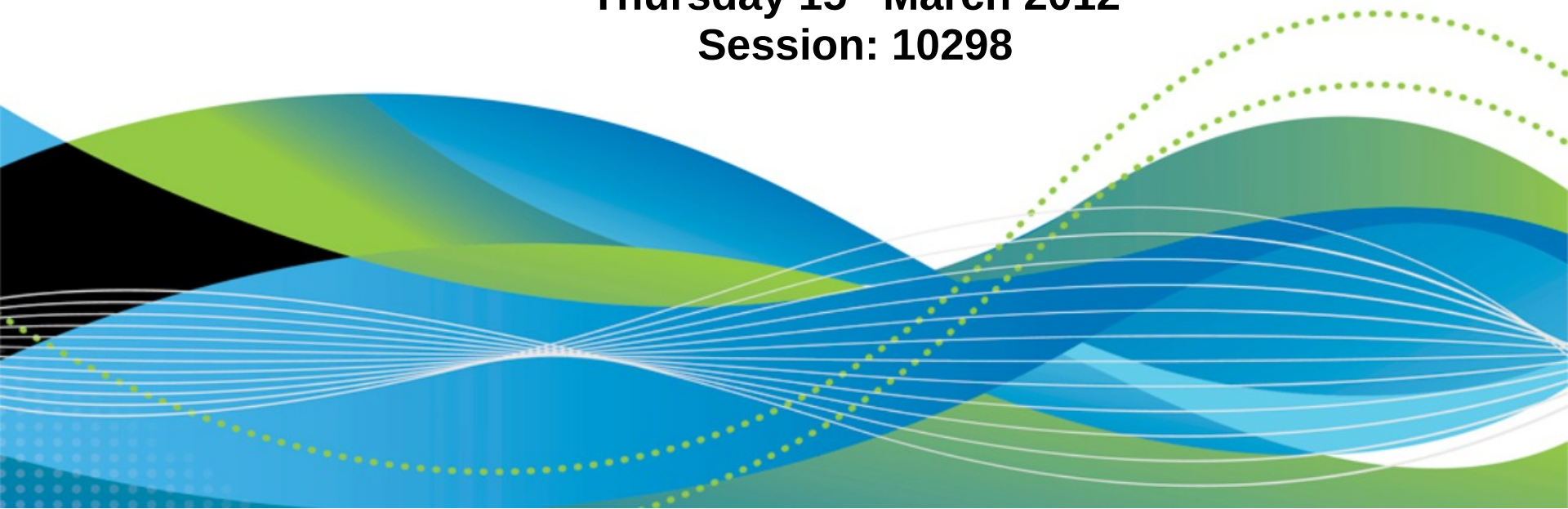


The CICS JVMServer and The WebSphere Operational Decision Manager Rules Execution Engine (ILOG)

Ian J Mitchell, IBM Distinguished Engineer

Thursday 15th March 2012
Session: 10298



Disclaimer

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.

The 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.

Agenda

- JVM Options in CICS TS 4.2
 - JVM Pool
 - JVM Server
- 64 Bit JVM Support
- OSGi for application management
- Java Web Services with Axis2
- WODM Rules Execution Engine



CICS Transaction Server for z/OS V4.2



Events



- System Events
- Assured Events
- Lifecycle Management

Management



- Transaction Tracking
- Workload Management
- Password Phrases

Java

- 64-bit Applications
- Multithreaded Server
- OSGi Management



Scalability

- More Threadsafe
- Optimised Threadsafe
- 64-bit Exploitation



Connectivity



- Axis2 Web Services
- Web Services Offload
- HTTP & IP Extensions

***new and enhanced
capability across five
major technology areas***



CICS Transaction Server for z/OS V4.2



Events



- Discover Faster
- React Quicker
- Compete Better

Management



- Understand More
- Control Better
- Manage Easier

Java

- Perform Better
- Manage Easier
- Innovate Quicker



Scalability

- Process More
- Lower Costs
- Perform Better



Connectivity



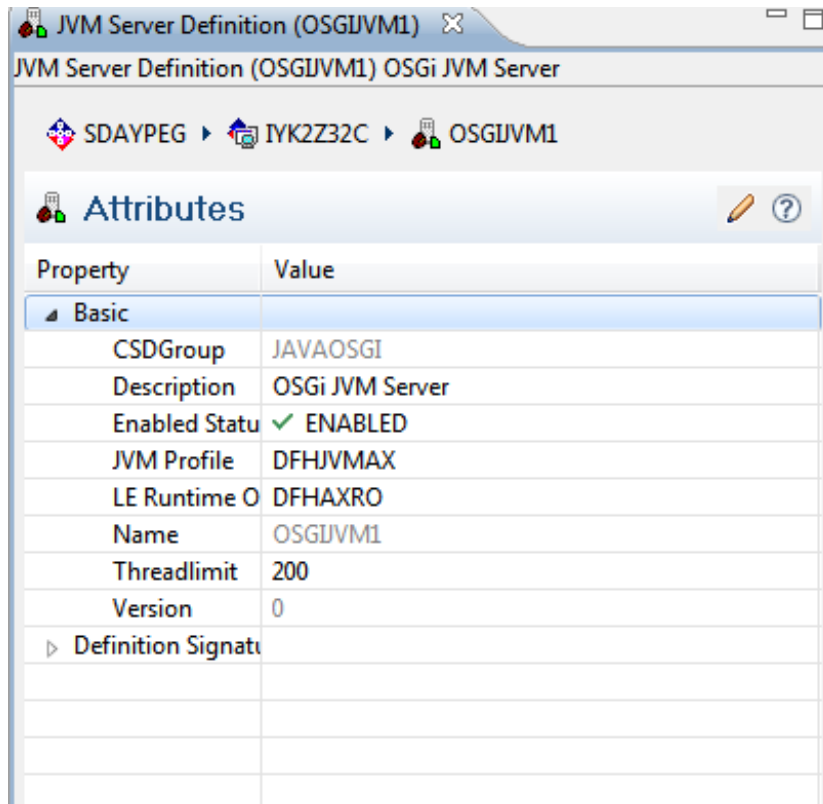
- Connect Easier
- Lower Costs
- Manage Better

*delivering a smarter
transaction processing
experience for everybody*

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

Defining a JVM server



Property	Value
Basic	
CSDGroup	JAVAOSGI
Description	OSGi JVM Server
Enabled Status	✓ ENABLED
JVM Profile	DFHJVMAX
LE Runtime Options	DFHAXRO
Name	OSGIJVM1
Threadlimit	200
Version	0
Definition Signature	

