#SHAREorg

SHARE

CICS and Java: How the JVM Server Transforms Java in CICS

Catherine Moxey
IBM

Thursday, February 7, 2013
Session Number 12446

SHARE in San Francisco 2013

Disclaimers

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. Any information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

The session and materials has been prepared by IBM or the session speaker and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Trademarks

Session Abstract

- CICS has for a long time provided a Java environment for application development. In recent releases of CICS the JVM Server has transformed CICS into a first-class hosting environment for Java. This session will provide a brief history of the development of the Java environment within CICS, followed by a detailed look at the capabilities offered by CICS version 5. In particular we will look at how the OSGi framework provides excellent lifecycle management of Java applications without having to restart the JVM Server, how Java applications can be eligible for zAAP offload thereby reducing the cost of a transaction, and how the JVM Server supports multiple concurrent transactions, reducing the storage requirements and the need for multiple JVM instances in a single region.

CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

A Java Analogy

CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

What is Java and what makes it different from traditional languages on z ?

Java is

- an object oriented,
- platform independent,
- broadly supported and prevalent
- state of the art language
- [and an Island in Indonesia]

- Something YOU should care about

Java is not

- something new and unreliable
- an error-prone language
- the only solution for good code
- something that only the distributed world should care about
- a composable workbox full of libraries

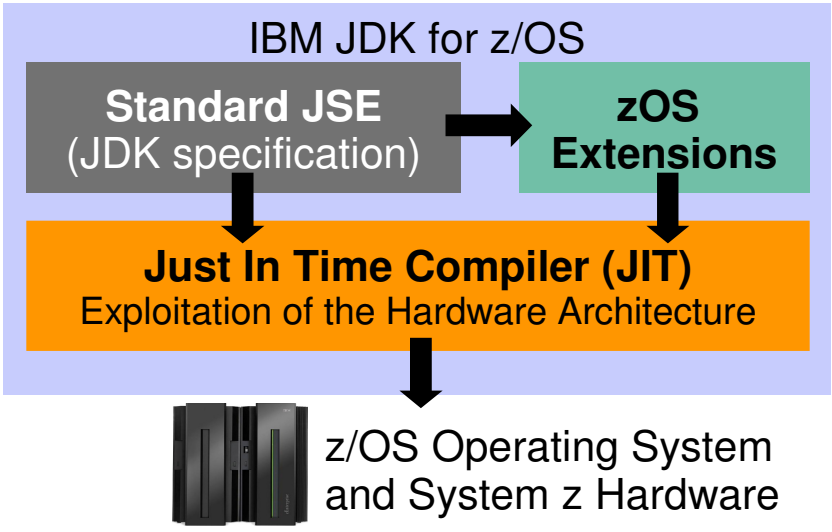
What makes Java different

Java differes from traditional languages like assembler, PL/I or COBOL:

- Java is not compiled into executable code, it is compiled to bytecode which runs in a virtual machine
- The virtual machine uses the just-in-time compiler (JIT) to execute the code
- Java is based on classes rather than programs
- Java uses a garbage collector, which removes unused objects from the storage
- Java development is often based on existing programming patterns (Design Patterns)
- Most of the built-in funtionality of Java is based on class libraries, that are part of the Java Runtime (JRE)
- Java contains a library for user interface development
- Java has type-safe variable declaration
- Java uses JNI for native system calls and JDBC for database calls

Java on z – why run here?

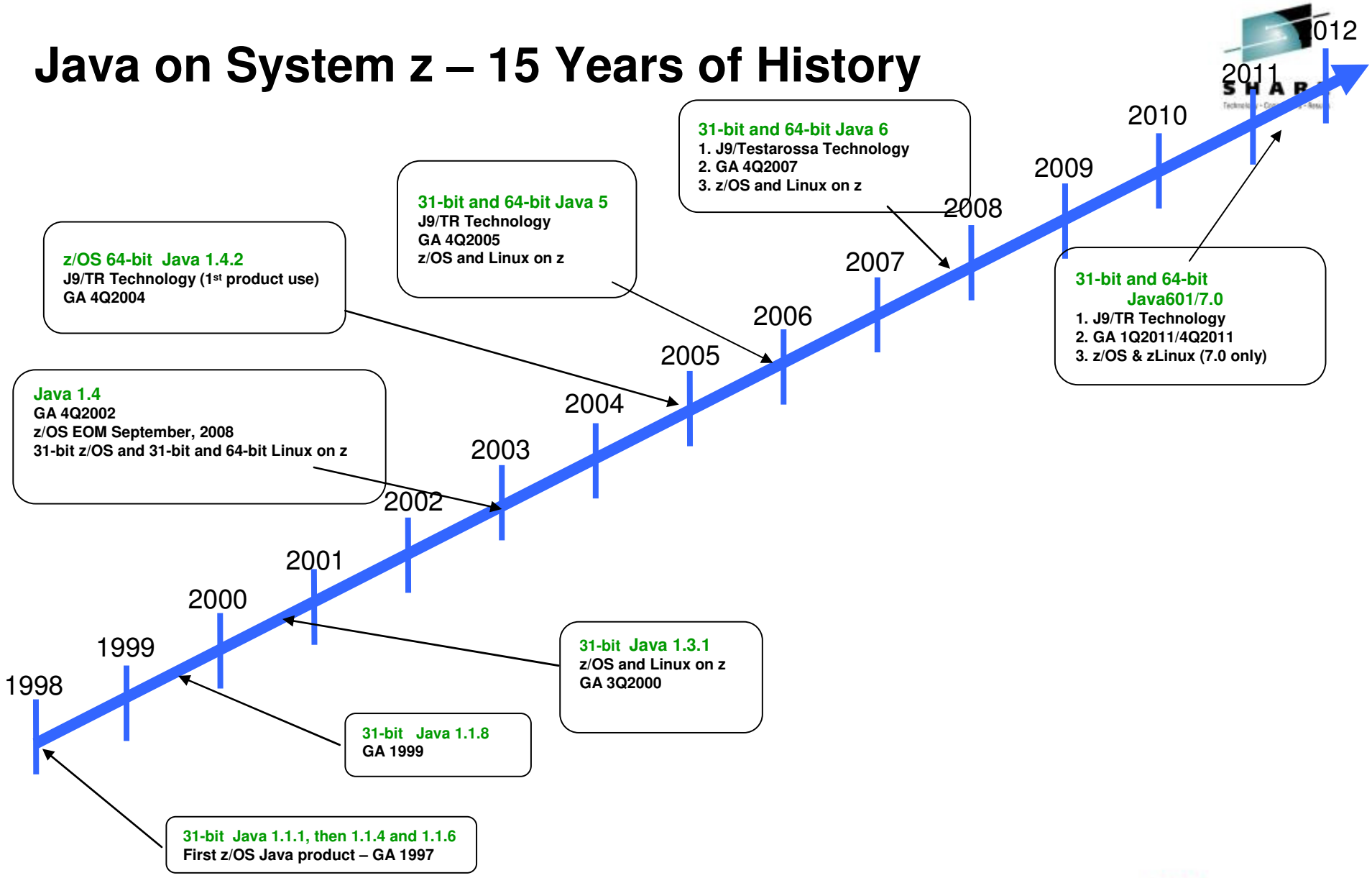
- IBM uses its own implementation of a JVM on mainframes that uses the underlying platform architecture
- The Java workload can be offloaded to a zAAP processor
- The Java logic can be a bridge between the mainframe and the distributed world
  - Java is a common language on many platforms so can help build dialog between departments
- Java is a language that is well known by many new professionals, so a good common ground when they start to develop for mainframe applications
- The language encourages good design and loose coupling of components (although does not ensure it!)



CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Java on System z – 15 Years of History



IBM J9 2.6 Technology Enhancements: System zEnterprise 196 and Java6.0.1/Java7

z196 and Java6.0.1/Java7: Engineered Together

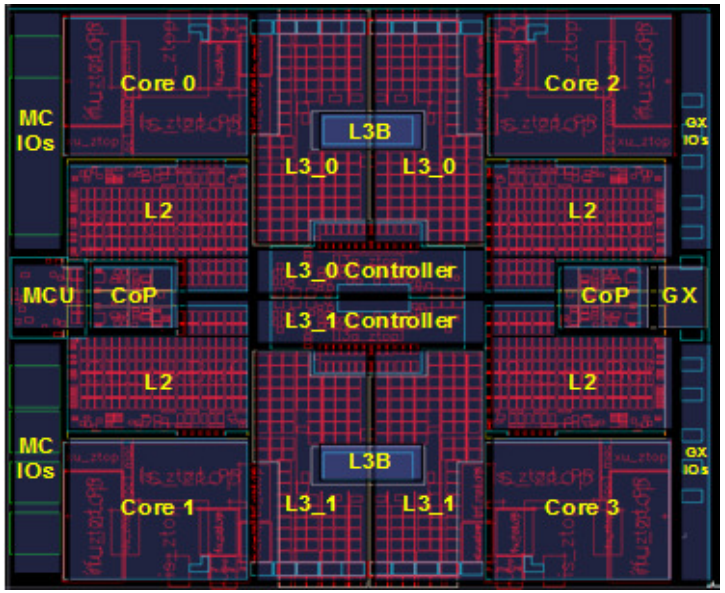
- Up-to 2.1x improvement to Java throughput
- Reduced footprint
- Tighter integration with z/OS facilities
- Improved responsiveness in application behavior

Hardware for Java

- New Out-Of-Order pipeline design
- New larger cache structure
- Higher clock speed (~5.2GHz)

70+ new instructions used by Java

- Register high-word facility
  - Facilitates use of upper 32-bits of registers
- Interlock facility update
  - Better Java concurrency
- Non-destructive operands
  - Reduce path-length
- Conditional-load/store
  - Remove expensive branches
- Instruction scheduler for Out-of-Order pipeline



zEC12 – Java as a workload-optimized system

Continued aggressive investment in Java on z

Significant set of new hardware features tailored and co-designed with Java

Hardware Transaction Memory (HTM)

Better concurrency for multi-threaded applications
eg. ~2X improvement to juc.ConcurrentLinkedQueue

Run-time Instrumentation (RI)

Innovative new h/w facility designed for managed runtimes
Enables new expanse of JRE optimizations

2GB page frames

Improved performance targeting 64-bit heaps

Pageable 1MB large pages using flash

Better versatility of managing memory

New software hints/directives

Data usage intent improves cache management
Branch pre-load improves branch prediction

New trap instructions

Reduce over-head of implicit bounds/null checks

*New **5.5 GHz** 6-Core Processor Chip*
***Large caches** to optimize data serving*
*Second generation **OOO design***

Up-to 45% improvement in throughput amongst Java workloads measured with zEC12

IBM SDK7 for z/OS Service Refresh3 (IBM Java 7 SR3) **–Xaggressive** command-line option enables a variety of new optimizations and zEC12 exploitations

Java 7.0

- Modularity
  - Strong, standard, language-integrated module interoperability and encapsulation
  - Easier re-use, higher security, clearer governance, improved reliability, lower costs, reduced maintenance and shorter downtimes
- New I/O
  - Meets the increasingly I/O intensive demands of data mining and analytics workloads
  - Significant performance and footprint gains from async I/O
- Concurrency Libraries
  - Exploit larger multi-core systems, such as next generation Power and System z, by providing better scalability, higher throughput and lower total cost of ownership from server consolidations
- Dynamic language support
  - Leverage the advantages of a single runtime for dynamic language applications written in PHP, Groovy, jRuby and jython
- Language improvements
  - Improved efficiency through simplified day-to-day programming tasks
  - Protect developer commitment to, and customer/ISV investment in, the Java ecosystem.

