S16614: Practical Experiences about COBOL Programming. Make SOA Possible in batch COBOL

Tom Ross  IBM

March 5, 2015
Title: Practical experiences about COBOL programming. Make SOA possible in COBOL

- Introduction
- Invoking web services in IMS, CICS and WAS
- ‘Calling’ Java from COBOL
- Example from COBOL Programming Guide
- Our ‘simple’ solution
- Recommended approach
- Hints and tips

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
Many applications are being rewritten as Web Services
New applications are often written as Web Services
These parts can be combined into new applications
In some cases, ‘old’ applications need to use these new forms of applications
  – Invoke a web service anywhere instead of just calling a sub program in my z/OS region!
Some solutions are available…. 
IMS Enterprise Suite SOAP Gateway

• IBM® IMS™ Enterprise Suite SOAP Gateway is an XML-based solution that enables your IBM IMS applications to communicate outside the IMS environment using SOAP, without requiring changes to your business logic. The solution helps you modernize and gain more value from your IMS assets, and is available at no cost.

• IMS Enterprise Suite SOAP Gateway provides these features and benefits:
  – IMS applications can provide and request web services regardless of platform, environment, application language or programming model.
  – Client applications, such as Microsoft .NET and Java, can submit SOAP requests into IMS to drive the business logic of your COBOL or PL/I applications.
  – IMS applications can send business event data to business event processing and monitoring engines such as IBM WebSphere® Business Events and IBM Business Monitor.
IBM® CICS Transaction Gateway (CICS TG), a market-leading Enterprise connector, is production proven by over a thousand customers as a high performing, security-rich, and scalable method of service-oriented architecture (SOA) access to CICS, which:

- Delivers Java Enterprise Edition (JEE) standards-based access to CICS applications, while requiring minimal changes to CICS and usually no changes to existing CICS applications
- Provides quick and easy connector access to CICS applications from a wide variety of environments, including Java, C/C++, Microsoft .NET, and COBOL run times
- Allows the reuse of existing CICS applications as services in comprehensive and sophisticated JEE and web services solutions hosted on powerful application servers such as WebSphere Application Server
WebSphere Application Server for z/OS

- IBM® WebSphere® Application Server for z/OS® helps provide availability and security while reducing costs for business critical applications. It uses the full capabilities of IBM System z® and IBM z/OS and enables: prioritized workload management, advanced transactional integrity, horizontal and vertical scalability and data and workload co-location.

- WebSphere Application Server for z/OS helps you:
  - **Optimize developer productivity** and provide continuous availability using System z features and Liberty profile, a streamlined runtime environment for web application deployments.
  - **Deploy and manage applications and services** to meet the demands of your growing business.
  - **Improve operations and resiliency** through advanced application availability, elasticity and quality of service.
  - **Provide rapid, scalable and secure enablement of web, cloud and mobile access to z/OS assets using IBM WebSphere Liberty z/OS Connect.**
  - **Enhance security and control** using integrated management and administrative tools.

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
What about z/OS batch?

- Typically no J2EE server available
- Java can still do SOME things more easily than COBOL can
  😊
- HTTP calls!
- But my batch programs are COBOL!
- ‘Call’ Java from batch COBOL on z/OS?’
What about z/OS batch?

• What we wanted to do:

  **COBOL**
  **QSAM**
  **appl**
  **JES**

  ?

  **RESTful Web Service**

  **WEB HTTP server**
What about z/OS batch?

- What we tried to do:

  - COBOL QSAM appl
  - Java With HTTP call
  - RESTful Web Service
  - WEB HTTP server

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
What about z/OS batch?

- More detail about what we tried to do:
What about z/OS batch?

- This presentation will focus on these parts

OO COBOL w/INVOKE → Java With HTTP call

INVOKER

HTTP CALL

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
‘Calling’ Java from COBOL

• Change the mindset
  – No programs in Java, no CALLs
  – You CAN Invoke a **Method** in a **Java class**

• Let’s start with the example in COBOL Programming Guide!