- JVM Profile
 - JVM profile in HFS in JVMPROFILEDR
 - DFHJVMAX is default

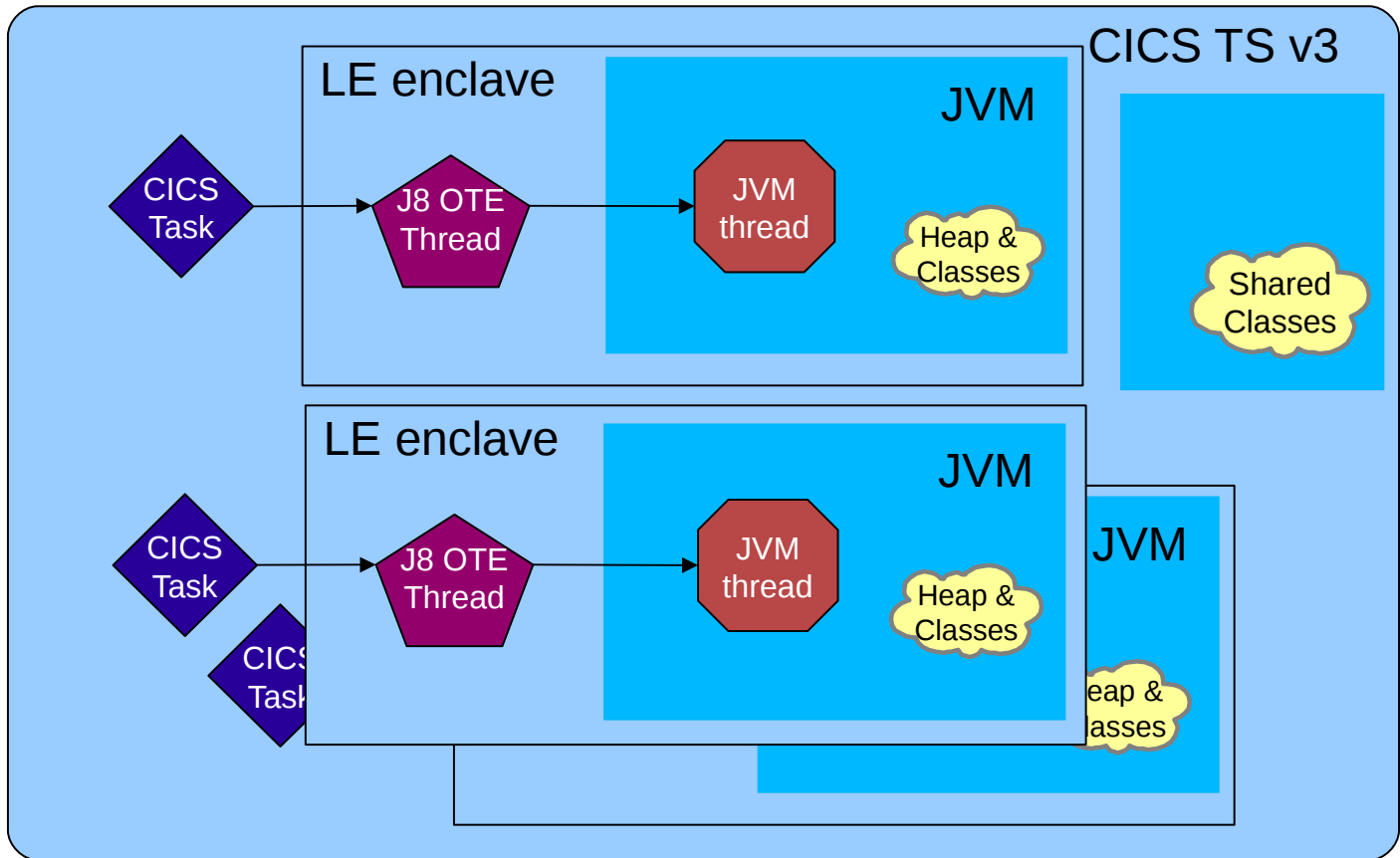
- LE Runtime Options
 - LE storage options
 - Defaults to DFHAXRO

- Threadlimit
 - Max number of T8 threads

JVMPool 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.

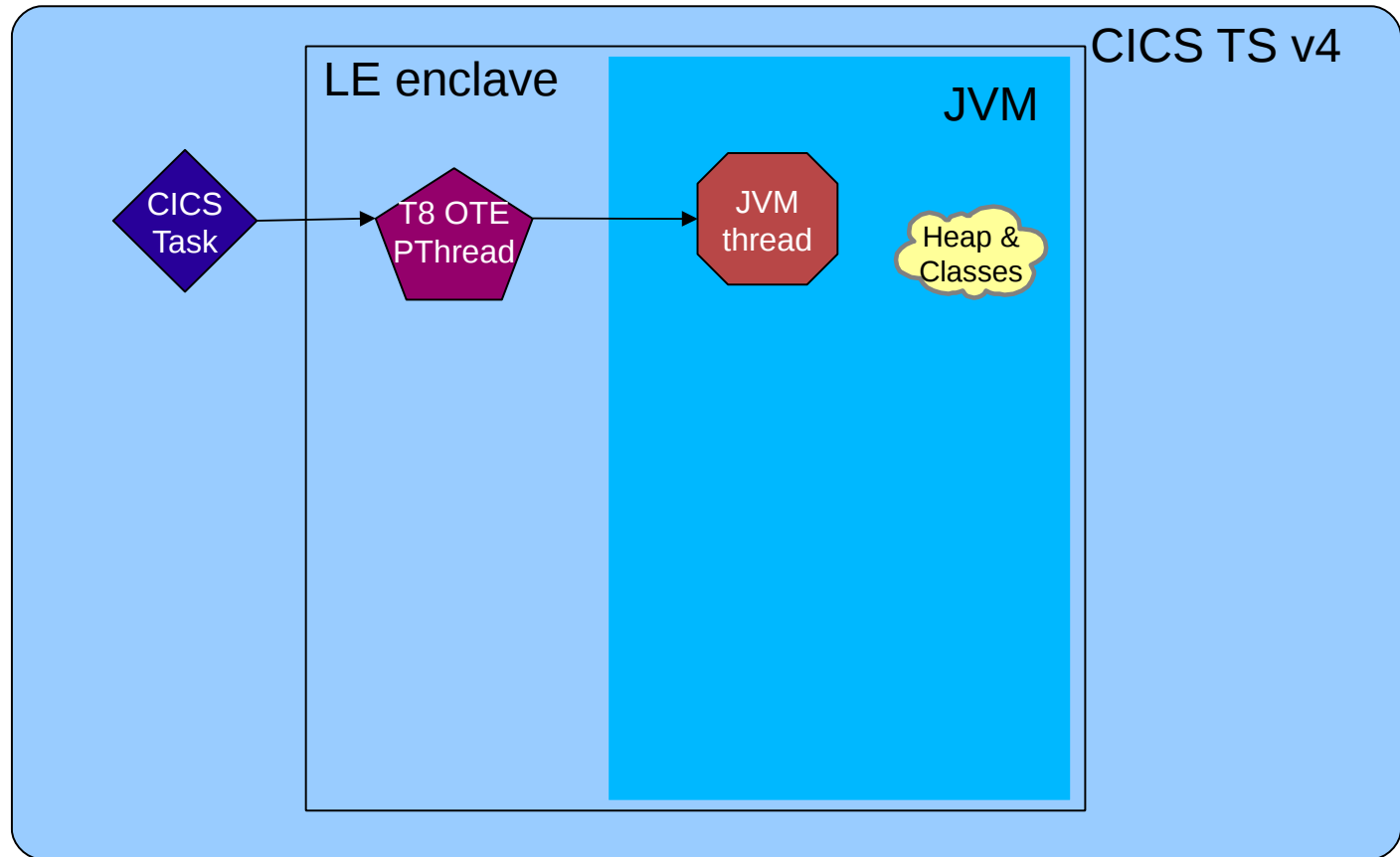


JVM Server Architecture

New CICS OTE TCB "mode".

Called "T8" - dubbed as both a CICS TCB and an LE "pthread".

JNI call to attach a pthread to an existing JVM.