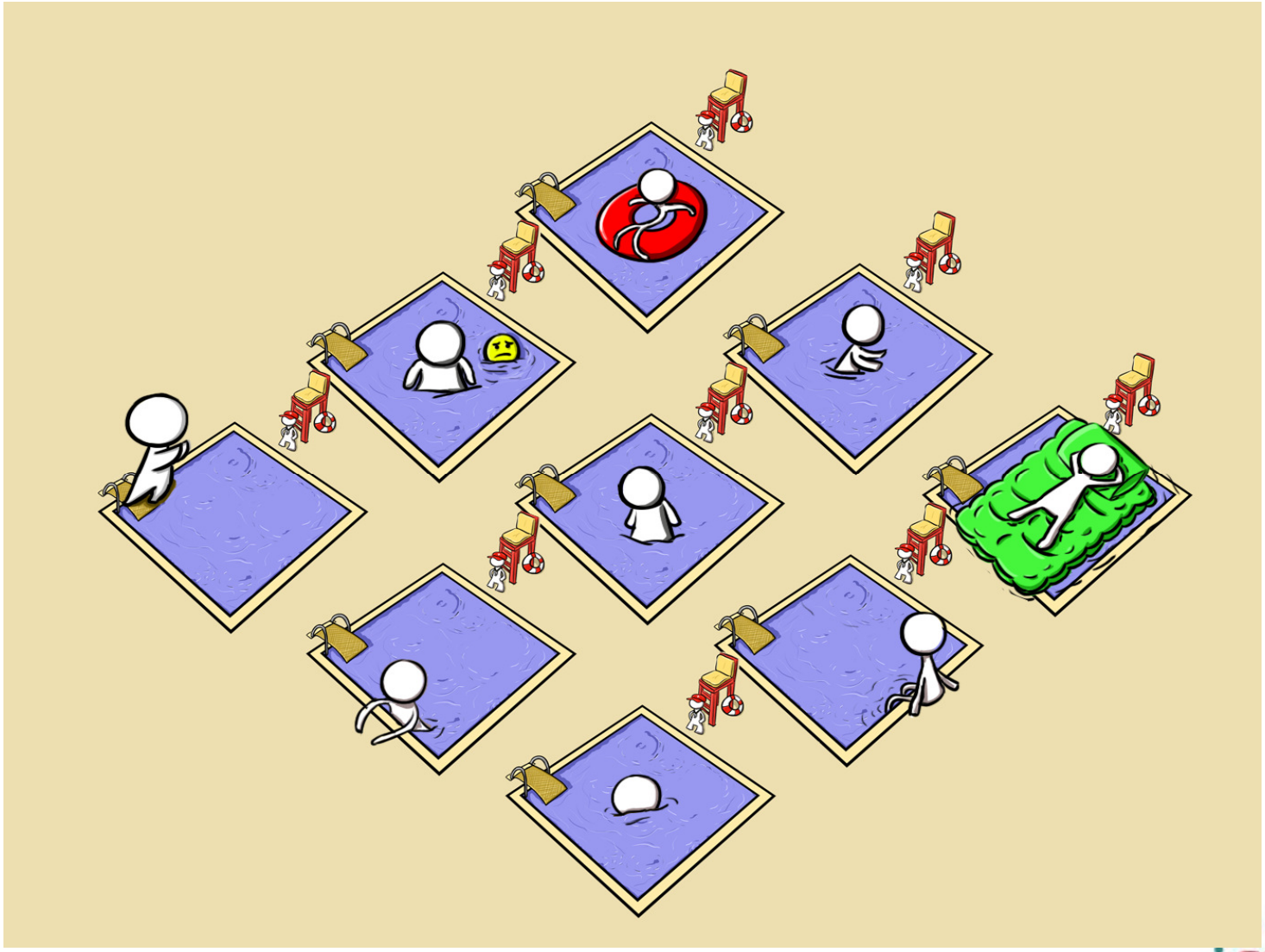
CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Overview of Java program support in CICS

- "Traditional" pooled JVMs
  - Multiple JVMs in a CICS region
  - Single-thread, program isolation
  - J8 (CICS Key) or J9 (User key) TCBs
  - MAXJVMTCBs in SIT
  - No JVM definition except in JVM profile via PROGRAM
  - EJB and CORBA support
- "New" JVM servers
  - Supports JCICS interfaces for CICS Java programs
  - Can have multiple JVM servers per region
  - Multi-threaded, up to 256 parallel tasks
  - Facilitates data-sharing between Java applications
  - Industry-standard
  - T8 TCBs
  - JVMSERVER and PROGRAM definitions required
  - Requires deployment as OSGi bundle within a CICS BUNDLE
  - No EJB or CORBA support

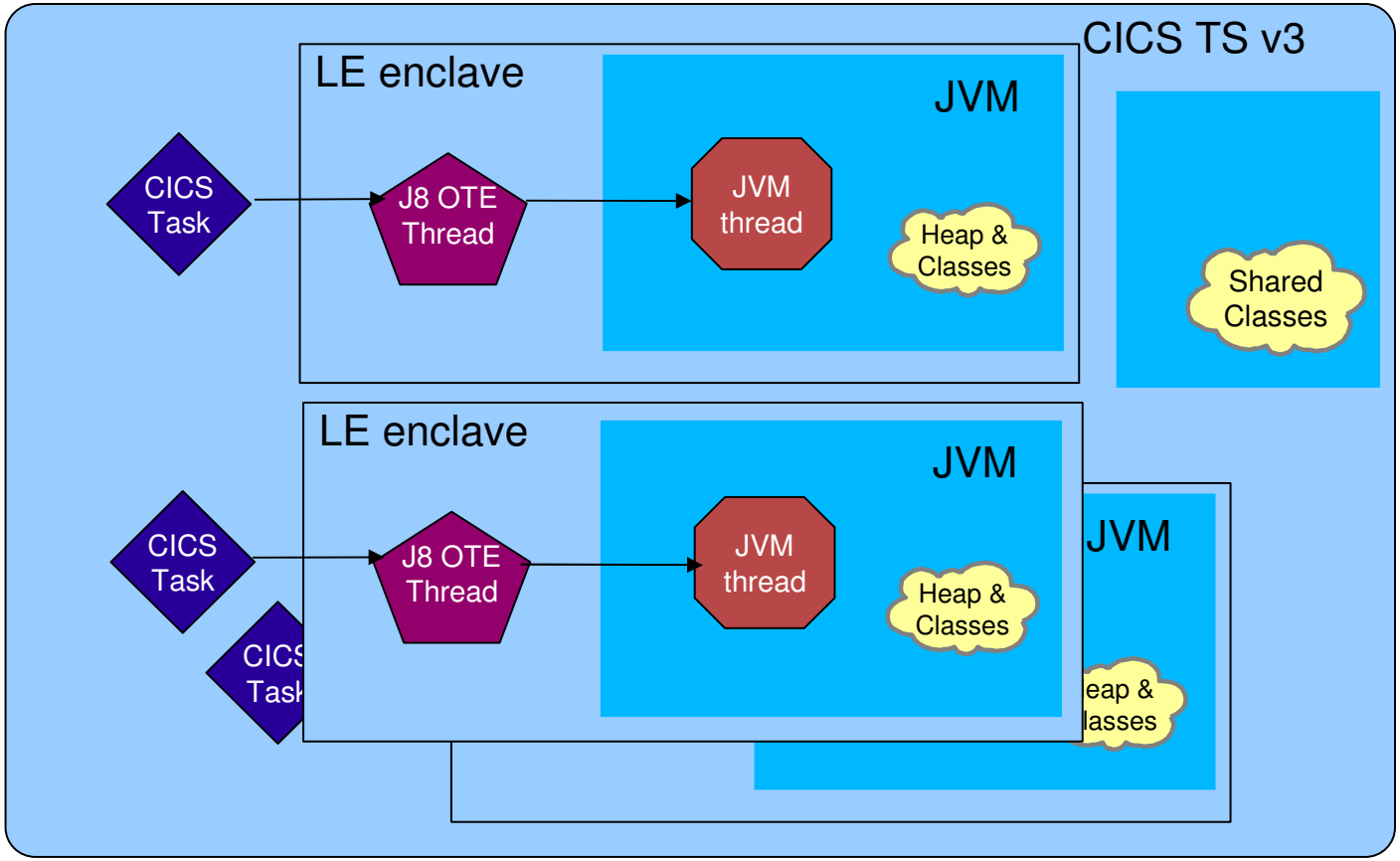
CICS Pooled JVM model



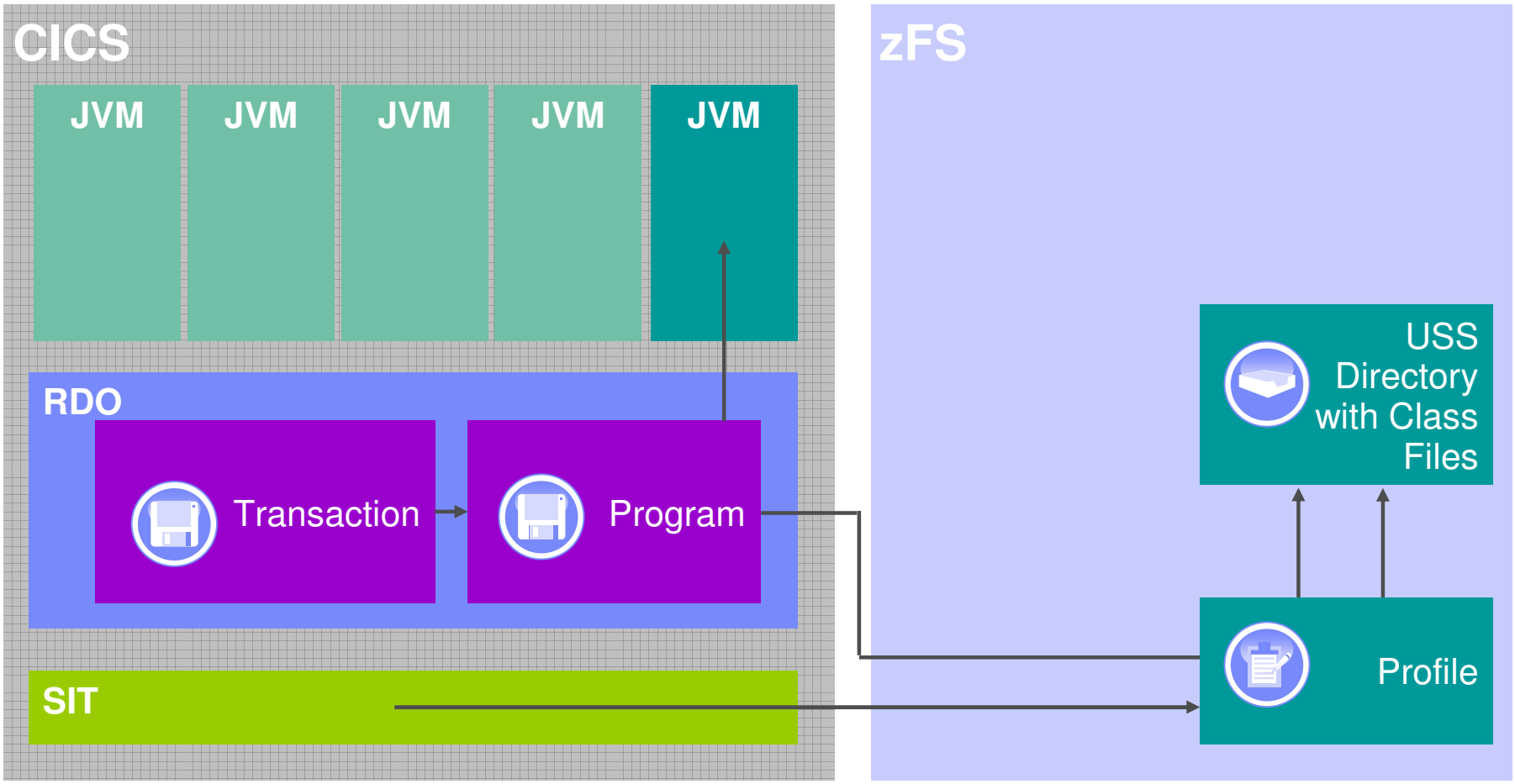
JVM Pool Architecture - CICS TS V3 (and V2)

A single CICS task dispatched into a JVM in the pool at a time. So concurrent task count limited to the number of JVMs that can fit in the 31-bit address space.

Each JVM 'costs' ~20Mb plus the application heap value.

