up environment variables. The following lines specify the installation location of ISKLM and Java, which you might need to modify:

```
export ISKLM_HOME="/usr/lpp/ISKLM"
export JAVA_HOME="/usr/lpp/java/J5.0"
```

The following line specifies the location of the Java archive (JAR) file that contains the software:

```
CLASSPATH=$CLASSPATH:/usr/lpp/ISKLM/IBMSKLM.jar
```

The following line is the name of the properties file:

```
export ZZZZ="/u/isklmsrv/ISKLMConfig.properties.zos"
```

3.2 ISKLM, the started task

Copy the ISKLM file into one of the started task data sets. You might need to modify some of the lines:

Specify the data set that contains the **JVMLDM<*version*>** member:

```
//   LIBRARY='ROOT.JAVA.LINKLIB', < STEPLIB FOR JVMLDM mod..
```

If necessary, change to reflect the version of Java that you use:

```
//    VERSION='50', < JVMLDM version: 14, 50, 56
```

If necessary, change to reflect the data set, or data set member, that contains the environment setup script.

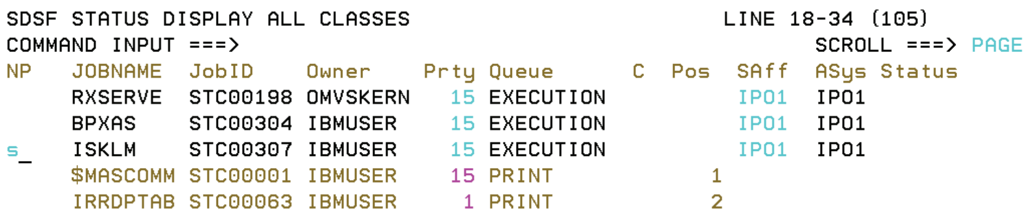
```
//STDENV DD DSN=USER.PARMLIB(ISKLMENV),DISP=SHR
```

3.3 Controlling ISKLM from a started task

1. To start ISKLM, run the following console command:

   ```
   S ISKLM
   ```

2. In SDSF, type **ST** to see the status of the ISKLM job. It is in the execution queue because ISKLM is currently running.



```
SDSF STATUS DISPLAY ALL CLASSES                        LINE 18-34 (105)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP    JOBNAME  JobID    Owner    Prty Queue      C  Pos  SAff  ASys Status
      RXSERVE  STC00198 OMVSKERN   15 EXECUTION          IPO1  IPO1
      BPXAS    STC00304 IBMUSER    15 EXECUTION          IPO1  IPO1
s_    ISKLM    STC00307 IBMUSER    15 EXECUTION          IPO1  IPO1
      $MASCOMM STC00001 IBMUSER    15 PRINT            1
      IRRDPTAB STC00063 IBMUSER     1 PRINT            2
```

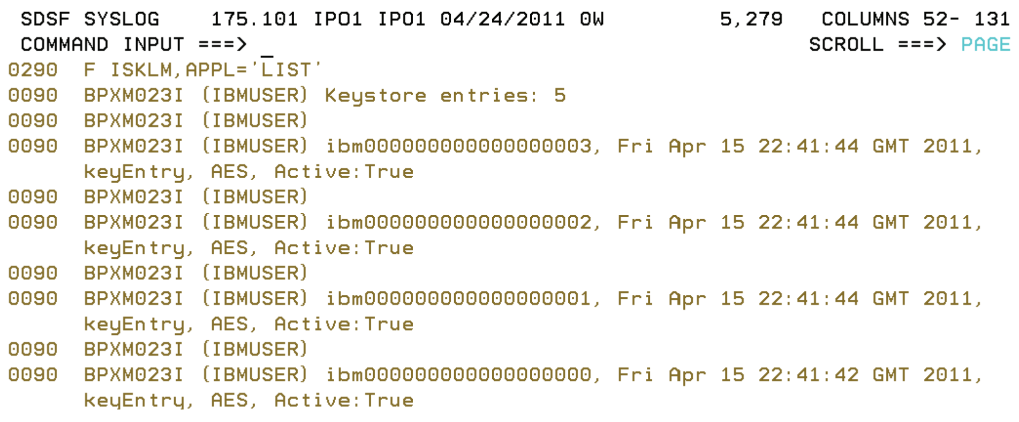
Figure 10: The ISKLM job in the execution queue

3. Type **S** next to the ISKLM job name and view the job results.

4. Run the following console command to get the key list:

   ```
   F ISKLM,APPL='list'
   ```

5. In SDSF, type **LOG** to view the system log, which contains the key list.



```
 SDSF SYSLOG    175.101 IPO1 IPO1 04/24/2011 0W         5,279   COLUMNS 52- 131
 COMMAND INPUT ===> _                                          SCROLL ===> PAGE
0290  F ISKLM,APPL='LIST'
0090  BPXM023I (IBMUSER) Keystore entries: 5
0090  BPXM023I (IBMUSER)
0090  BPXM023I (IBMUSER) ibm000000000000000003, Fri Apr 15 22:41:44 GMT 2011,
      keyEntry, AES, Active:True
0090  BPXM023I (IBMUSER)
0090  BPXM023I (IBMUSER) ibm000000000000000002, Fri Apr 15 22:41:44 GMT 2011,
      keyEntry, AES, Active:True
0090  BPXM023I (IBMUSER)
0090  BPXM023I (IBMUSER) ibm000000000000000001, Fri Apr 15 22:41:44 GMT 2011,
      keyEntry, AES, Active:True
0090  BPXM023I (IBMUSER)
0090  BPXM023I (IBMUSER) ibm000000000000000000, Fri Apr 15 22:41:42 GMT 2011,
      keyEntry, AES, Active:True
```

Figure 11: The ISKLM key list in the system log

6. Run the following console command to stop ISKLM:

```
P ISKLM
```