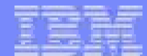
Java interoperability

Martina Schmidt

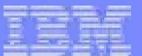
© 2007 IBM Corporation

IBM Systems



Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - Crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler



Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

A short z/OS data overview

MVS dataset access

- Special Java APIs for z/OS file access available
  - Reading and writing of data
  - Allocation of datasets
- java.io for USS file access available
  - Applications are portable!

API comparison

Type of access required	JZOS	JRIO	java.io
C/C++ library interface			
Java data set record stream abstractions			
Fine access to system error codes			
Data set access (Text Stream, Binary Stream and Record mode)			
Data set access (Record mode)			
Portable text file processing (HFS)			
Portable text file processing (data sets)			
VSAM data set access (KSDS, ESDS, RRDS)			
HFS access			

Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

MVS console communication

- **MVS console commands**
  - **Start:** */s jobName,commands*
  - **Modify:** */f jobName,commands*
  - **Stop:** */p jobName*

- **Code:**

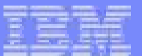
```
MvsConsole.registerMvsCommandCallback(new MvsCommandCallback() {
         public void handleModify(String s) {

         }

         public void handleStart(String s) {

         }


         public boolean handleStop() {
             return true;
         }
     });
```



MVS console communication

- **Write To Operator (WTO) API available**

```
{
    MvsConsole.wto("CIE00001: Ready to accept commands.",0x0020, 0x4000);
}
```

```
Session A - [24 x 80]
File  Edit  View  Communication  Actions  Window  Help

  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF SYSLOG   1683.101 SYS1 SYS1 01/09/2007 0W    8110      COLUMNS  26 105
 COMMAND INPUT ===> _                                         SCROLL ===> CSR
19:05:37.29 JOB01919 00000090  $HASP373 WTO      STARTED - INIT 2    - CLASS A -
19:05:37.30 JOB01919 00000090  IEF403I WTO - STARTED - TIME=19.05.37
19:05:37.71          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 948 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.71 STC01920 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.75 STC01920 00000090  $HASP373 BPXAS    STARTED
19:05:37.75 STC01920 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:37.82          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 952 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.82 STC01921 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.86 STC01921 00000090  $HASP373 BPXAS    STARTED
19:05:37.86 STC01921 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:39.38 JOB01919 00000090 @CIE00001: Ready to accept commands.
19:05:39.44 JOB01919 00000090  IEF404I WTO - ENDED - TIME=19.05.39
19:05:39.44 JOB01919 00000090  $HASP395 WTO      ENDED
19:05:39.48          00000090  $HASP309 INIT 2    INACTIVE ******** C=AB
******************************** BOTTOM OF DATA ********************************
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
MA     a                                                                  05/021
Connected to remote server/host 9.155.67.226 using lu/pool FTS10030 and port 23
```

z/OS message processing

```
Display  Filter  View  Print  Options  Help
SDSF SYSLOG   1683.101 SYS1 SYS1 01/09/2007 OW    8110      COLUMNS  26 105
COMMAND INPUT ===>                                          SCROLL ===> CSR
19:05:37.29 JOB01919 00000090  $HASP373 WTO      STARTED - INIT 2    - CLASS A -
19:05:37.30 JOB01919 00000090  IEF403I WTO - STARTED - TIME=19.05.37
19:05:37.71          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 948 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.71 STC01920 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.75 STC01920 00000090  $HASP373 BPXAS    STARTED
19:05:37.75 STC01920 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:37.82          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 952 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.82 STC01921 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.86 STC01921 00000090  $HASP373 BPXAS    STARTED
19:05:37.86 STC01921 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:39.38 JOB01919 00000090 @CIE00001: Ready to accept commands.
19:05:39.44 JOB01919 00000090  IEF404I WTO - ENDED - TIME=19.05.39
19:05:39.44 JOB01919 00000090  $HASP395 WTO      ENDED
19:05:39.48          00000090  $HASP309 INIT 2    INACTIVE ******** C=AB
******************************** BOTTOM OF DATA ********************************
 F1=HELP     F2=SPLIT    F3=END      F4=RETURN   F5=IFIND    F6=BOOK
 F7=UP       F8=DOWN     F9=SWAP     F10=LEFT    F11=RIGHT   F12=RETRIEVE
                                                                         05/021
```