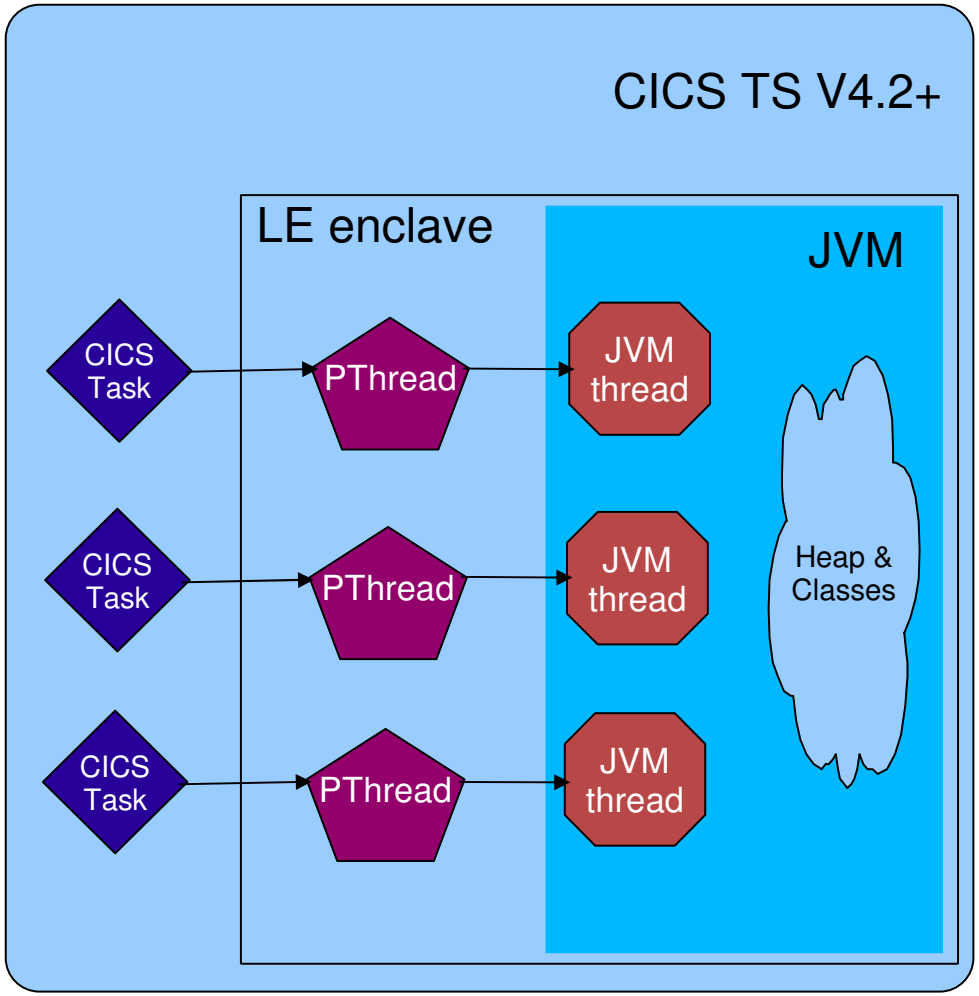


CICS JVM Pooled Architecture



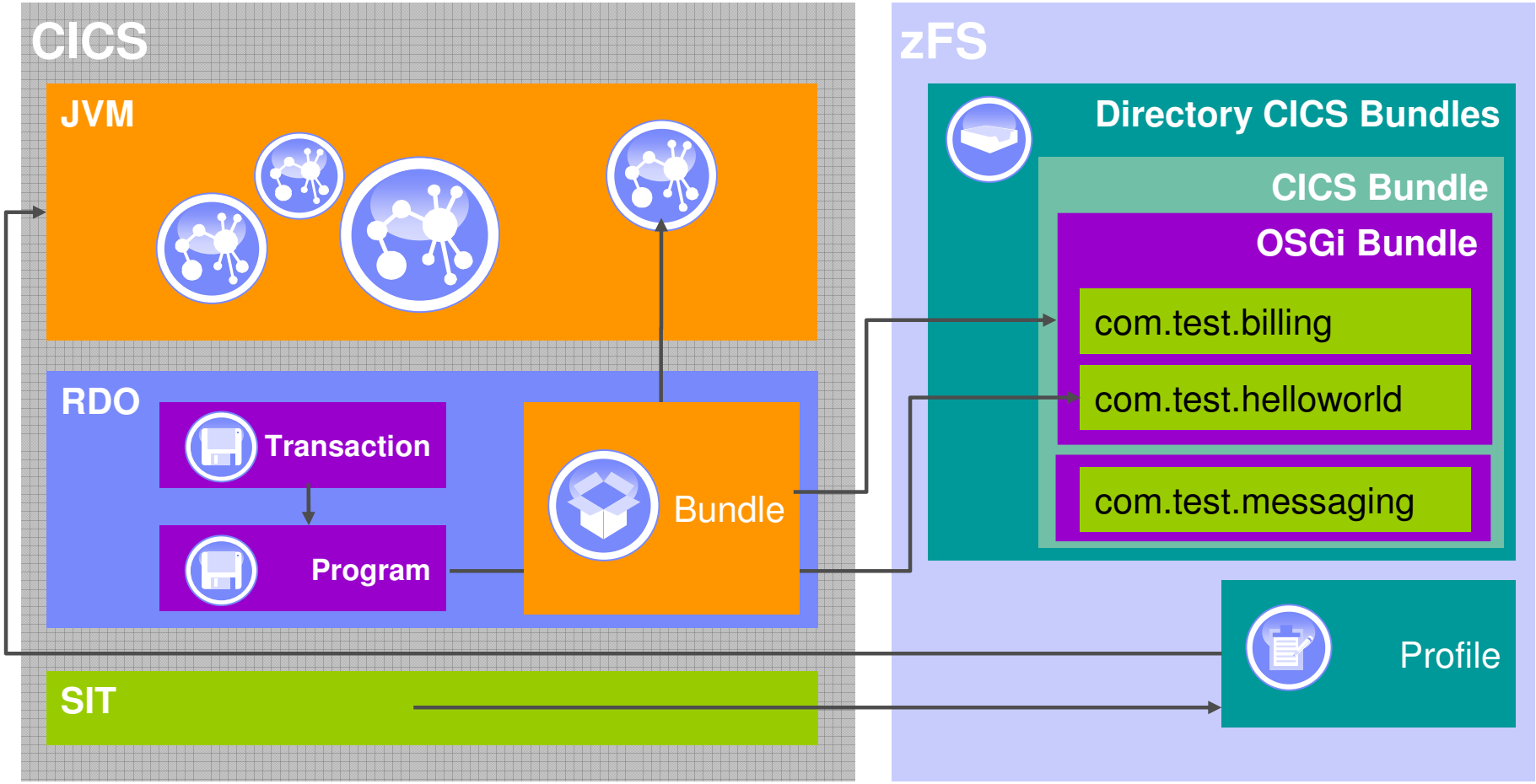
CICS JVM server model

CICS JVM server



- CICS requests storage from MVS, sets up a Language Environment enclave, and launches the 64-bit JVM in the enclave
- IBM® 64-bit SDK for z/OS, Java Technology Edition, Version 6.0.1 or Version 7
- Up to 256 parallel tasks (threads) per JVM & 2000 threads per CICS
- Applications
  - Must be threadsafe
  - Must be deployed as OSGi bundles (in CICS bundles)
- Dynamic updates without restart
- No EJB support

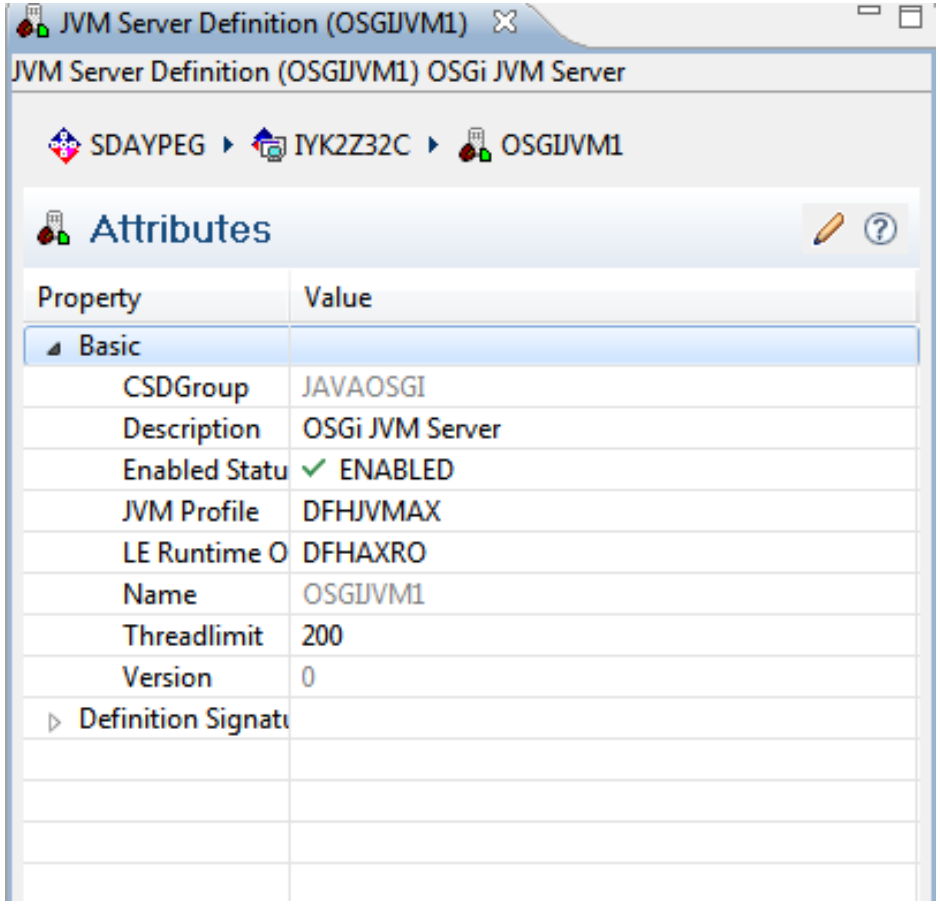
CICS JVM server architecture



JVM server Architecture

- New CICS OTE “T8” TCB mode, dubbed as both a CICS TCB and an LE “pthread”
- JNI call to attach a pthread to an existing JVM
- Can attach multiple pthread/T8/CICS tasks to the JVM at the same time
  - Serve many **more requests** using a single JVM
- JVM server thread “cost” is very small, result is **hundreds of tasks** concurrently per region
- Also architected to allow multiple JVM servers in a single CICS
  - For different types of work, or just a degree of isolation

Defining a JVM server



- JVM Profile
  - JVM profile in zFS in JVMPROFILEDR
  - DFHJVMAX is default
- LE Runtime Options
  - LE storage options
  - Defaults to DFHAXRO
- Threadlimit
  - Max number of T8 threads

JVM server: Threadsafety and OTE

- Java and Threadsafety – *yes, it may be a concern*
  - In a pooled JVM, static objects are 'mine' – there's only one application thread
  - In a JVM server, static objects are shared (visible and accessible) with all the other threads/tasks/transactions in the same server
  - Validate whether objects should be thread-local or static
  - Ensure the concurrent versions of library classes are used

- OTE – T8 and L8 threads
  - In V4.2, T8 TCBs are for Java server threads, L8 TCBs are required for DB2 → TCB switch for every JDBC command
  - **New in V5.1**, T8 TCBs can be used for DB2 so the TCB switch is saved.

Java Pool and EJB Statement of Direction

- CICS TS V4.2 announce letter
- A future release of CICS TS intends to **discontinue support for session beans using Enterprise Java Beans (EJB), and the Java pool infrastructure**. Customers are encouraged to migrate Java applications to the new JVM server infrastructure, and to migrate EJB applications to Java SE components and make them available through web services or the JEE Connector Architecture (JCA).

GONE IN V5.1 !!!

- **CICS will continue to support Java as a first class application programming language for CICS applications**, including enhancements to the CICS interfaces, the deployment infrastructure, and Java runtime environment.

CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Support for Java 64-bit JVMs

- CICS now supports 64-bit JVMs
  - Java 6.0.1 (CICS V4.2) or Java 7 (CICS V5.1)
    - If JAVA_HOME points to unsupported Java version
      - Error message DFHSJ0900 (pooled), DFHSJ0210 (JVM server)
  - Java byte codes do not need recompilation (write once run anywhere)
  - Support for 31-bit JVMs dropped (as of CICS V4.2)

Support for Java 6 64-bit JVMs

- Pooled JVMs **(V4.2 only)**
  - Support for many more JVMs per CICS region
    - 100+ can be possible
  - Larger heap sizes
    - Reduces impact of Garbage Collection
  - Profile changes
    - JAVA_HOME=/usr/lpp/java6_64/J6.0_64
    - USSHOME replaces CICS_HOME system initialization parameter

Support for 64-bit JVMs

- JVM server
  - Much larger heaps possible
  - Garbage Collection runs after an allocation failure
    - CJGC transaction is no longer used
    - Default GC policy uses more efficient gencon model
    - Heap dynamically sized by JVM
  - -Xcompressedrefs option uses 32-bit pointers to address 64-bit storage
    - Works for heaps up to 25GB
      - Reduces CPU consumption but only recommended for use with single JVM server regions

Support for 64-bit JVMs

- MEMLIMIT
  - Java stack and heap now allocated in above the bar storage
  - Above the bar requirement per Pooled JVM
    - –Xmx value in JVM profile
    - HEAP64 value in DFHJVMRO (default 8M)
    - LIBHEAP64 value in DFHJVMRO (default 1M)
    - STACK64 value in DFHJVMRO (default 1M) times 5 (application thread plus system threads)
  - Above the bar requirement per JVM server
    - –Xmx value in JVM profile (default 512M)
    - HEAP64 value in DFHAXRO (default 50M)
    - LIBHEAP64 value in DFHAXRO (default 1M)
    - STACK64 value in DFHAXRO (default 1M) times number of threads
      - THREADLIMIT plus system threads
      - Number of GC helper threads depends on –Xgcthreads parameter
      - Default is one less than the number of physical CPUs available

Support for 64-bit JVMs

- JDBC and SQLJ
  - DB2 8.1 or 9.1 required to support 64-bit applications
  - DB2 FP4 required for CICS TS 4.2 Java
  - Make sure you have the latest DB2 JDBC (JCC) Fixpack
- WebSphere MQ
  - 64-bit driver required
  - OSGi bundle required for JVM server
- Middleware bundles (MQ and DB2)
  - Need to be added to JVM servers using OSGI_BUNDLES and LIBPATH_SUFFIX settings in JVM profile
- Native DLLs (JNI)
  - All native DLLs must be recompiled with LP64 compiler option and bound as AMODE(64)
  - LE will not allow an AMODE(31) DLL to be loaded by an AMODE(64) DLL

CICS and Java – Agenda

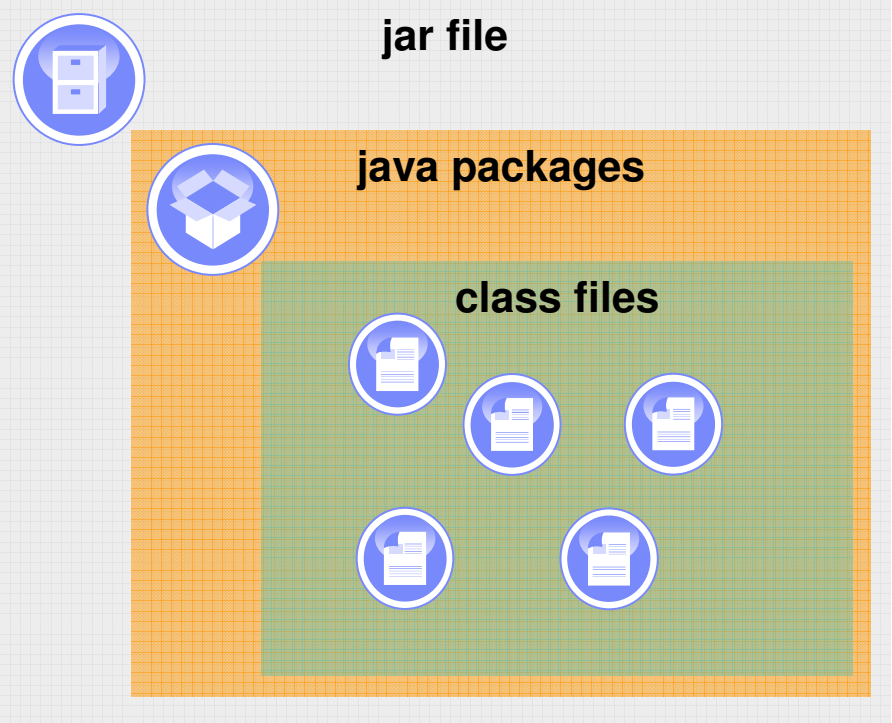
- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

OSGi – A dynamic module System for Java

> *provides a general-purpose, secure, and managed Java framework that supports the deployment of extensible and downloadable applications known as bundles.*
>
> The OSGi Alliance, Core Specification

Sounds familiar? Isn't that already possible with Java?