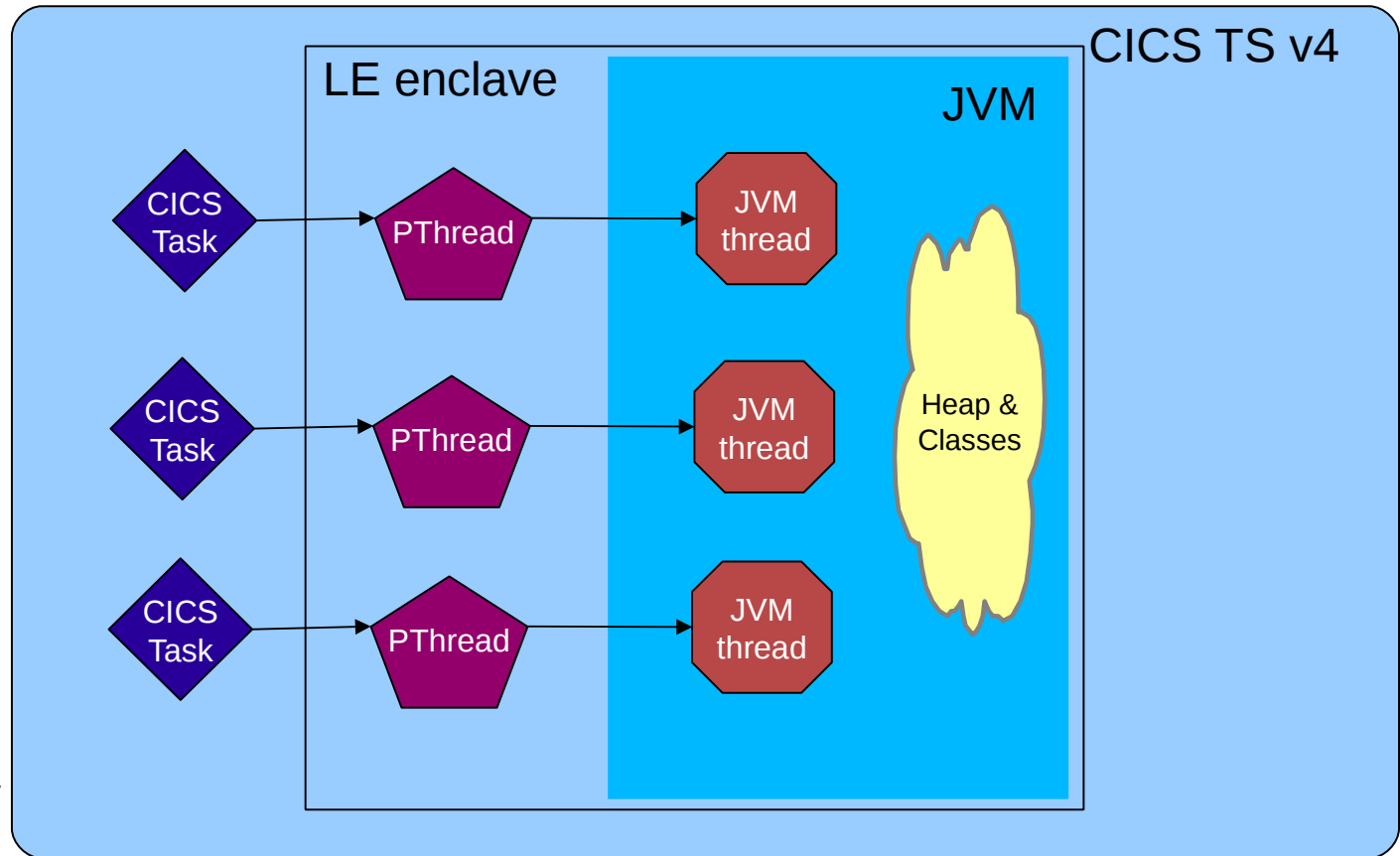
JVM Server Architecture

Can attach multiple pthread/T8/CICS tasks to the JVM at the same time.

Therefore serve **more requests** using a single JVM.

JVMServer thread “cost” is very small.

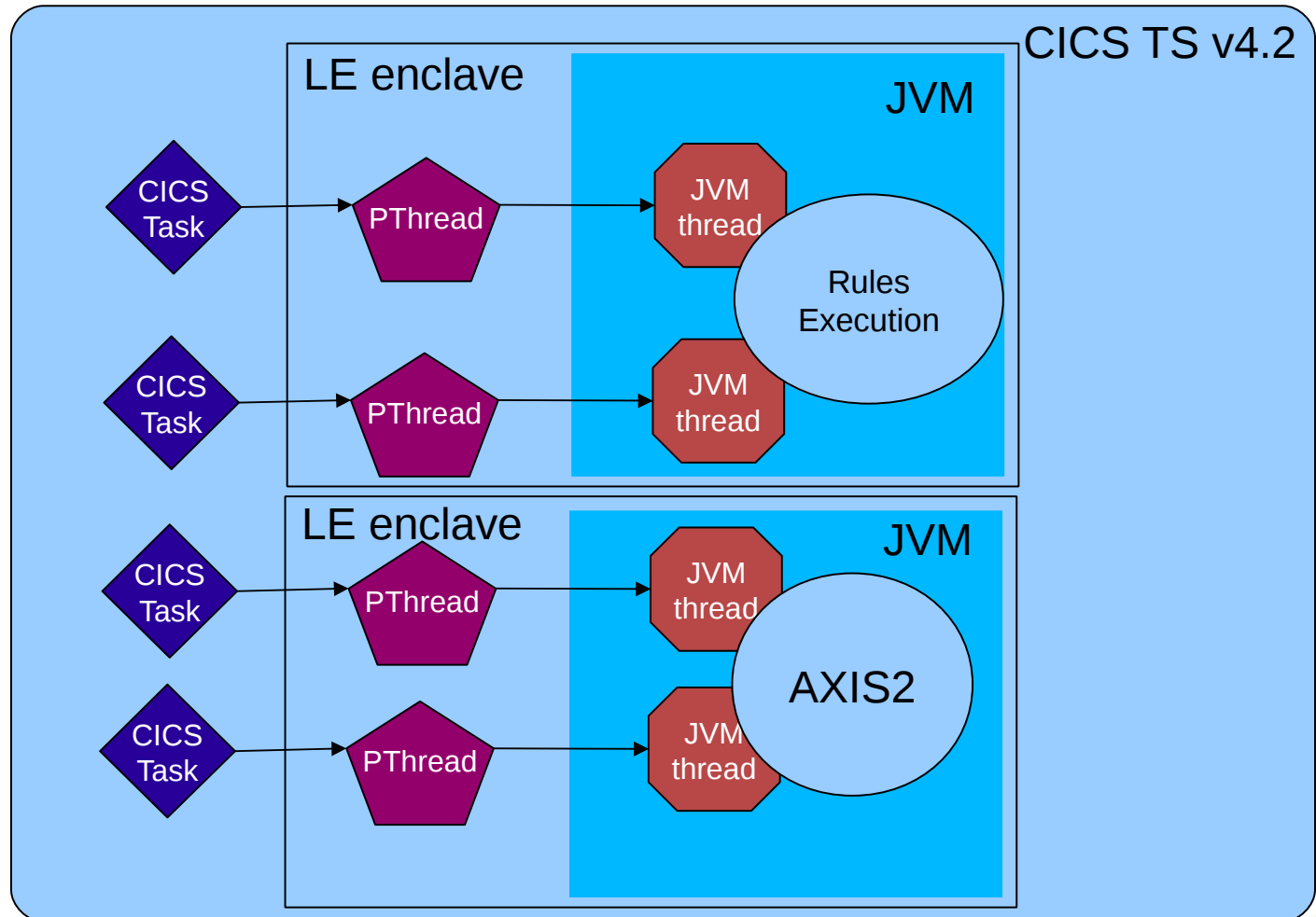
Result is **hundreds of tasks** concurrently per region.



JVM Server Architecture

Architected to allow multiple JVMServers in a single CICS.

Different types of work, or just a degree of isolation.



Support for Java 6 64-bit JVMs

Support for Java 6 64-bit JVMs

- CICS now supports 64-bit JVMs
 - Both Pooled JVMs and JVMSERVERs
 - Java 6.0.1 only
 - *If JAVA_HOME points to other than JVM then abend ASJJ*
 - *DFHSJ0900 09/27/2010 11:00:07 IYK2ZIK1 Illegal Java version. CICS requires Java version 1.6.0 but has found Java version 1.5.0.*
 - Java byte codes do not need recompilation (write once run anywhere)
 - Support for 31-bit JVMs dropped
 - *If JAVA_HOME points to a 31-bit installation, then abend ASJD*
 - *DFHSJ0503 09/27/2010 10:50:21 IYK2ZIK1 DFHJVMPR Attempt to load DLL libjvm.so has failed. Runtime error message is EDC5253S An AMODE64 application is attempting to load an AMODE31 DLL load module. (errno2=0xC40B0013)*
 - Java 6.0.1
 - *IBM zEnterprise optimized version of Java 6 JVM*
 - *Exploits new z196 instruction set*
 - *Improved GC*
 - *Improved JIT – (stores interpreter profiling information in class cache)*
 - *Significant performance improvements*
 - *Download from z/OS Java website*

Support for Java 6 64-bit JVMs

- Pooled JVMs
 - 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 Java 6 64-bit JVMs

- JVM Server
 - Messages now DFHSJxxxx instead of DFHLExxxx
 - 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 Java 6 64-bit JVMs

- MEMLIMIT
 - Java stack and heap are 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)*

Support for Java 6 64-bit JVMs

- MEMLIMIT
 - 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 Java 6 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
- WMQ
 - 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 OSGi Support

CICS OSGi Support Overview

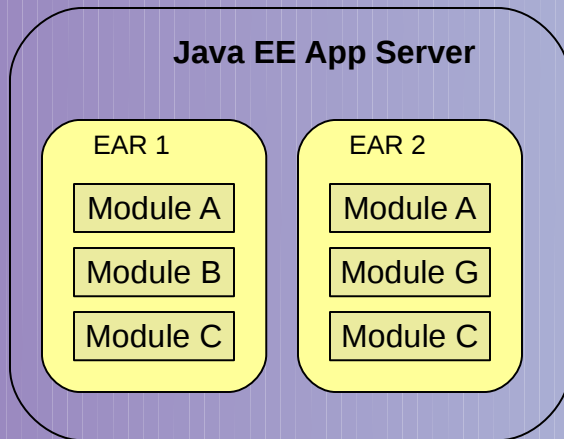
- OSGi
 - OSGi development and packaging now **required to deploy** CICS applications to a JVM server 1
 - Existing CICS Java applications using main() method linkage can run unchanged if wrapped in an OSGi bundle
 - All JVM server applications must be **thread-safe** and can't use stabilised CICS EJB 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 (i.e 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 applications classes/JARs can be wrapped and deployed to OSGi JVM servers