• Chapter 16, TSTHELLO example in section: *Example: compiling, linking, and running an OO application using JCL*

• Well, I thought it would be easy…
Problems found in PG example

- Copying text from .pdf to ISPF EDIT gave me non-editable chars for apostrophes
  - Or the apostrophes did not get copied in at all
- Executable (SYSLMOD) could not be in temp dataset!
- Bad format of run-time options
- Wrong attribute on STEPLIB
- Extraneous comma in JAVAOUT DD
- Invalid indentation for JAVAERR DD
- Missing .: in ENV file
Problems found in PG example

- Executable (SYSLMOD) could not be in temp dataset

```c
//SYSLMOD DD
DSN=&GOSET(TSTHELLO),DISP=(MOD,PASS),UNIT=VIO,
// SPACE=(CYL,(1,1,1)),DSNTYPE=LIBRARY
```

- I could not get this to work with COBOL V5!
Problems found in PG example

This is what I got when I tried temp PDSE load library:

```plaintext
$HASP373  TSTHELLO  STARTED  -  WLM  INIT  -  SRVCLASS  PRDBATHI  -  SYS  SA0W

HTRT01I  CPU (Total)  Elapsed  CPU
HTRT02I  Jobname  Stepname  RC  I/O  hh:mm:ss.th  hh:mm:ss.th  hh:mm:ss.th
HTRT03I  TSTHELLO  COMPILE  00  9972  00.05  01.77  00.05
HTRT03I  TSTHELLO  LKED  00  460  00.02  00.23  00.02

IEW4009I  FETCH  FAILED  FOR  MODULE  TSTHELLO  FROM  DDNAME  STEPLIB  BECAUSE  OF  AN  I/O  ERROR.

CSV031I  LIBRARY  SEARCH  FAILED  FOR  MODULE  TSTHELLO,  RETURN  CODE  24,  REASON  CODE  2706043E,  DDNAME  STEPLIB
CSV028I  ABEND806-2C  JOBNAME=TSTHELLO  STEPNAME=GO
IEA995I  SYMPTOM  DUMP  OUTPUT  938

SYSTEM  COMPLETION  CODE=806  REASON  CODE=0000002C
```

- I changed to a permanent dataset and it worked fine!
Problems found in PG example

• Bad format of run-time options
  
  ```
  //GO EXEC PGM=TSTHELLO,COND=(4,LT,LKED),
  //
  PARM='//ENVAR("_CEE_ENVFILE=/u/username/ootest/tsthello/ENV")
  // POSIX(ON)
  XPLINK(ON)'
  ```

• Should be:

  ```
  //GO EXEC PGM=TSTHELLO,COND=(4,LT,LKED),
  //
  PARM='//ENVAR("_CEE_ENVFILE=/u/username/ootest/tsthello/ENV")
  // POSIX(ON) XPLINK(ON)'
  ```
Problems found in PG example

• Wrong attribute on STEPLIB

  //STEPLIB DD DSN=*.LKED.SYSLMOD,DISP=SHR

• Should have been (for temp dataset):

  //STEPLIB DD DSN=*.LKED.SYSLMOD,DISP=PASS
Problems found in PG example

- Missing PATHOPTS for JAVAOUT DD

//JAVAOUT DD PATH=’/u/userid/ootest/tsthello/javaout’,

- Should have been:

//JAVAOUT DD PATH=’/u/userid/ootest/tsthello/javaout’,
//       PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//       PATHMODE=(SIRUSR,SIWUSR,SIRGRP)

- Result? No println output!
Problems found in PG example

• Invalid indentation for JAVAERR DD

```//JAVAERR DD PATH='/u/userid/ootest/tsthello/javaerr',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
```

• Should have been:

```//JAVAERR DD PATH='/u/userid/ootest/tsthello/javaerr',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
```
Problems found in PG example

Environment variable settings file, ENV

PATH=/bin:/usr/lpp/java/J5.0/bin.
LIBPATH=/lib:/usr/lib:/usr/lpp/java/J5.0/bin:/usr/lpp/java/J5.0/bin/j9vm
CLASSPATH=/u/userid/ootest/tsthello

Should be:

CLASSPATH=.:/u/userid/ootest/tsthello
Our ‘simple’ solution

- Batch program processing QSAM data
- Needs actuarial information from Internet Web Service
  - In our example, we used a simple system status Web Service instead 😊
- Make DYNAMIC call to COBOL Web Service wrapper
- Web service wrapper uses INVOKE of Java
- Java will make HTTP call to Web Service using Apache
- Return info to Java, then to COBOL wrapper, then to Batch application
- Is it do-able?
Our ‘simple’ solution