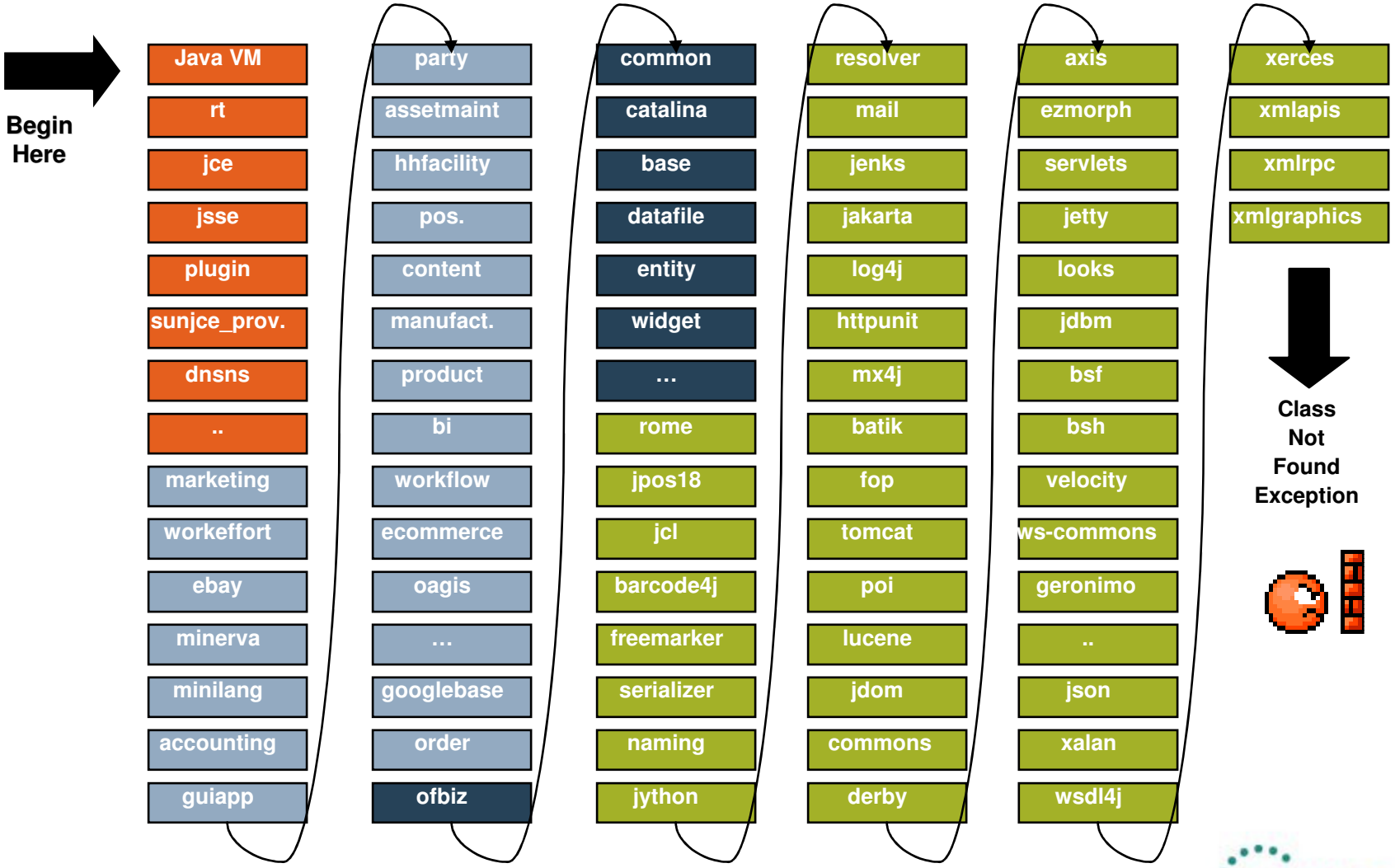
Java packaging and class hierarchies



- Java modularity:
  - –Classes contain data and logic
  - –Packages contain these classes and organize them
    - This is just a virtual form of organization
  - –Jar files contain the classes and are the base for enterprise applications
- Java visibility settings:
  - –Private, protected, package private, public
- At runtime, there are just a lot of classes on a classpath

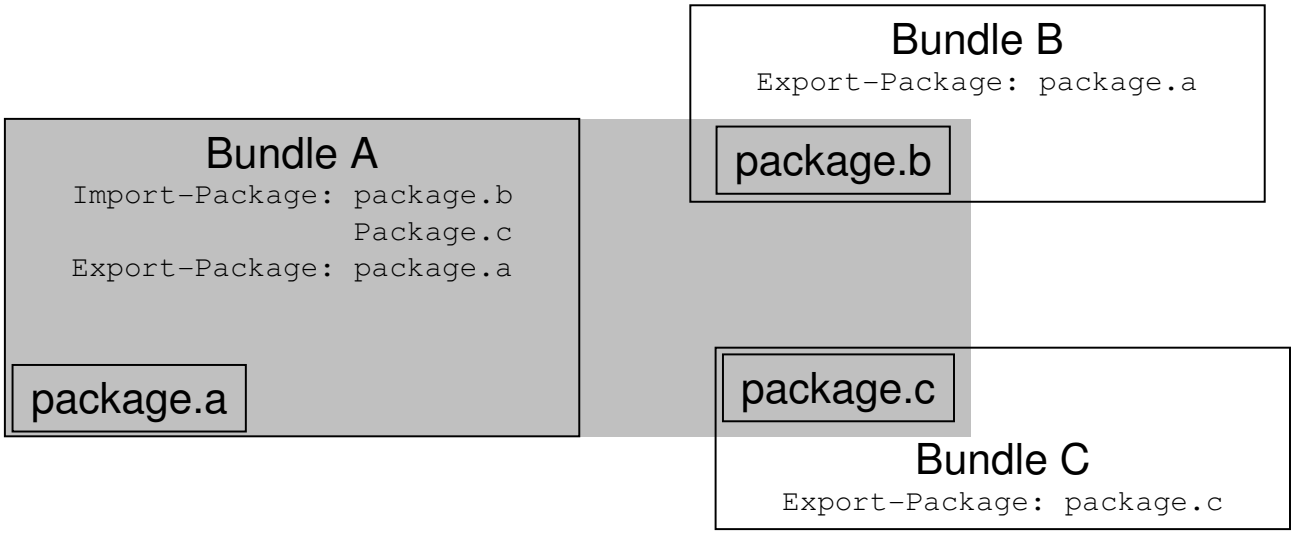
There are some features missing: jar visibility, versioning, dependencies

The Global Classpath



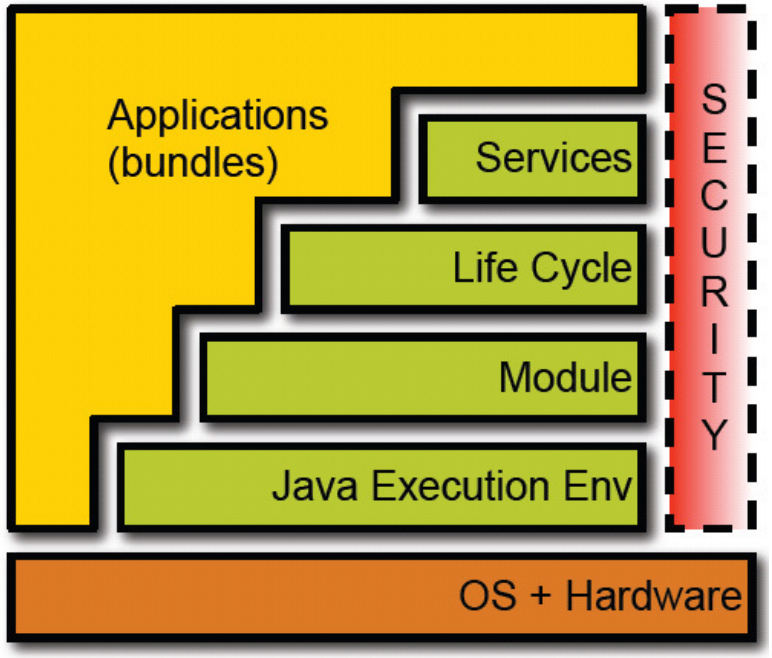
Class loading with OSGi

- No more CLASSPATH
  - Each bundle has its own class loader
- Class space is the classes required for the bundle
- Smallest unit is a package



What does OSGi provide?

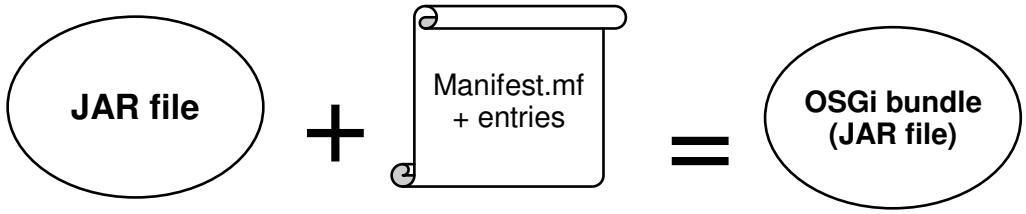
- The OSGi Service Platform specifies a modular architecture for dynamic component based systems
  - Execution Environment
  - Module Layer
  - Life Cycle Layer
  - Service Layer
  - Security-Layer (optional)
- Runs on a variety of standard Java profiles.
- Introduces *Bundles* as modules



OSGi bundles

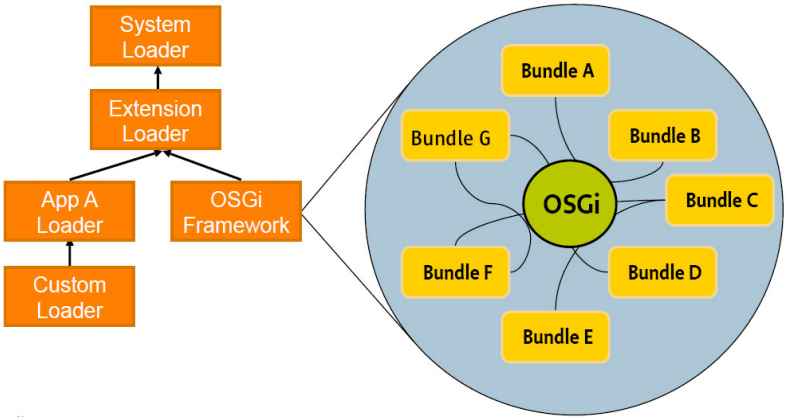
OSGi Bundle – A jar containing:

- Classes and resources
- OSGi Bundle manifest
  - Bundle-Version: Multiple versions of bundles can live concurrently
  - Import-Package: What packages from other bundles does this bundle depend upon?
  - Export-Package: What packages from this bundle are visible and reusable outside of the bundle?




```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Hello Plug-in
Bundle-SymbolicName: com.ibm.cics.server.examples.hello
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: J2SE-1.4, J2SE-1.5, JavaSE-1.6
Import-Package: com.ibm.cics.server;version="[0.0.0,2.0.0)“
Export-Package: examples.hello
```

OSGi Class Loader Model

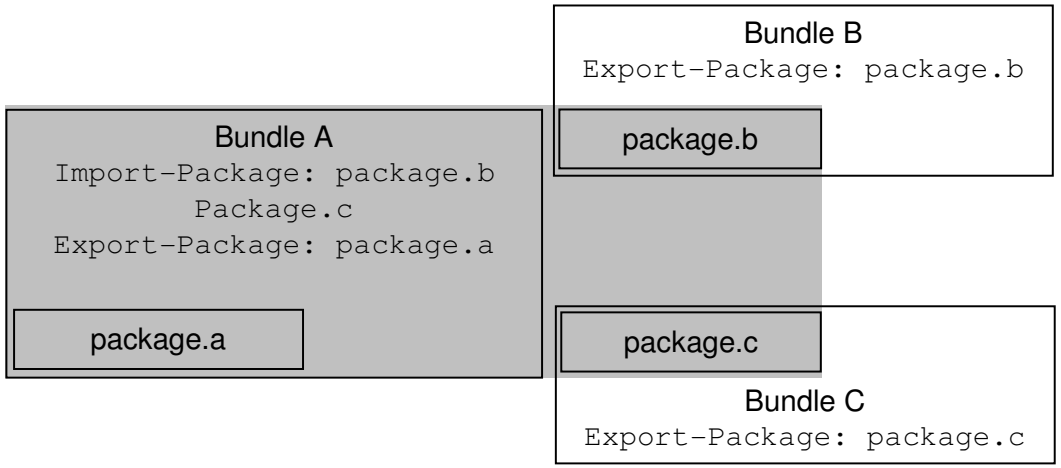


When bundles are installed into OSGi framework, the module layer

1. Processes metadata in the manifest file
2. Reconciles declared external dependencies against exports & version information declared by other installed modules
3. Works out all dependencies and calculates independent required class path for each bundle