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

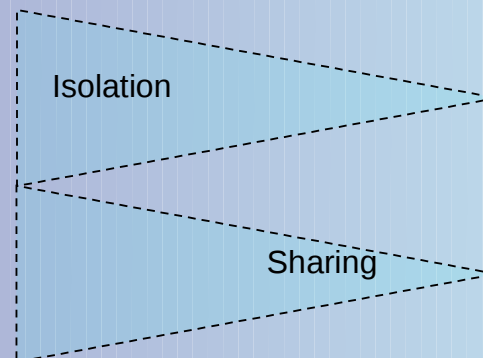
OSGi - Isolated and Shared Bundles

- In Java EE, modules are isolated within an application and applications are isolated from one another.
 - Makes sharing modules difficult
- OSGi 4.2 all bundles have shared visibility to the externals of all others bundles within an OSGi framework (JVM)

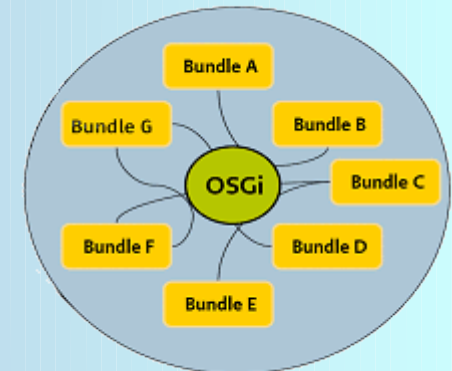
Java EE App Server



Everything isolated



OSGi v4.2 Framework



Everything shared

OSGI Bundle types in CICS

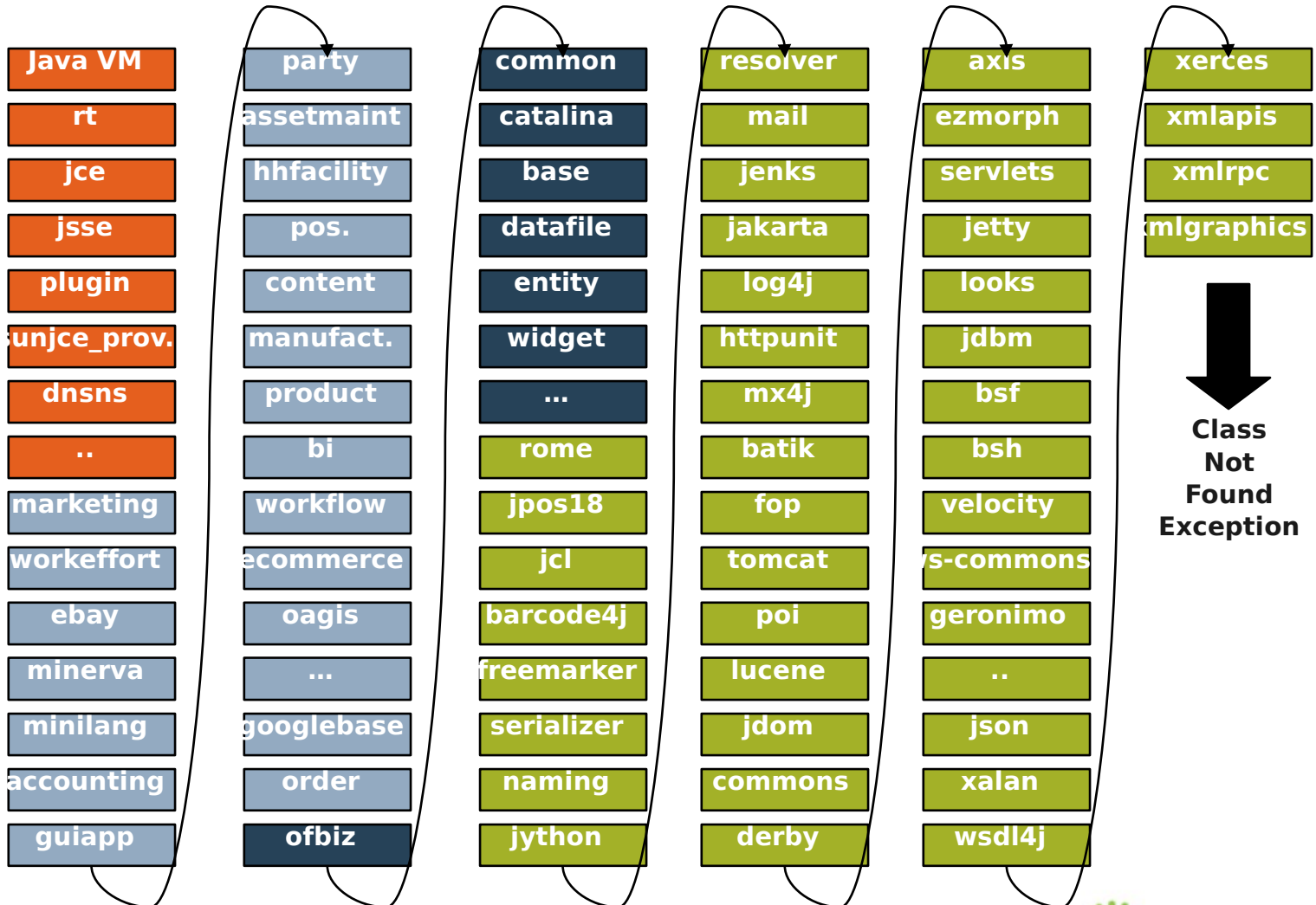
- OSGi Bundles
 - Just a jar with a few extra lines in the jar manifest file
- Application Bundles
 - Provide one or more entry points which can be LINKed too by CICS.
 - This is done by using the CICS-MainClass directive
 - Can import packages from other bundles, i.e. JCICS
- Library Bundles
 - Provide no entry points but simply export code to be used by other bundles
 - Shared library services

Manifest.mf

```
Bundle-SymbolicName: com.ibm.cics.server.examples.hello
Bundle-Version: 1.0.0
...
CICS-MainClass: examples.hello.HelloCICSWorld
Export-Package: my.library.classes 1.0.0
```