- Changes to batch application?
  - Add dynamic CALL to COBOL wrapper
  - Add runtime options:
    - Must run with XPLINK runtime option
    - Must also have ENVAR set

```
//GO EXEC PGM=CALLINVK, COND=(4, LT, LKED),
//
// PARM='/ENVAR("_CEE_ENVFILE=/home/tmross/Java/ENVS")
//                POSIX(ON) XPLINK(ON)'
```

- If no pointer to ENV file with LIBPATH to JVM, then:
If no pointer to ENV file with LIBPATH to JVM, then:

COBOL program CALLINVK entered
CEE3501S The module libjvm.so was not found.

From entry point GetJVMPtr at compile unit offset
+000000B2 at entry offset +000000B2 at address 26EDF6F2.
CEE3DMP V2 R1.0: Condition processing resulted in the unhandled condition.

06/02/14 10:06:08 PM
Our ‘simple’ solution
COBOL wrapper for getting to Java
First: TSTHELLO example from PG

cbl dll,thread,pgmname(longmixed)
Identification division.
Program-id. "TSTHELLO" recursive.  <* Upper case

Environment Division.
Configuration Section.
Repository.  <* Case of class
name

Class HelloJ is "HelloJ".  <* must match class
name

Data Division.
Procedure Division.
Display "COBOL program TSTHELLO entered"

Invoke HelloJ "sayHello"

Display "Returned from java sayHello to TSTHELLO"

Goback.
End program "TSTHELLO".
Our ‘simple’ solution
COBOL wrapper for getting to Java
First: TSTHELLO example from PG

• This was what we ‘wrapped’: HelloJ.sayHello
• Hello in System.out.println

class HelloJ {
    public static void sayHello() {
        System.out.println("Hello World, from Java!");
    }
}

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
Our ‘simple’ solution
COBOL wrapper for getting to Java
First: TSTHELLO example from PG

- Job output:

**** END OF MESSAGE SUMMARY REPORT ****

COBOL program TSTHELLO entered
Returned from java sayHello to TSTHELLO

- Contents of javaout:
Our ‘simple’ solution
COBOL wrapper for getting to Java
gradually add more: HelloString

cbl dll,thread,pgmname(longmixed)
Program-id. "INVKHSTR" recursive.
Environment Division.
Configuration Section.
Repository.
    Class HelloString is "HelloString"
    Class jstring is "jstring".
Data Division.
Working-Storage Section.
77 url Pic X(50) Value
    z'Tom'.
77 jstring1 Object Reference jstring.
77 jstring2 Object Reference jstring.
77 rc Pic s9(9) Comp-5.
77 ptr Pointer.
77 jstringlen Pic s9(9) Comp-5.
77 Returned_string Pic X(50).
Our ‘simple’ solution

COBOL wrapper for getting to Java gradually add more: HelloString

Procedure Division.
Display "COBOL program INVKHSTR entered"

****************************************************************************
**
* Convert string into Java string object
****************************************************************************
**
Call "NewStringPlatform"      /* Case matters
using by value JNIEnvPtr
   address of Url                /* input
   address of jstring1          /* output
   0

   returning rc
   If rc Not = zero Then
      Display "Error occurred creating jstring object"
      Stop run
   End-if

name
Invoke HelloString "sayHello"        /* Same method name!
using by value jstring1
   returning jstring2

Display "Returned from java sayHello to INVKHSTR"
**Our ‘simple’ solution**

COBOL wrapper for getting to Java gradually add more: HelloString

*************************************************** ***********
* Convert Java string object back into string - get length
*************************************************** ***********

Set ptr To address of jstringlen  <- Get output addr
Call "GetStringPlatformLength"  <- Case matters
    using by value JNIEnvPtr
    jstring2 <- input
    ptr <- output

returning rc
Display "Returned from GetStringPlatformLength"
If rc Not = zero Then
    Display "Error retrieving len of jstring object"
    Stop run
Else
    Display "The length of returned string is:" jstringlen
End-if
Our ‘simple’ solution
COBOL wrapper for getting to Java gradually add more: HelloString