Ensure that

- Each bundle provides **visibility** only to Java packages that it explicitly **exports** - export at specific versions possible
- Each bundle declares its package **dependencies** explicitly - import at specific / range of versions possible
- Multiple versions of a package can be available concurrently to different clients



CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

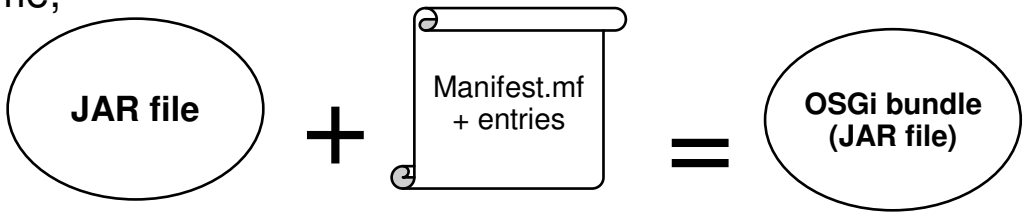
CICS OSGi Support Overview

- OSGi
  - OSGi development and packaging now required to deploy CICS applications to a JVM server[1]
  - Existing CICS Java applications using main() method linkage can run unchanged if wrapped in an OSGi bundle
  - All JVM server applications must be thread-safe and cannot use stabilized/removed CICS EJB or CORBA functions
  - Equinox used as OSGi implementation
- CICS Explorer SDK
  - Provides CICS Java development toolkit for use in any Eclipse 3.6.2 IDE (RAD 8.0 or vanilla Eclipse SDK)
  - Can be used to develop and deploy applications for any release of CICS (CICS TS 3.2 onwards)
  - Java projects are developed as Plug-in Projects and then packaged in a CICS bundle and exported to zFS
  - CICS TS V3.2/V4.1 Pooled JVM application classes/JARs can be wrapped and deployed to OSGi JVM servers

[1] **Note** Axis2 CICS Web Services applications do not support OSGi packaging

OSGi bundles within CICS

Most Parts of the descriptor are the same, except the CICS-MainClass:



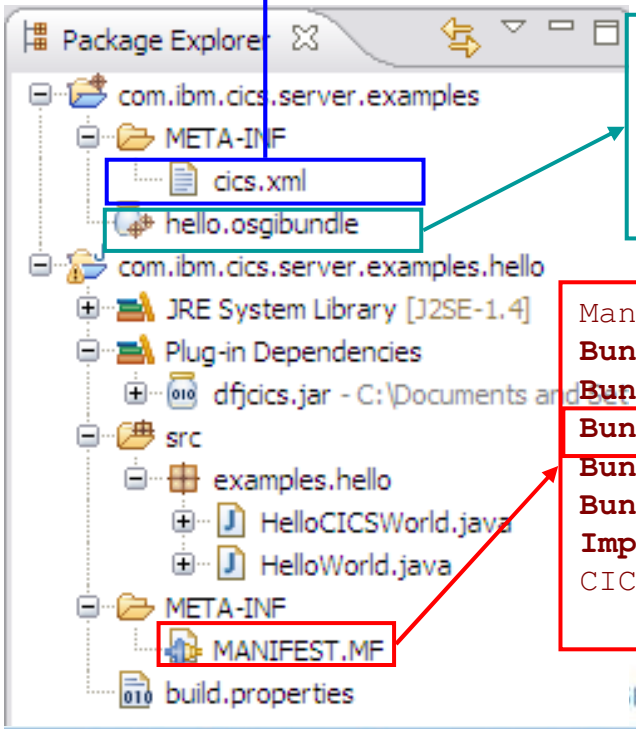
- CICS needs to know which Class can be called as a program
- CICS processes the metafile before it is handed to the OSGi framework and the information is stored in the CICS repositories.
- OSGi bundles can be viewed using CICS Explorer

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Hello Plug-in
Bundle-SymbolicName: com.ibm.cics.server.examples.hello
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: J2SE-1.4, J2SE-1.5, JavaSE-1.6
Import-Package: com.ibm.cics.server;version="[0.0.0,2.0.0)"
Export-Package: examples.hello
CICS-MainClass: examples.hello.HelloCICSWorld, examples.hello.HelloWorld
```

CICS bundle and OSGi bundle manifests

cics.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<manifest xmlns="http://www.ibm.com/xmlns/prod/cics/bundle"
bundleVersion="1" bundleRelease="0" build="Not Found">
 <meta_directives>
     <timestamp>2010-11-25T10:55:24.052Z</timestamp>
 </meta_directives>
 <define name="hello"
      type=http://www.ibm.com/xmlns/prod/cics/bundle/OSGIBUNDLE
      path="hello.osgibundle"/>
</manifest>
```



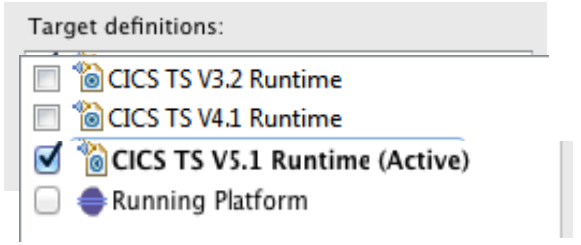
hello.osgibundle

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<osgibundle
   symbolicname="com.ibm.cics.server.examples.hello"
   version="1.0.0" jvmserver="DFH$JVMS"/>
```

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Hello Plug-in
Bundle-SymbolicName: com.ibm.cics.server.examples.hello
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: J2SE-1.4,J2SE-1.5,JavaSE-1.6
Import-Package: com.ibm.cics.server;version="[0.0.0,2.0.0)"
CICS-MainClass: examples.hello.HelloCICSWorld,
                                        examples.hello.HelloWorld
```

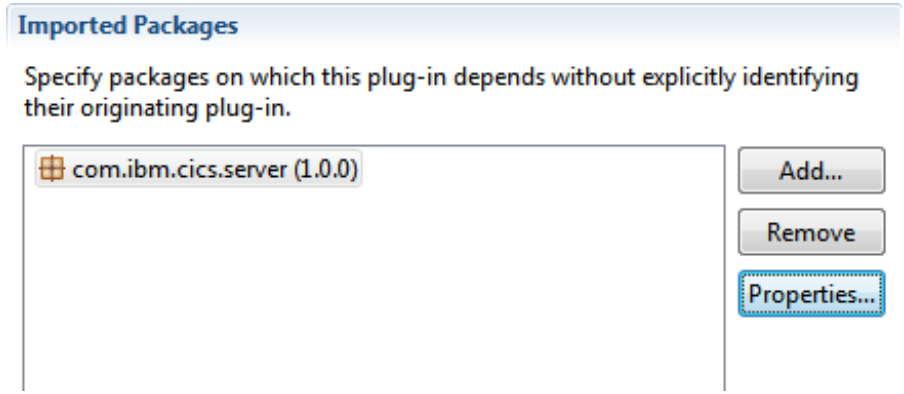
CICS Explorer SDK - Development

1. Install CICS Explorer SDK into Eclipse
2. Set Target Platform
   (sets JCICS and JVM levels)
   - Window → Preferences…→
     Target Platform → Add… → Template



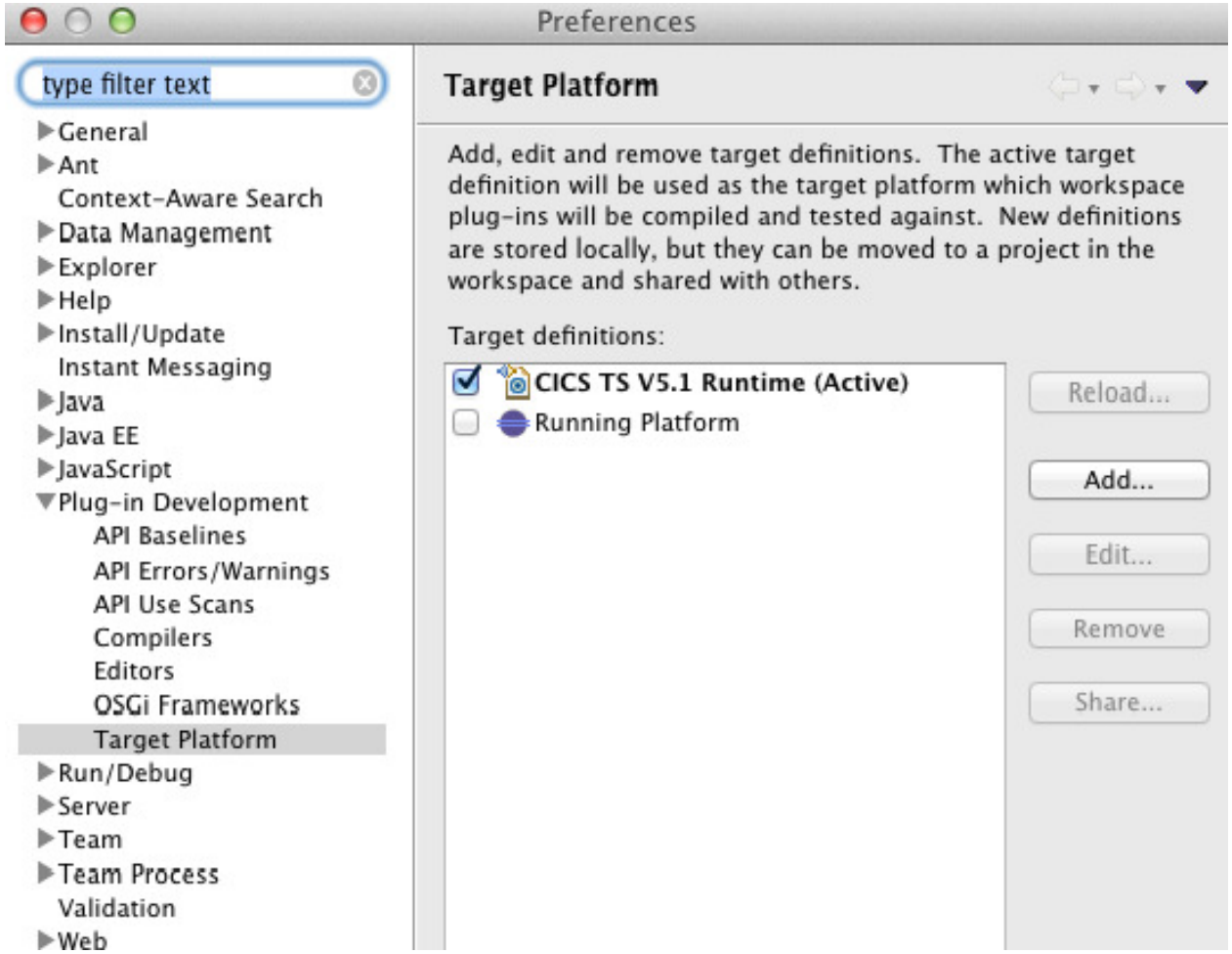
3. Create New OSGi Project
   - New → Plug-in Project

4. Provide access to JCICS package
   - MANIFEST.MF → Dependencies →
     Imported Packages →
     com.ibm.cics.server
   - Add other bundle imports if required