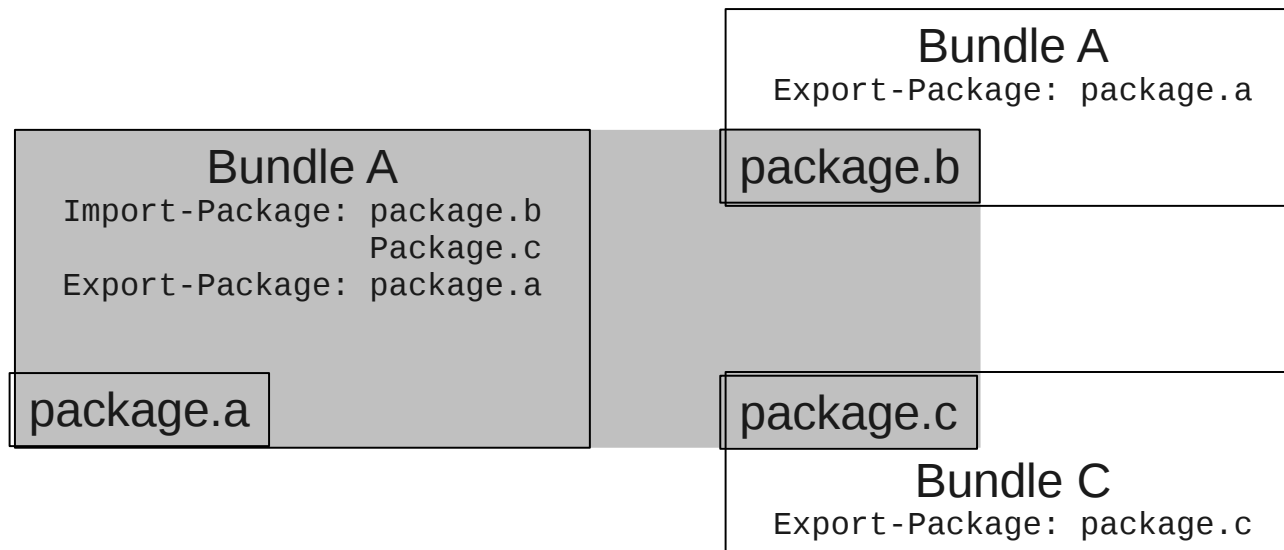
The Global Classpath

Begin Here →

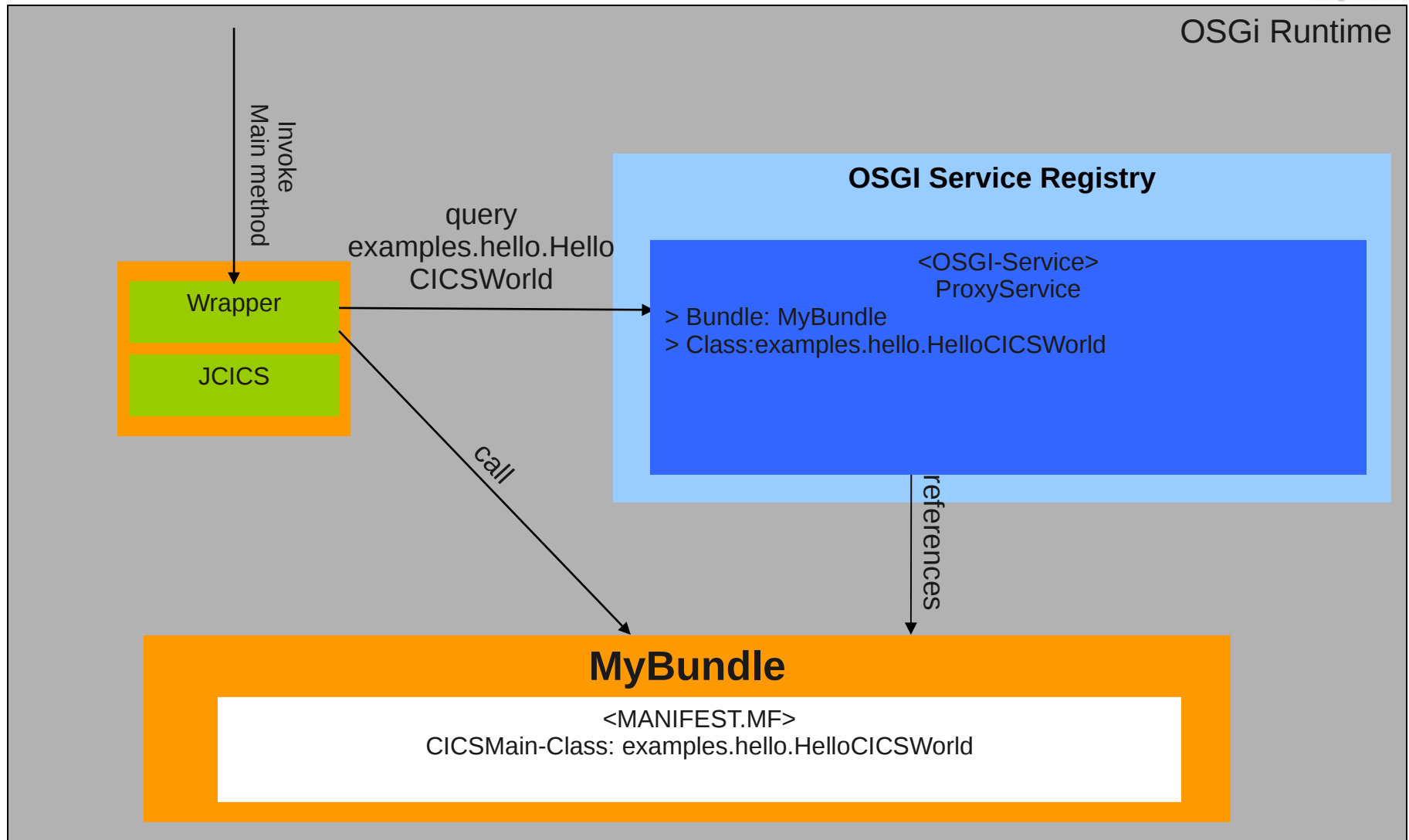


Class loading with OSGi

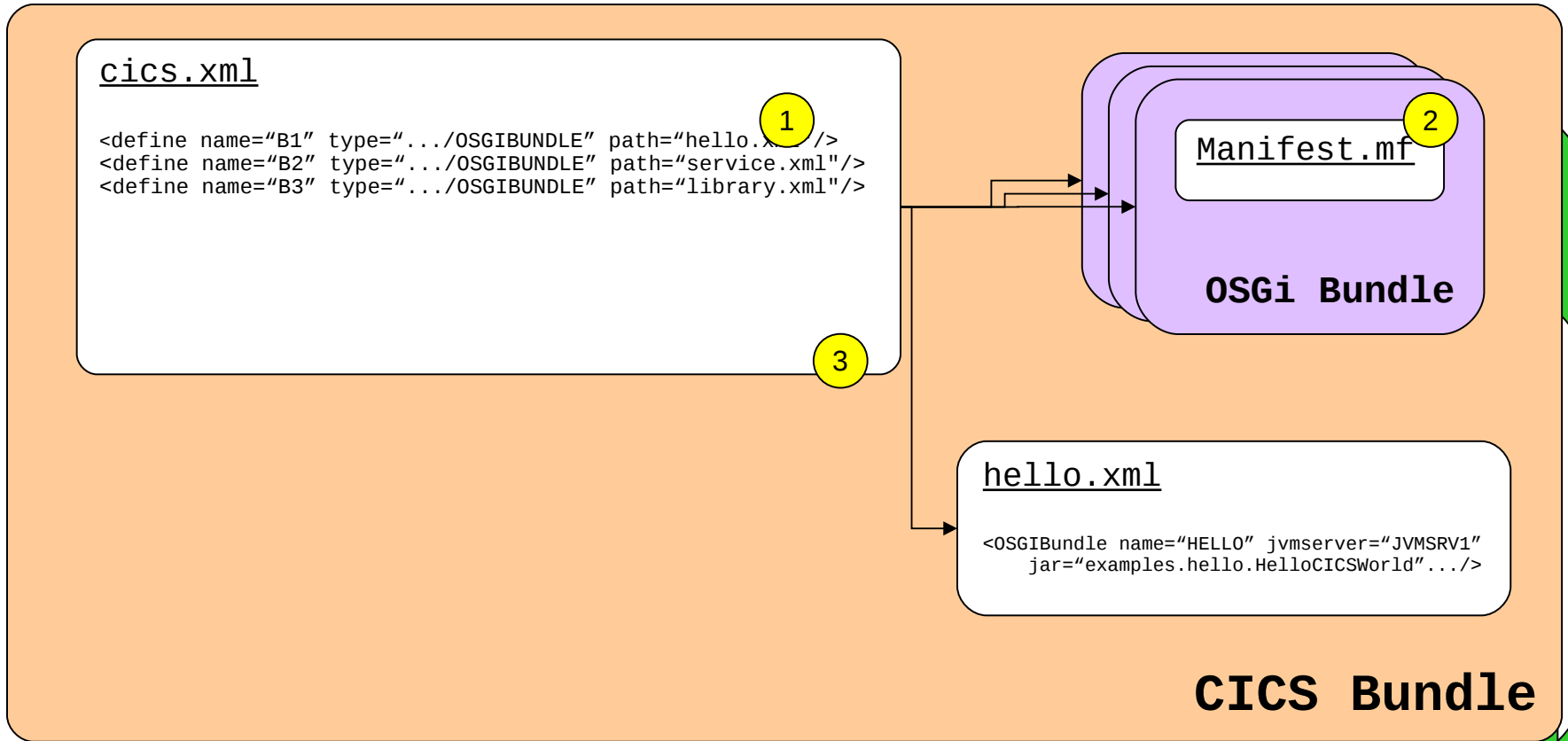
- **No more CLASSPATH**
 - Each bundle has its own class loader
- Class space is the classes required for the bundle
- Smallest unit is a package