*************************************************************************
* Convert Java string object back into string – get string                *
*************************************************************************
Call "GetStringPlatform"        <*> Case matters
   using by value JNIEnvPtr
   jstring2 address of Returned_string
   length of Returned_string
   rc
If rc Not = zero Then
   Display 'Error occurred getting string ' \\
      'from jstring object'
   Stop run
End-if
Display 'sayHello returned: ' \\
   Returned_string(1:jstringlen)
Display "About to leave INVKHSTR"
   Goback.
End program "INVKHSTR".
Our ‘simple’ solution
COBOL wrapper for getting to Java gradually add more: HelloString

• This is newer version of the Java: HelloString.sayHello
• Hello in println and in return value

class HelloString {
    public static String sayHello(String name) {
        System.out.println("Hello, " + name);
        return "Hello, " + name + " from Java!";
    }
}
Our ‘simple’ solution
COBOL wrapper for getting to Java gradually add more: HelloString

• Job output:
  **** END OF MESSAGE SUMMARY REPORT ****
  COBOL program INVKHSTR entered
  Returned from java sayHello to INVKHSTR
  Returned from GetStringPlatformLength
  The length of returned string is: 0000000022
  sayHello returned: Hello, Tom from Java!
  About to leave INVKHSTR

• Contents of javaout:

  EDIT /home/tmross/Java/javaout
  Command ==> ____________________________

  000001 Hello, Tom
  000000

  Complete
Debugging JNI calls is hard!

- A parm with no storage usually gets an 0C4, but with JNI services...

<table>
<thead>
<tr>
<th>HTRT01I</th>
<th>CPU (Total) Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTRT02I</td>
<td>Jobname Stepname ProcStep RC I/O hh:mm:ss.th hh:mm:ss.th</td>
</tr>
<tr>
<td>JVMDUMP032I</td>
<td>JVM requested System dump using 'TMROSS.JVM.TDUMP.INVKREST.D140519.T182116' in response to an event</td>
</tr>
<tr>
<td>IGD101I</td>
<td>SMS ALLOCATED TO DDNAME (SYS00007) 925</td>
</tr>
<tr>
<td>IGD104I</td>
<td>DSN (TMROSS.JVM.TDUMP.INVKREST.D140519.T182116 ) STORCLAS (OS390) MGMTCLAS (STANDARD) DATACLAS ( )</td>
</tr>
<tr>
<td>IGD104I</td>
<td>TMROSS.JVM.TDUMP.INVKREST.D140519.T182116 RETAINED, DDNAME=S</td>
</tr>
<tr>
<td>JVMDUMP032I</td>
<td>JVM requested Java dump using '/home/tmross/javacore.20140519.182116.33558008.0002.txt' in response to an event</td>
</tr>
<tr>
<td>BPXM023I</td>
<td>(TMROSS) 929</td>
</tr>
<tr>
<td>JVMDUMP032I</td>
<td>JVM requested Snap dump using '/home/tmross/Snap.20140519.182116.33558008.0003.trc' in response to an event</td>
</tr>
</tbody>
</table>

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
Debugging COBOL to Java is hard!

- What happens when the JVM cannot find your called Java method?
  - For example, incorrect location of Java package in CLASSPATH...
  - `.:` comes first in CLASSPATH
  - Name the `.jar` package, not just the directory
  - If you make a mistake...

- And I did not have a main method!

**** END OF MESSAGE SUMMARY REPORT ****

Exception in thread "main"

*************************** BOTTOM OF DATA ***
Debugging COBOL to Java is hard!

• Name the .jar package, not just the directory
  – Two things here
    1. ‘.’ For current directory
    2. Directory that contains hello.jar
      • hello.jar contains HelloJ.sayHello and HelloString.sayHello

CLASSPATH=.:./home/tmross/Java/hello.jar
Our ‘simple’ solution
COBOL wrapper for getting to Java
finally add: invokeGETAsXML

- We wrote a Java method
  invokeGETAsXML
- It makes an HTTP call using Apache
- The HTTP server returns a system status
  - In XML or JSON (we chose XML)
- Pass a url from COBOL to Java for the
  HTTP server
Our ‘simple’ solution
COBOL wrapper for getting to Java
finally add: invokeGETAsXML