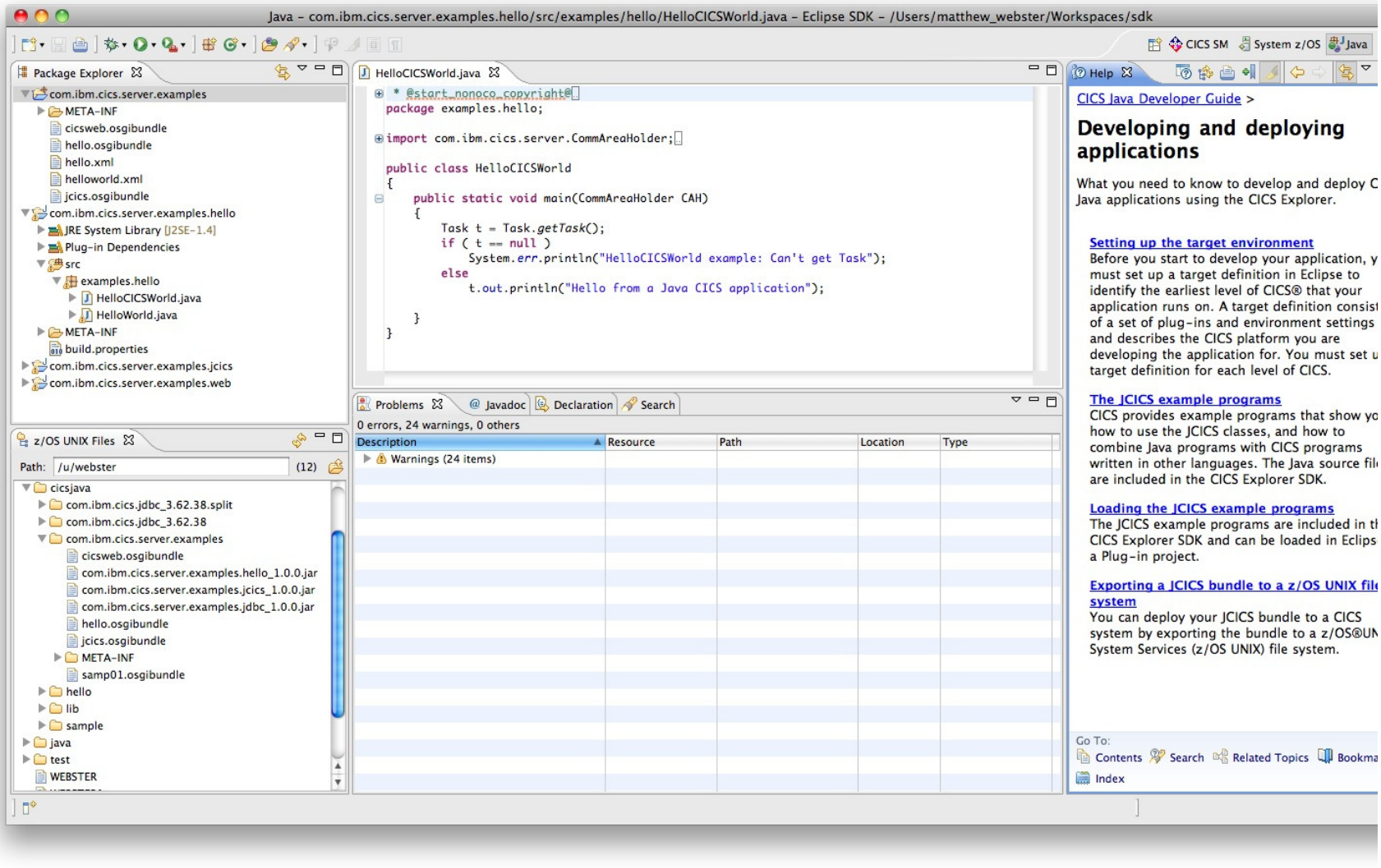


5. Import/Create your Java class

Target platform dialogue

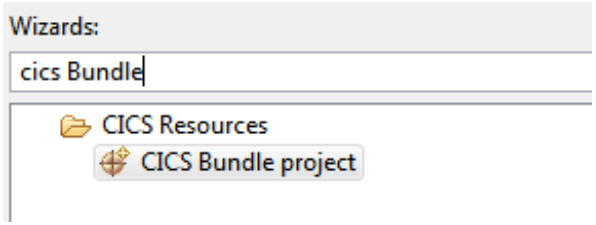


Java development in CICS Explorer SDK

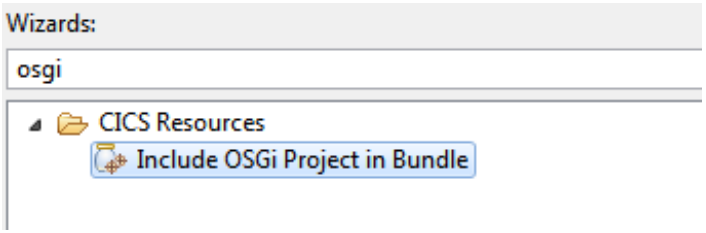


CICS Explorer SDK - Deployment

6. Create CICS Bundle
   - New→CICS Bundle Project



7. Add OSGi bundle meta-data file to CICS Bundle
   - New→Include OSGi Project in Bundle

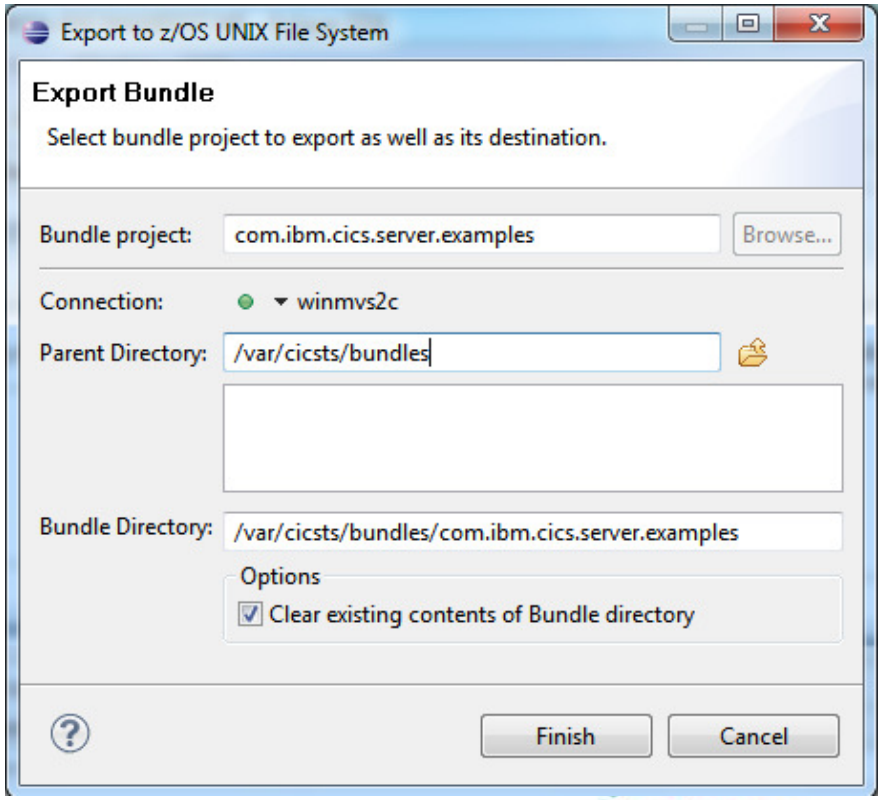


CICS Explorer SDK – Deployment 2

8. Provide CICS region userid read access to bundledir
   - mkdir /var/cicsts/bundles
   - chmod 750 /var/cicsts/bundles1

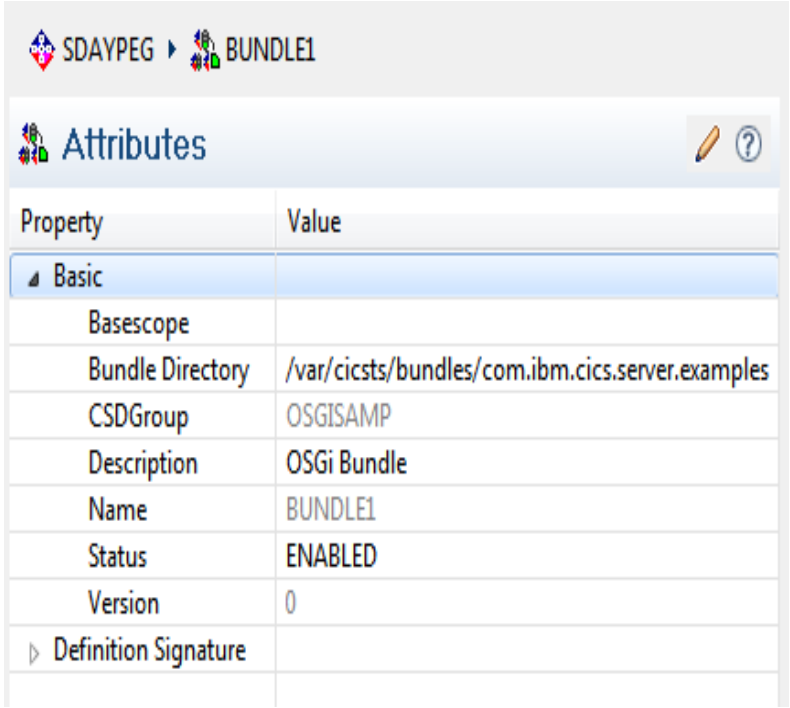
9. Connect CICS Explorer to USS FTP daemon
   - Windows → Open Perspective → z/OS

10. Export CICS Bundle to CICS
    - →CICS to z/OS UNIX File System



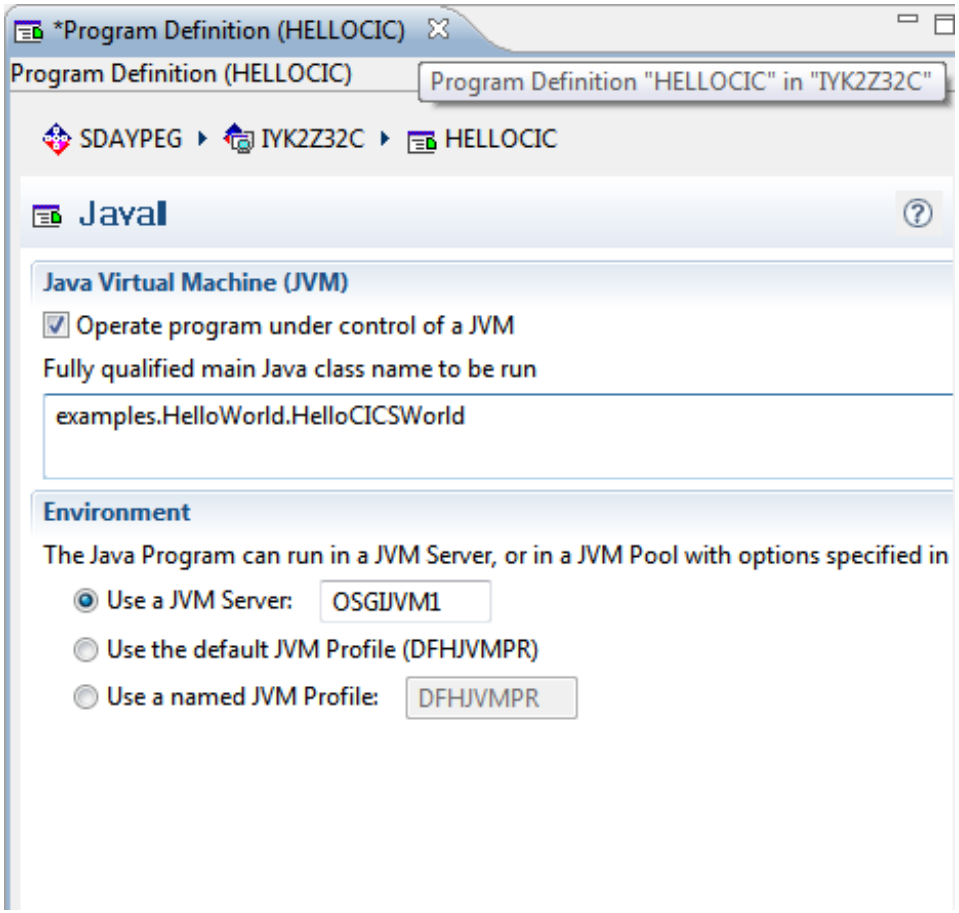
[1] **Note:** CICS region userid and FTP user must be in same USS group

Defining a CICS BUNDLE



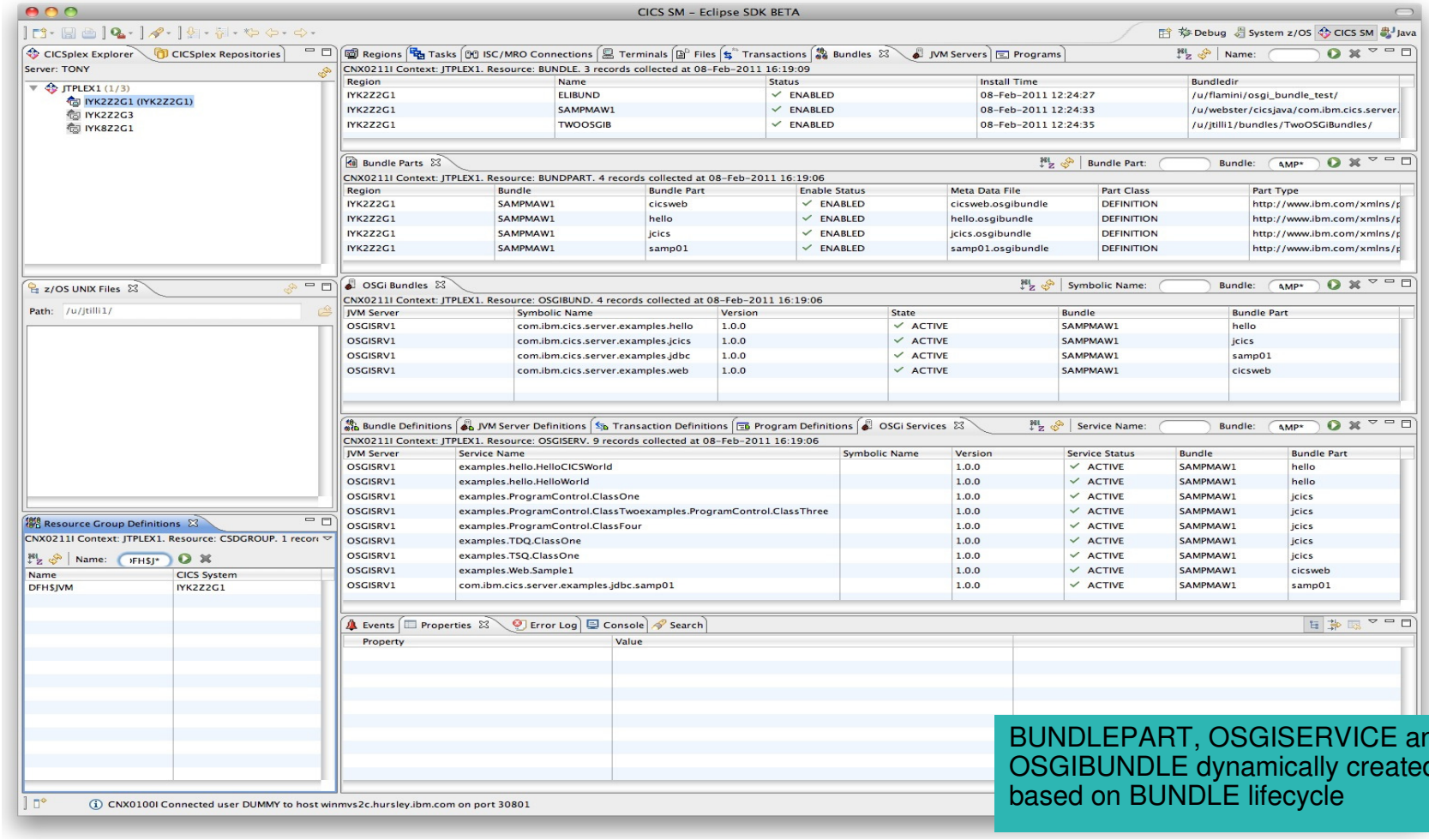
- Bundle Directory
  - Name of directory containing deployed JAR and bundle meta data files
- Status
  - ENABLED→Activate on install of resource