JVMSERVER OSGi Details



Deployment with CICS Bundles

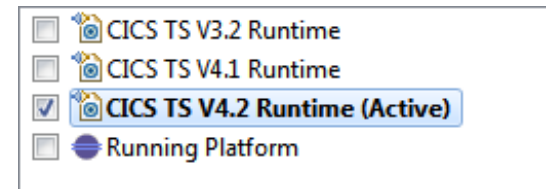


1. Define OSGi bundles
2. Declare PROGRAM "service(s)"
3. Define PROGRAM

Manifest.mf
Bundle-SymbolicName: com.ibm.cics.server.examples.hello
Bundle-Version: 1.0.0
...
CICS-MainClass: examples.hello.HelloCICSWorld

CICS Explorer SDK - Development

- 1) Install CICS Explorer SDK into Eclipse
- 2) Set Target Platform (sets JCICS and JVM levels)



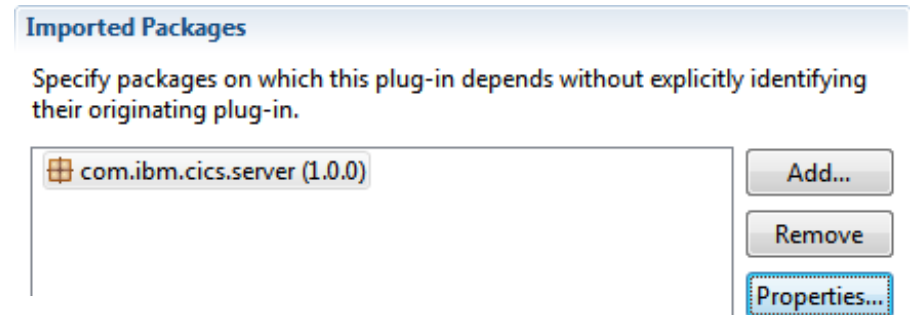
- Window → Preferences... → Target Platform → Add... → Template”

- 3) Create New OSGi Project

- New -> Plug-in Project

- 4) Provided access to JCICS package

- MANIFEST.MF → Dependencies → Imported Packages → com.ibm.cics.server
- Add other bundle imports if required

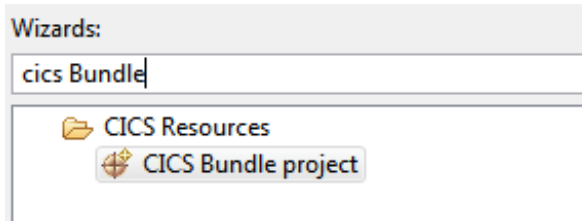


- 5) Import/Create your Java class

CICS Explorer SDK - Deployment

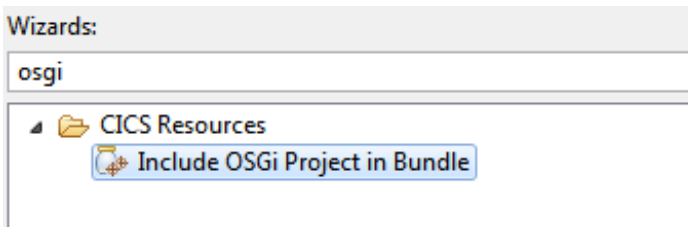
6) Create CICS Bundle

- 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`

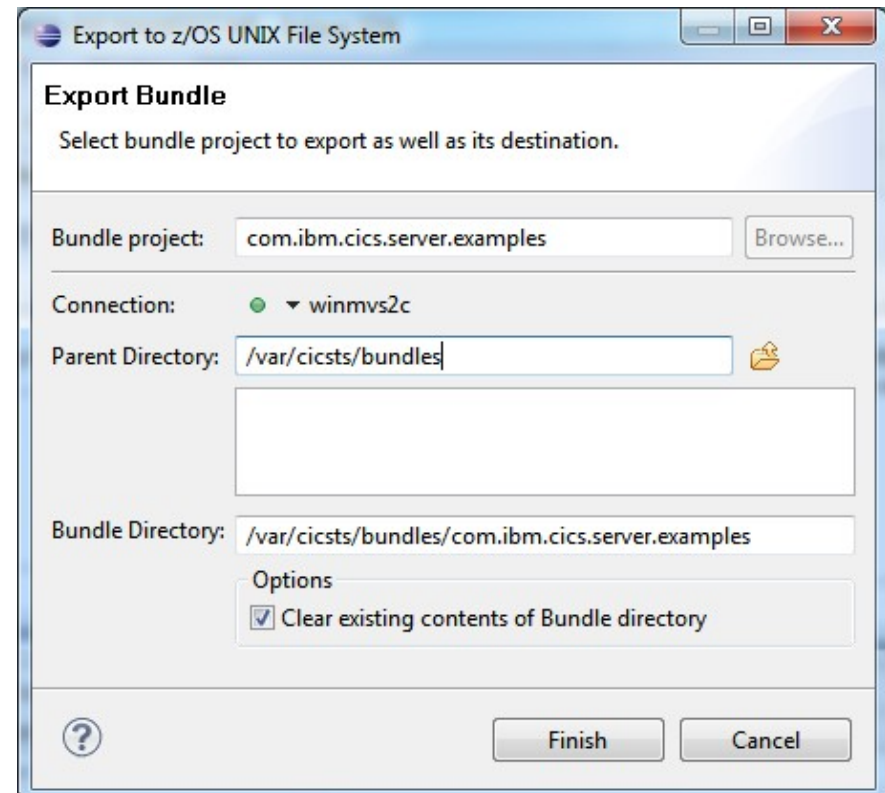
9) Connect CICS Explorer to USS FTP daemon

- Windows -> Open Perspective -> z/OS

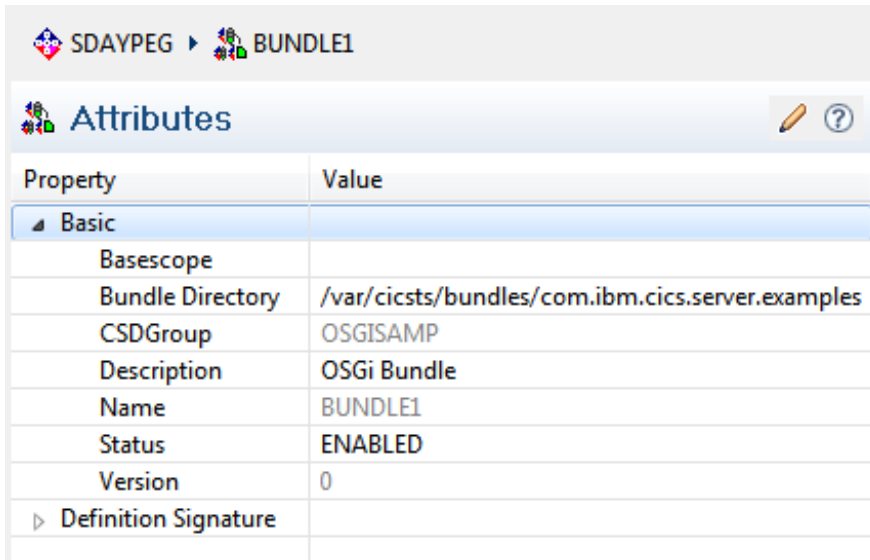
10) Export CICS Bundle to CICS

- -> 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



The screenshot shows the configuration page for a CICS bundle named BUNDLE1. The page is titled "Attributes" and displays a table of properties. The "Basic" section is expanded, showing the following properties:

Property	Value
Basic	
Basescope	
Bundle Directory	/var/cicsts/bundles/com.ibm.cics.server.examples
CSDGroup	OSGISAMP
Description	OSGi Bundle
Name	BUNDLE1
Status	ENABLED
Version	0
Definition Signature	