77 Url Pic X(60) Value
   z'http://rdpweb01.ibm.com:7999/ZOS/resserv/status'.

* Followed by the same calls to JNI services as
* as earlier to convert Url string to jstring1

   Invoke CobRest 'invokeGETAsXML'
       using by value jstring1
       returning jstring2
Structure of the sample project

Invoke Apache HttpClient from Java on z

Our sample methods to invoke from COBOL

Our sample convenience and utility methods that invoke Apache HttpClient

Apache HttpComponent libraries

IBM JSON4J libraries

More Apache HttpComponent libraries
Simple REST interface

```java
public class CobRest {

    public static void main(String[] args) {
        System.out.println("Entered main...");
        String respBody;
        try {
            respBody = invokeGET("http://rdpweb01.torolab.ibm.com:7999/ZOS/resserv/status");
            System.out.println("Response body as JSON: " + respBody);

            System.out.println("-----------------------------------------------");

            respBody = invokeGETAsXML("http://rdpweb01.torolab.ibm.com:7999/ZOS/resserv/status");
            System.out.println("Response body as XML: " + respBody);

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("Exited main...");
    }
}
```

Invoke GET on a sample service that returns another server’s status (UP or DOWN) in JSON format

Same service but returning result in XML format
What is needed for Java on z/OS?
Same as on other platforms!

- File system - HFS / zFS
- Where is Java installed? What level is installed?
- Some handy environment variables
- RDz – Makes Java easier on/for z/OS
- Java Basics
  - To compile – javac
  - To execute the byte code - java
The Environment setup for Java – things to know

• Where is Java Installed?
  – `JAVA_HOME=/usr/lpp/java/IBM/J7.0`
  `export JAVA_HOME`

• Where is the Java application executable?
  – `CLASSPATH=.:home/tmross/Cobrest.jar`
  `export CLASSPATH`

• Where are the tool executables?
  – `PATH=.:usr/lpp/java/IBM/J7.0/bin`
  `export PATH`
Writing, building, execution of Java 7 – similar to other platforms

- Java application (CobRest.java)
- Use the Java Perspective in RDz, create a project and write the Java application using all of the Eclipse support
- Export the jar file (external jar)
- Setup a launch configuration to test
  - Run … -> Host Java Application (New)
  - Fill in details, include the CLASSPATH and any environment variables
Writing, building, execution of Java 7 – similar to other platforms

• Now you are ready to test the application – a few ways to do this in RDz:
  – From the Java Perspective
    • Run … -> Host Java Application
      (select the launch configuration you setup)
  – From the zOS Perspective
    • Launch the USS Shell
    • Set the CLASSPATH, TZ, other env vars
      (I use a shell script)
    • java <thePackageName>
Result of running CobRest.java in RDz

```console
<terminated> TOROLABW CobRest (2) [Host Java Application] CobRest

Entered main...
Response body as JSON: [{"host":"mvs099.rtp.raleigh.ibm.com:6768","status":"DOWN"}]
-----------------------------------------------
Response body as XML: <hosts> <host ip=mvs099.rtp.raleigh.ibm.com:6768 status=DOWN/> </hosts>
Exited main...
```
Our ‘simple’ solution
COBOL wrapper for getting to Java finally add: invokeGETAsXML

My ENV file in: /home/tmross/Java/ENVS

• First attempt we put all packages in CobRest.jar
• So, my ENVS file looked like this:

PATH=/bin:/usr/lpp/java/IBM/J7.0/bin
LIBPATH=/lib:/usr/lib:/usr/lpp/java/IBM/J7.0/bin:/usr/lpp/java/IBM/J7.0/bin/j9vm
CLASSPATH=../home/tmross/Java/CobRest.jar
COBJVMINITOPTIONS=-Xdump:ceedump -Xcheck:jni -Xjit:verbose

• Explanation of JVM options:

-Xdump:ceedump  *> Tells the JVM to put out a CEEDUMP
-Xcheck:jni     *> Use to investigate possible JNI problems
-Xjit:verbose   *> Enables JIT tracing
-XCheck:jni:trace *> Enables JNI call tracing
Result of running CobRest.java from COBOL return of “Debugging Java is hard!”