Defining a Program to run in JVM server



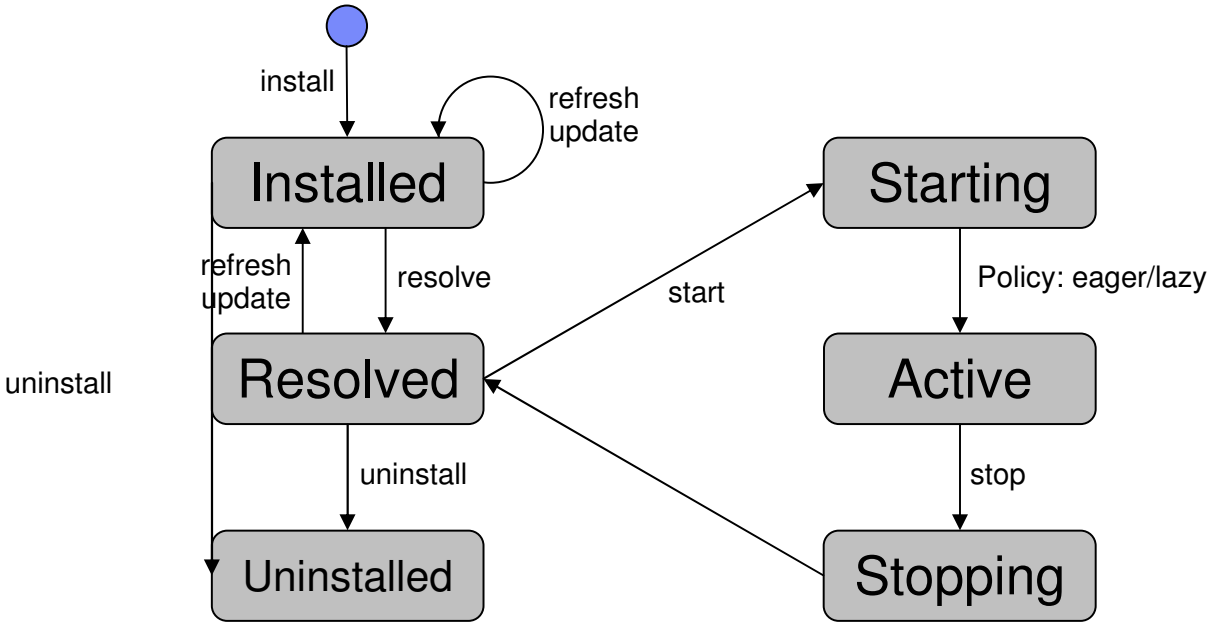
- JVMServer
  - Name of JVM server resource
- Main Java class
  - OSGIService defined in the OSGi bundle manifest
  - Either an alias or the full package.class name
- Also required
  - CONCURRENCY(THREADSAFE)
  - EXECKEY(CICS)

OSGi Bundles and Services in CICS Explorer



BUNDLEPART, OSGISERVICE and OSGIBUNDLE dynamically created based on BUNDLE lifecycle

OSGi Bundle Lifecycle



OSGi bundle state displayed in CICS Explorer OSGi Bundle view

Tasks | Programs | JVM Serve | Bundles | Bundle Par | OSGi Bund | OSGi Servi | JVM Serve | Bundle De | Program D | Transactio

CNX0211I Context: IYK2Z32C. Resource: OSGIBUND. 4 records collected at 24-Mar-2011 09:41:29 — Symbolic Name:

Region	Symbolic Name	State	Bundle Part	Bundle	JVM Server	Install Time	Version	Bundle ID
IYK2Z32C	com.ibm.cics.server.examples.hello	✓ ACTIVE	hello	SAMPLES	OSGIJVM1	24-Mar-2011 09:41:11	1.0.0	13
IYK2Z32C	com.ibm.cics.server.examples.jcics	✓ ACTIVE	jcics	SAMPLES	OSGIJVM1	24-Mar-2011 09:41:11	1.0.0	14
IYK2Z32C	com.ibm.cics.server.examples.web	✓ ACTIVE	cicsweb	SAMPLES	OSGIJVM1	24-Mar-2011 09:41:11	1.0.0	15
IYK2Z32C	sleep	✓ ACTIVE	sleep	SLEEP	OSGIJVM1	23-Mar-2011 21:49:46	1.1.0	12

CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Provider Mode Java Application Handler

- New application handler written in Java
  - Use is optional
  - Executes in a JVM server
  - Eligible for zAAP off-load processing
    - XML data conversion can be off-loaded
- Based on Axis2 technology
  - An Open Source project from the Apache organization
    - http://ws.apache.org/axis2/

Java Web services with Axis2

- New Java CICS provided application handler

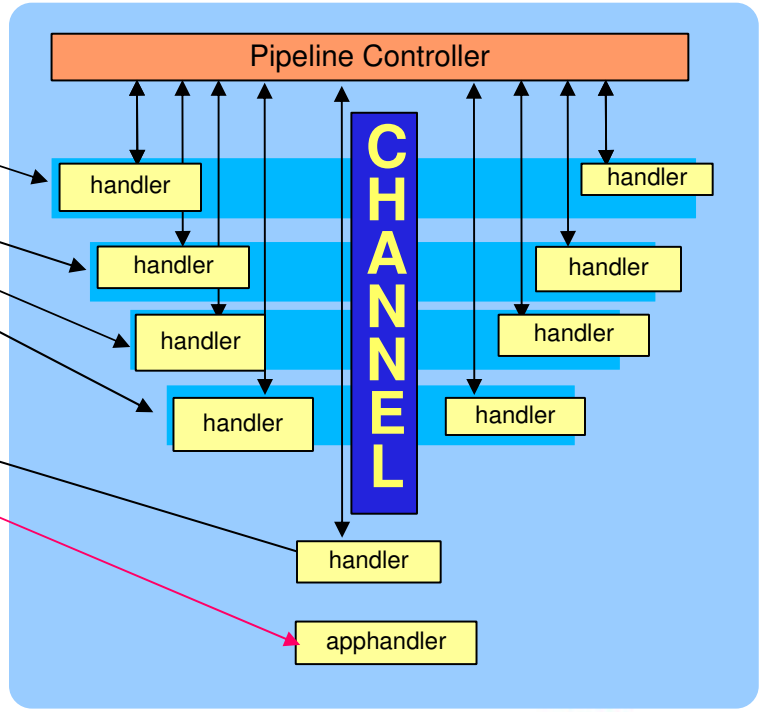
```
<transport>
  :
</transport>
```

```
<service>
  <service_handler_list>
   :
  </service_handler_list>
  <terminal_handler>
    <cics_soap_1.1_handler_java/>
  </terminal_handler>
</service>
```

```
<apphandler_class>...</apphandler_class>
```

```
<service_parameter_list />
```

```
</provider_pipeline>
```



Configuration for the Java application handler

- PROGRAM definition for the supplied handler
  - JVM set to YES
  - JVMCLASS
    - com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler
  - JVMSERVER name specified
    - Must match the name specified in the cics_soap_1.1_handler_java element in the configuration file
- JVMSERVER
  - Define a JVMSERVER for execution
- JVM profile updates
  - Add JAVA_PIPELINE=YES to the profile used
- Pipeline configuration file changes
  - jvmserver
  - Repository
  - addressing

Example pipeline configuration file:

```
<service>
   <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>MYJVM</jvmserver>
      <repository>/u/zem/wsdl/axis2</repository>
      <headerprogram>
        <program_name>MYPROG</program_name>
        <namespace>http://www.example.org/headerNamespace</namespace>
        <localname>*</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.1_handler_java>
   </terminal_handler>
  </service>
<apphandler_class>my.example.AppHandler</apphandler_class>
```

Provider Mode Axis2 Web Service

- Start with an existing Java application
  - POJO using JAX-WS
- Compile the Java application
  - javac TestAxis2.java
- Generate the WSDL and Bindings
  - wsgen –cp TestAxis2 – wsdl
- Package the application
  - Jar –cvf TestAxis2.jar *

Provider Mode Axis2 Web Service…

- Deploy the jar file to the Axis2 repository
  - Must be deployed to a servicejars directory in the repository
  - Repository is specified in the pipeline configuration file
- Define and install a URIMAP
  - Automatic install of a URIMAP cannot be used
  - Path name must follow Axis2 conventions
    - /name_of_serviceService.name_of_portPort/suffix
- A WEBSERVICE definition is not used

Provider Mode Axis2 Web Service…

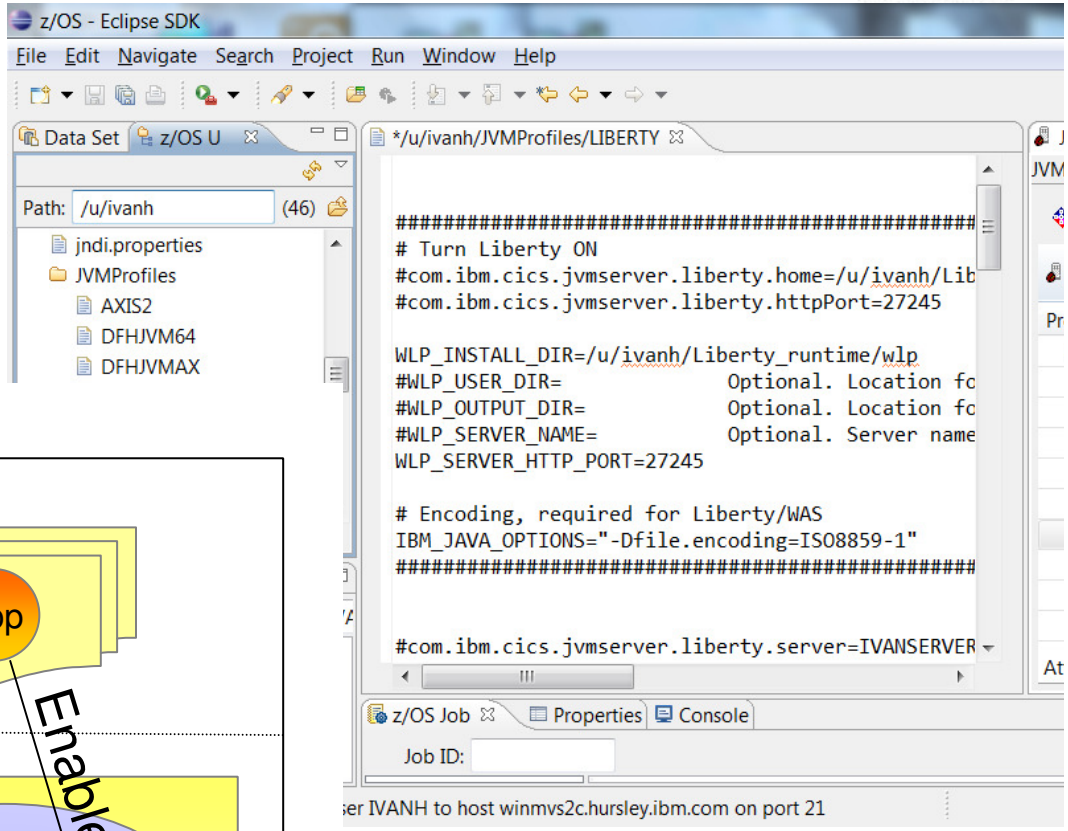
- CICS Services replaced by Java services
  - Axis2 applications interact with CICS with the Axis2 programming model
  - Some CICS services are not applicable
    - SOAPFAULT CREATE
    - WSACONTEXT GET
    - DFHWS-OPERATION container
    - DFHWS-MEP container
    - DFHWS-USERID container
    - DFHWS-TRANID container
    - Web services security