SYSLOG

WTO

MPF

SSI

DB2

JES

…

z/OS
subsystems

Simple Automation

IF MSGID = 'IFB040I'
THEN EXEC(CMD('MVS S CLRLOG') ROUTE(ONE AUTO1));

Complex Automation

Tivoli System Automation

/f MyServer,clearlog

Java and z/OS console communication

Java based server

1) WTO “My log is full, stopping my work”

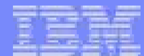
4) /F JSERV,CONTINUE

System automation

2) /S CLRLOG

Log cleaner

3) WTO “Finished!”



Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

Security interfaces in the IBM z/OS Java SDK

- **Java Cryptography Extension (IBMJCE)**
  - **Java Cryptography Extension in Java 2 Platform Standard Edition, Hardware Cryptography (IBMJCECCA)**
- **Java Secure Sockets Extension (IBMJSSE)**
- **Java Certification Path (CertPath)**
- **Java Authentication and Authorization Service (JAAS)**
- **Java Generic Security Services (JGSS)**
- **SAF interfaces**

SAF interfaces

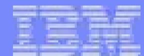
- Check to see if the Security Server or a specific security server class is active
- Extract the userid in effect for the current running thread
- Check the userid in effect for access rights to a resource
- Check if a userid is a member of a group
- Change a user's password
- Authentication
  - Authenticate a userid and password
- Authorization
  - Authorize a userid against resources
- Administration API:
  - … Will come soon!

Why should I use those security APIs?

- **Give your Java application a unique security add-on value**
- **Exploitation of hardware cryptography features**
  - **Enables cost savings**
  - **FIPS certified security**
  - **Faster encryption**
- **Certificate management with RACF transparently done in the background**
- **RACF/SAF**
  - **Enables your application in an easy way to integrate into the z/OS security infrastructure**

Ever hacked a mainframe application?

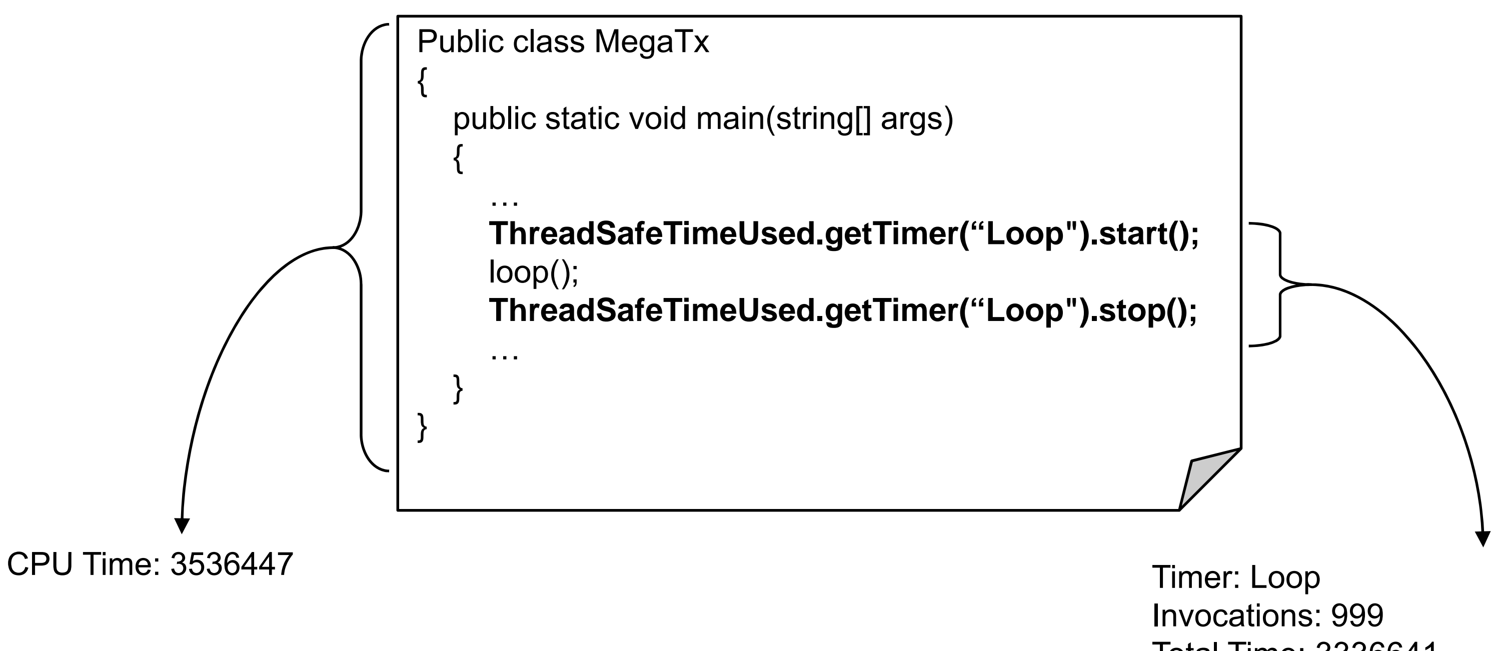
Whenever your application uses encryption like SSL, it automatically can make use of these platform advantages!



Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

Examining SMF accounted CPU time: Stand-alone Java programs



CPU Time: 3536447

Timer: Loop
Invocations: 999
Total Time: 3336641
Average Time: 3339
Minimum Time: 1372
Maximum Time: 293959
Overhead: 2 (Calculated but not removed)

Reference: IBM Redbook SG24-7177-00 (Java Stand-alone Applications on z/OS, Volume I
http://www.redbooks.ibm.com/abstracts/sg247177.html

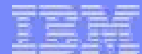
Examining SMF accounted CPU time: WebSphere z/OS

- **SMF 120 records provide CPU usage at the method level**

- **CPU Time service (WSC program) can be used for your own detailed measurements**

- **Use the method SMFJActivity.obtainTotalCpuTimeUsed() in pmi.jar**
  - **WAS V5:** ***http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD101339***
  - **WAS V6:** ***http://www-03.ibm.com/support/techdocs/atsmastr.nsf/fe582a1e48331b5585256de50062ae1c/8c027cb74803879a8625703f00492f84?OpenDocument***

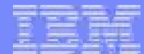
- **SMF Browser Plug-in available which adds summary report based on**
  - **each J2EE server instance, servlet, JSP, EJB, and method from the SMF type 120 records**
  - **See** ***http://www-03.ibm.com/support/techdocs/atsmastr.nsf/fe582a1e48331b5585256de50062ae1c/963f24f539c6cb7c86256db1005cc375?OpenDocument***



SMF API

- There are existing C API for SMF
- This API can be wrapped via JNI
  - You can use this API to write SMF records that write CPU time on a user level instead on application level for accounting reasons
  - You can also use it for writing data concerning access violations

→ A unique added value that could make your Java program a real mainframe application!



Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

Data sharing – What's does it mean?