- With the Apache and other .jar files in CobRest.jar we got abort in JVM when calling JNI services to convert returned string object to string
- We used the extra debugging options for JVM and pulled in a Java expert to diagnose the problem
- If we commented out the JNI GetString* calls, the job ended with no clue that there had been an exception in the Java code!
Result of running CobRest.java from COBOL

- With JNI trace option set on, we got this:

HTRT02I Jobname Stepname ProcStep RC I/O hh:mm:ss.th
JVMJNCK028E JNI error in GetStringLength: This function cannot be called when an exception is pending
VMJNCK080E Error detected in the outermost frame of an attached thread
JVMJNCK024E JNI error detected. Aborting.
HTRT03I INVKREST GO 1111 24755 00.41
Result of running CobRest.java from COBOL

- `-Xcheck:JNI:trace` was what gave us the information

```java
  at java.lang.J9VMInternals.verifyImpl(Native Method)
  at java.lang.J9VMInternals.verify(J9VMInternals.java:94)
  at java.lang.J9VMInternals.initialize(J9VMInternals.java:171)
  at CobRest.invokeGETAsXML(CobRest.java:65)

  at java.net.URLClassLoader.findClass(URLClassLoader.java:599)
  at java.lang.ClassLoader.loadClassHelper(ClassLoader.java:760)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:728)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:325)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:707)
```

Complete your session evaluations online at www.SHARE.org/Seattle-Eval
Result of running CobRest.java from COBOL

- It turns out we could not put all of the jar files in CobRest.jar.
- We separated them out, added the .jar paths to JENVS file:

```bash
PATH=/bin:/usr/lpp/java/IBM/J7.0/bin
LIBPATH=/lib:/usr/lib:/usr/lpp/java/IBM/J7.0/bin:/usr/lpp/java/IBM/J7.0/bin/j9vm
CLASSPATH=.:/home/tmross/Java/httpcore-ab-4.2.4.jar:
    /home/tmross/Java/commons-logging-1.1.2.jar:
    /home/tmross/Java/org.apache.httpcomponents.httpclient_4.2.3.jar:
    /home/tmross/Java/JSON4J.jar:
    /home/tmross/Java/httpcore-nio-4.2.4.jar:
    /home/tmross/Java/httpcore-4.2.4.jar:
    /home/tmross/Java/CobRest.jar
COBJVMINITOPTIONS=-Xdump:ceedump -Xcheck:jni
```
Result of running CobRest.java from COBOL

- Better, we got data back from the Web Service!
- But, we broke the Java compiler....

**** END OF MESSAGE SUMMARY REPORT ****

JVMJNCK001I JNI check utility installed. Use -Xcheck:jni:help for usage

COBOL program INVKREST entered

Unhandled exception

Type=Floating point error vmState=0x000565ff
J9Generic_Signal_Number=00040020 Signal_Number=00000008 Error_Value=000
Handler1=277155D8 Handler2=278145C8
Program_Unit_Name=./Profiler.cpp
Program_Unit_Address=27F86090 Entry_Name=TR_BranchProfileInfoManager::gR_Compilation*)
Entry_Address=27F86090

Method_being_compiled=java/util/zip/InflaterInputStream.read([BI)
Target=2_60_20140106_181350 (z/OS 02.01.00)
CPU=s390 (24 logical CPUs) (0x1000000000 RAM)

----------- Stack Backtrace -----------
Result of running CobRest.java from COBOL

- So, until we get the Java fix, we turned off profiling in JSENV...
- COBJVMINITOOPTIONS= -Xjit:disableInterpreterProfiling

**** END OF MESSAGE SUMMARY REPORT ****
JVMJNCK001I JNI check utility installed. Use -Xcheck:jni:help for usage
COBOL program INVKREST entered
Returned from Java invokeGETAsXML to INVKREST

Returned from GetStringPlatformLength
The length of returned string is:0000000070
invokeGETAsXML returned: <hosts> <host
   ip=mvs099.rtp.raleigh.ibm.com:6968 status=UP/> </hosts>
About to leave INVKREST

************ END OF MESSAGE SUMMARY REPORT ************
ISPF tip, it helped me a lot in this exercise!

I could avoid jumping back and forth from OMVS to ISPF.
ISPF tip, helped a lot in this exercise!
QUESTIONS?