CICS and Java – Agenda

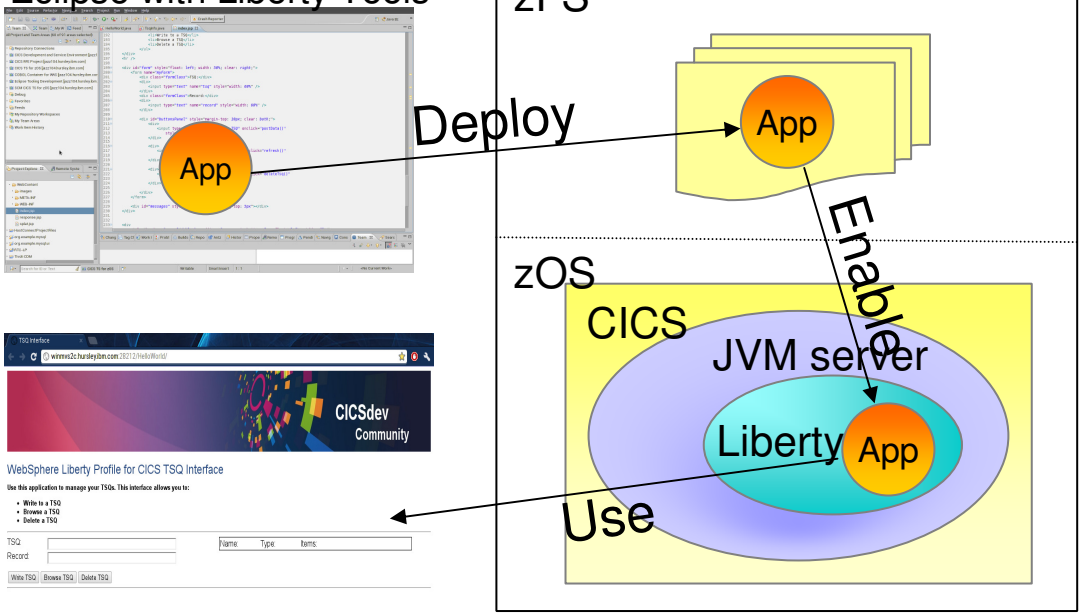
- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Liberty Profile in CICS TS V5.1

- Running WebSphere Application Server Liberty profile we container within a CICS JVM server enables Servlets in CICS with just a few additional Options in the JVM profile file



Eclipse with Liberty Tools



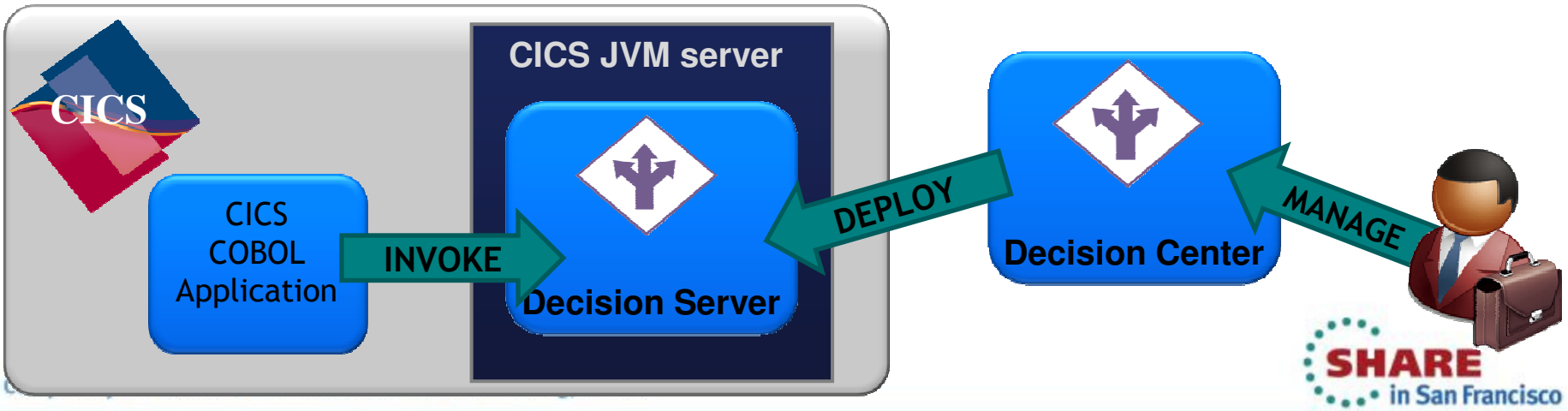
CICS and Java – Agenda

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
  - Support for 64-bit JVMs
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management

Operational Decision Manager & CICS

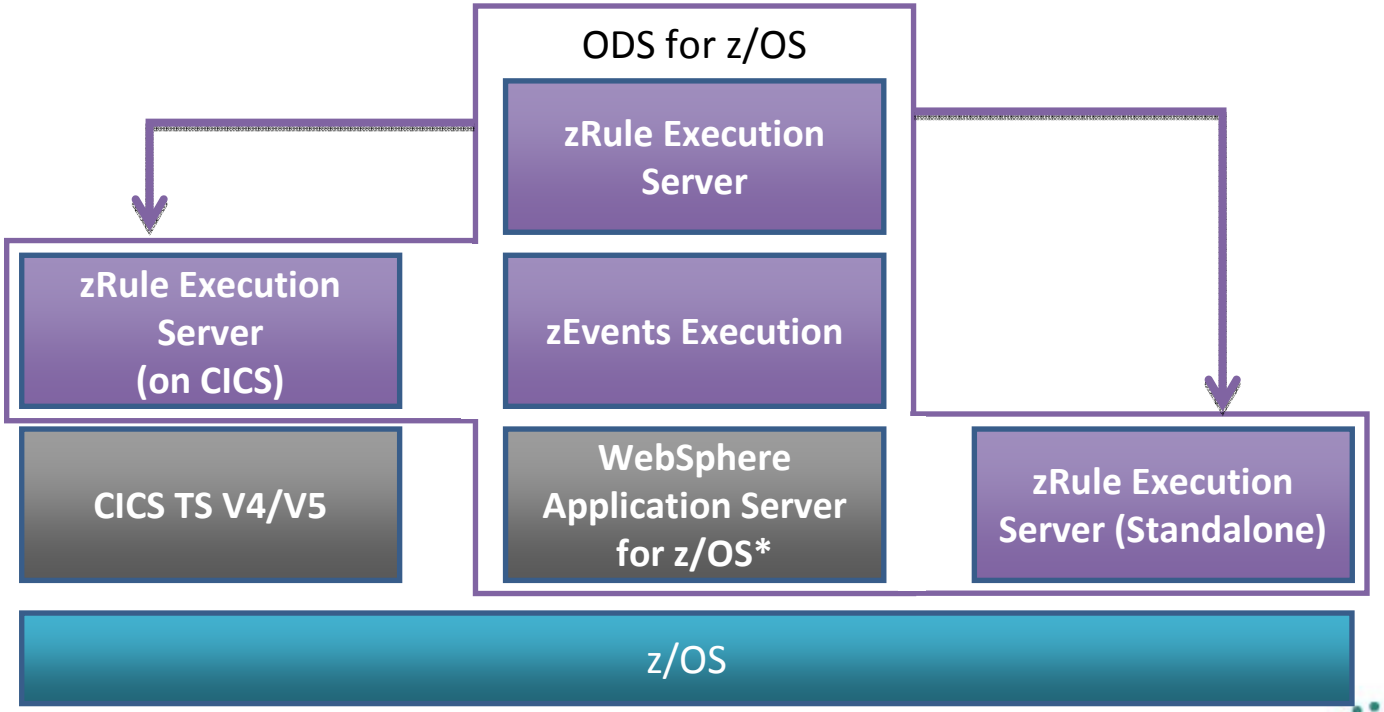
Externalize embedded business rule logic & execute within CICS

- *Gain business agility with existing and new CICS applications*
  - Manage decision logic on a separate lifecycle to application code
  - Ability to react to changes in a fast paced, competitive marketplace
- *Lower the cost of maintaining your business applications*
  - Improvement operational efficiency and total cost of ownership
- *Consistent Decision evaluation across the enterprise*
  - Author decision rules once and deploy to multiple systems on z/OS and distributed
- *Optimized decision execution*
  - Highly efficient rule execution engine
  - Local optimization of Decision Server within the CICS JVM server environment



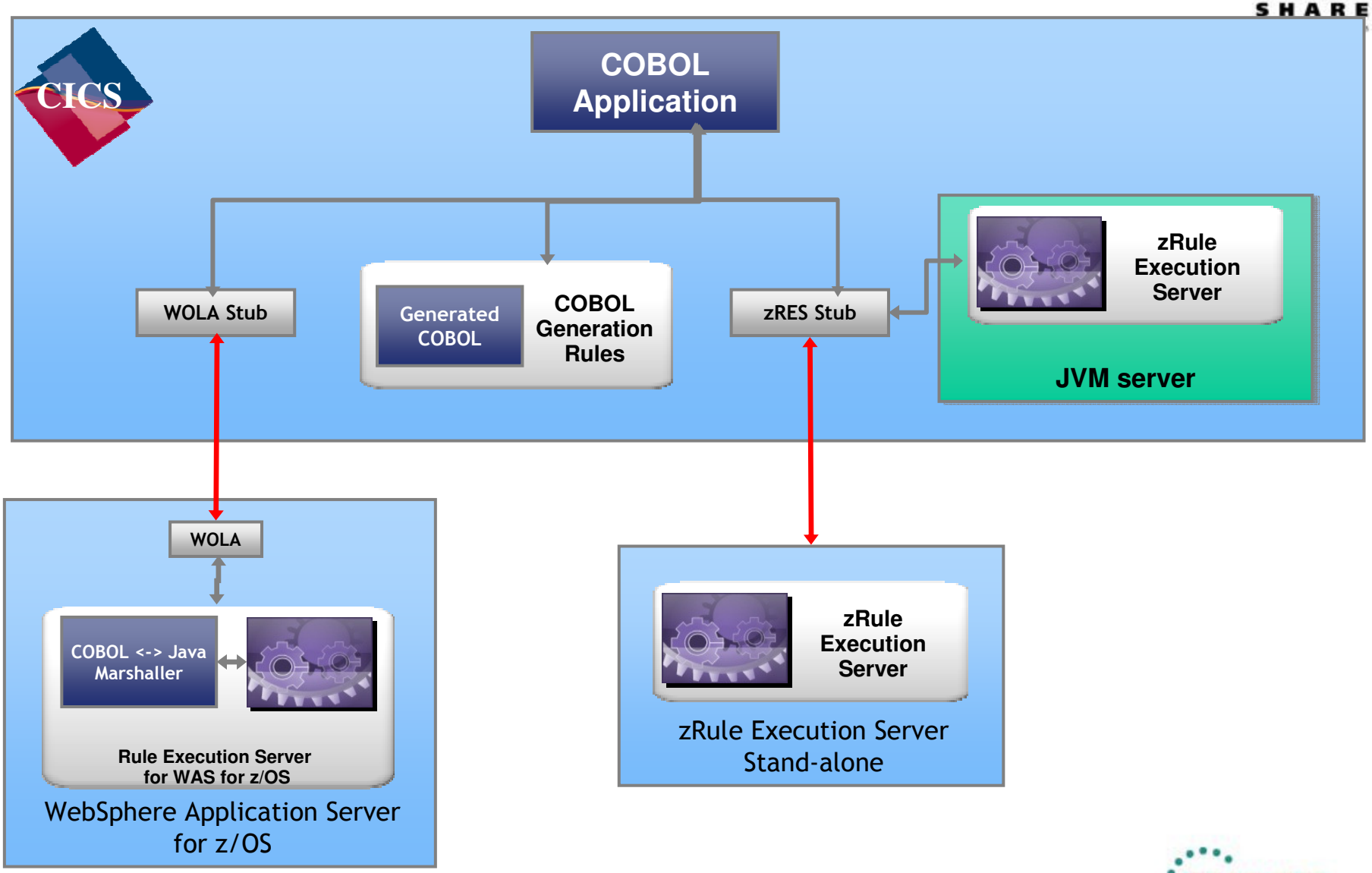
Decision Server for z/OS

- Decisions can be invoked from existing CICS and batch applications
- Runtime support for COBOL data types
- Flexible runtime deployment to fit any System z environment:
  - Deployed on WebSphere Application Server for z/OS
  - Deployed standalone to z/OS
  - Deployed in CICS TS V4.x/V5.1 JVM server environment



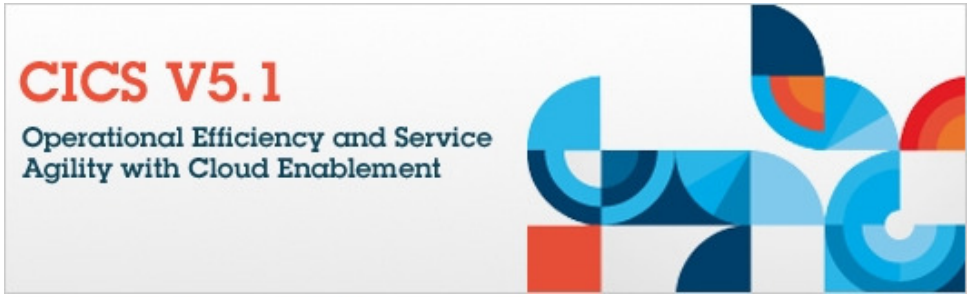
*OEM

Rule invocation options for CICS



CICS and Java – Summary

- Java, and why Java on z
  - Java on z roadmap
- Java in CICS
  - Pooled JVM and JVM server
- Introduction to OSGi
- OSGi in CICS
- Other CICS Java enhancements
- Exploiters of CICS JVM server
  - WebSphere Liberty profile
  - WebSphere Operational Decision Management



Any Further Questions?

I ♥ CICS