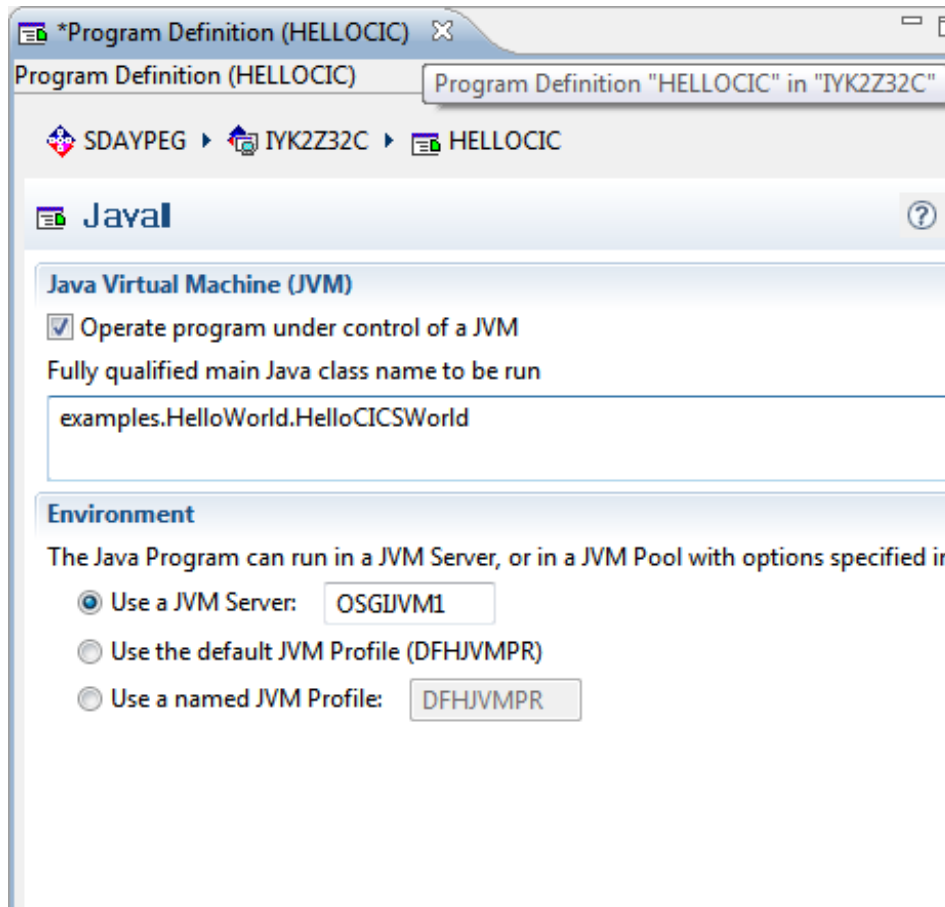
■ Bundle Directory

- Name of directory containing deployed JAR and bundle meta data files

■ Status

- ENABLED → Activate on install of resource

Defining a Program to run in JVMSERVER



- 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)