Data
MVS Datasets (PDS/PDSe), Files, HFS, VSAM, Byte IRLM, CICS/DB2,…

Storage
Caches, Locks, data, structures,…

Services
RACF, JES,…

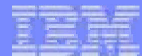
GRS, XCF

Data
MVS Datasets (PDS/PDSe), Files, HFS, VSAM, Byte IRLM, CICS/DB2,…

Storage
Caches, Locks, data, structures,…

Services
RACF, JES,…

Data sharing = Concurrent access while maintaining integrity

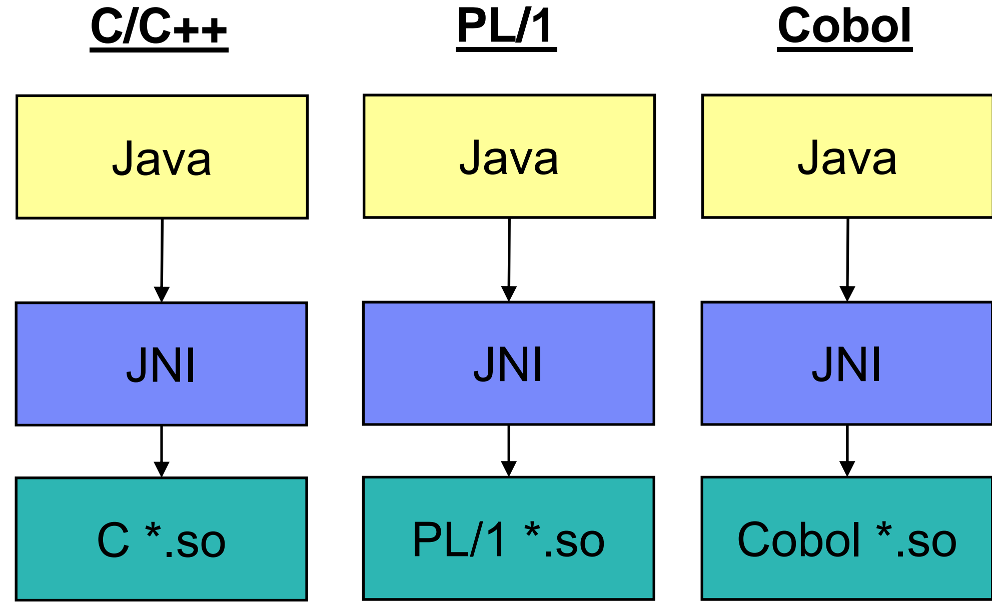


Agenda

- z/OS data access
  - MVS datasets
  - VSAM
  - HFS files
- MVS console communication
  - MVS commands
  - WTO
- Automation, event monitoring
- Security interfaces
  - Racf
  - crypto features
- SMF
- Data sharing
- JNI for C, C++, Cobol, PL/1, Assembler

JNI – Accessing C, C++, PL/1, Cobol

- **Write Java Code**
- **Compile Java code with javac.exe**
- **C/C++: Run javah.exe → *.h is generated**
- **Write native code, compile it and link it into a shared library**



- **You can also call explicitly a JVM From C/C++, Cobol and PL/1!**

Note: JNI is a wonderful thing, but treat it with care! It might be dangerous to your JVM…

Assembler access from Java

Java source code

1) Compile with javac

2) Run javah

C header file

3) Develop your C source code

C source code

Assembler source code

4) Compile as dll

5) Compile as dll

C *.o

Assembler *.o

6) Link object files into dll

Java *.class file

7) Run your Java appl.

lib*.so dll

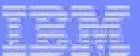
Reference: IBM Redbook SG24-7177-00 (Java Stand-alone Applications on z/OS, Volume I
http://www.redbooks.ibm.com/abstracts/sg247177.html

Platform dependant vs. platform independent

- **Writing platform dependant Java code does not seam reasonable at first….**
- **… But there is a solution:**

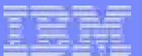
```
public static void main(String[] args) {
   if (args[1].equals("MVS")) {
         System.out.println("Cool, it is a z/OS system!");
         //z/OS specific code
         …
   }
}
```

→ **Having platform specific code within the same common code base can be a real add-on value!**



Summary

- **The access of native z/OS features like data and other languages is possible without any problem and can enhance your application concerning security and platform integrity**
- **Using Java for platform dependant coding is not a problem if you use it as an additional enrichment to your application!**



Questions

?