CICS Bundle Operations

BUNDLE
 BUNDLEDIR=/var/cicsts/bundledir
 com.ibm.cics.bundle

INSTALL
ENABLE
DISABLE
DISCARD

BUNDLEPART - B1
PARTYPE=OSGIBUNDLE

BUNDLEPART - B1
PARTYPE=OSGIBUNDLE

PROGRAM
 Java Class = mypkg.ClassB1
 JVMSERVER=MYJVM

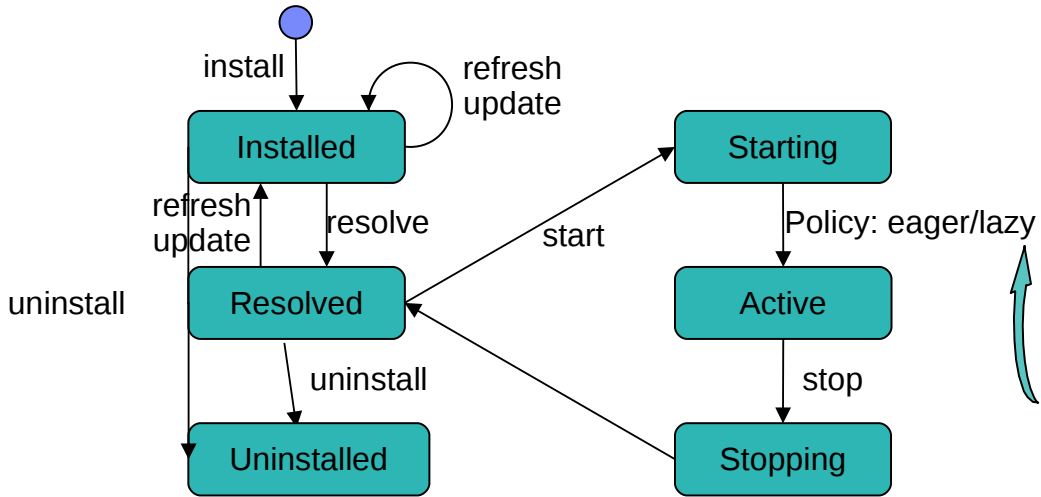
INSTALL
ENABLE
DISABLE
DISCARD

OSGISERVICE

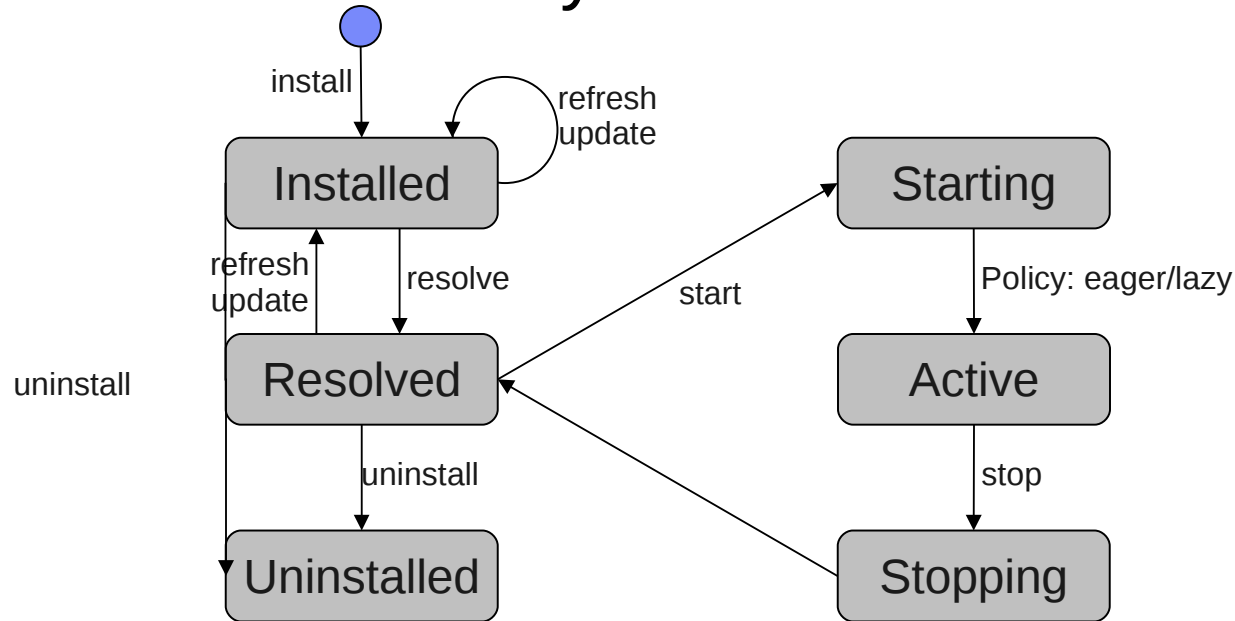
CICS-MainClass: mypkg.ClassB1

OSGIBUNDLE

Note: 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

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

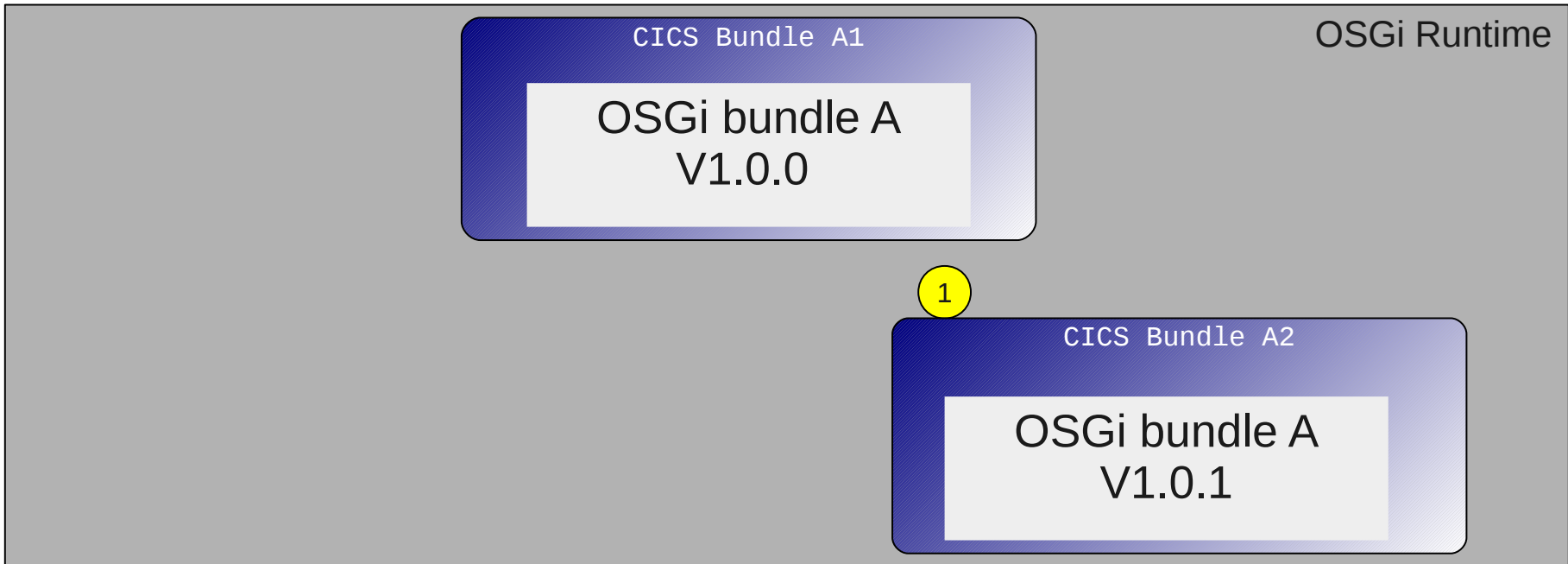
BUNDLE	BUNDLEPART	OSGIBUNDLE	OSGISERVICE
DISABLING	DISABLING	STOPPING	n/a
DISABLED	DISABLED	INSTALLED RESOLVED UNINSTALLED ²	n/a
	UNUSABLE	n/a	n/a
ENABLING	ENABLING	n/a	n/a
ENABLED	ENABLED	STARTING ¹	n/a
		ACTIVE	ACTIVE INACTIVE ³

1 – Bundle activation policy = lazy

2 – Transitory state during termination

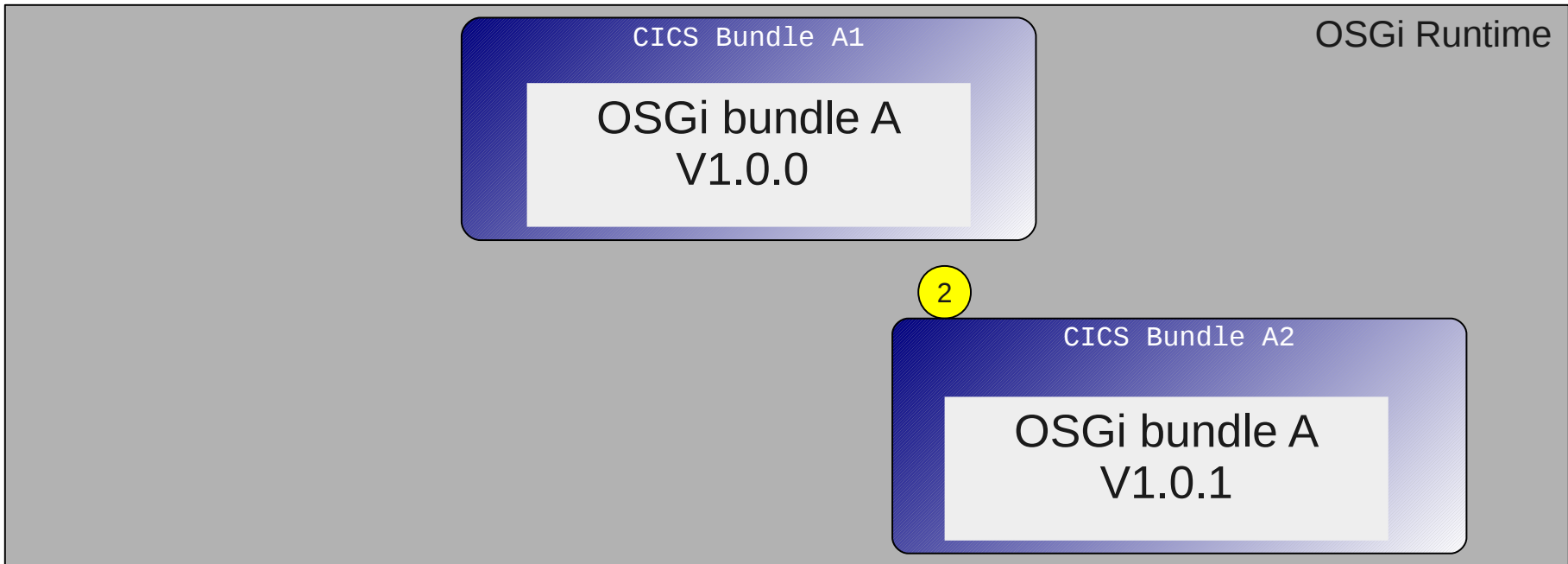
3 - Inactive OSGIService if duplicates existing active OSGI service or Main class invalid

OSGi Versioning Procedure – Application Bundles



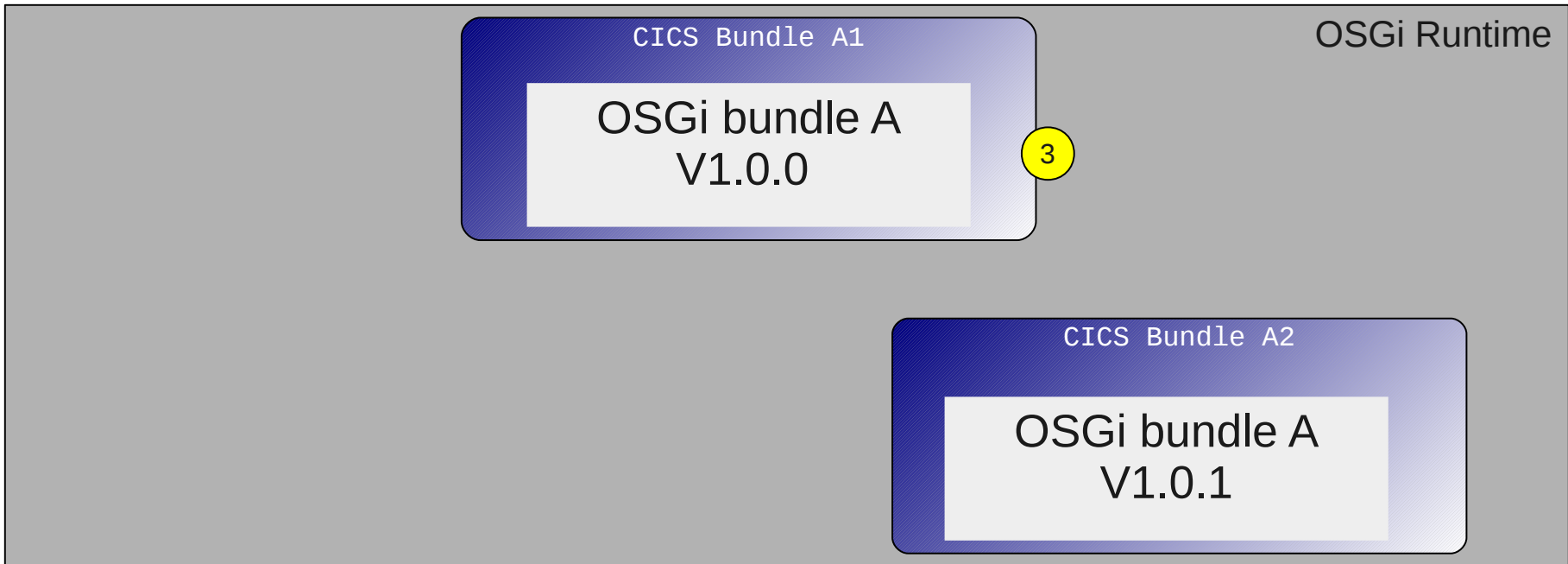
1. Install new CICS Bundle A2 containing new version of OSGi Bundle A

OSGi Versioning Procedure – Application Bundles



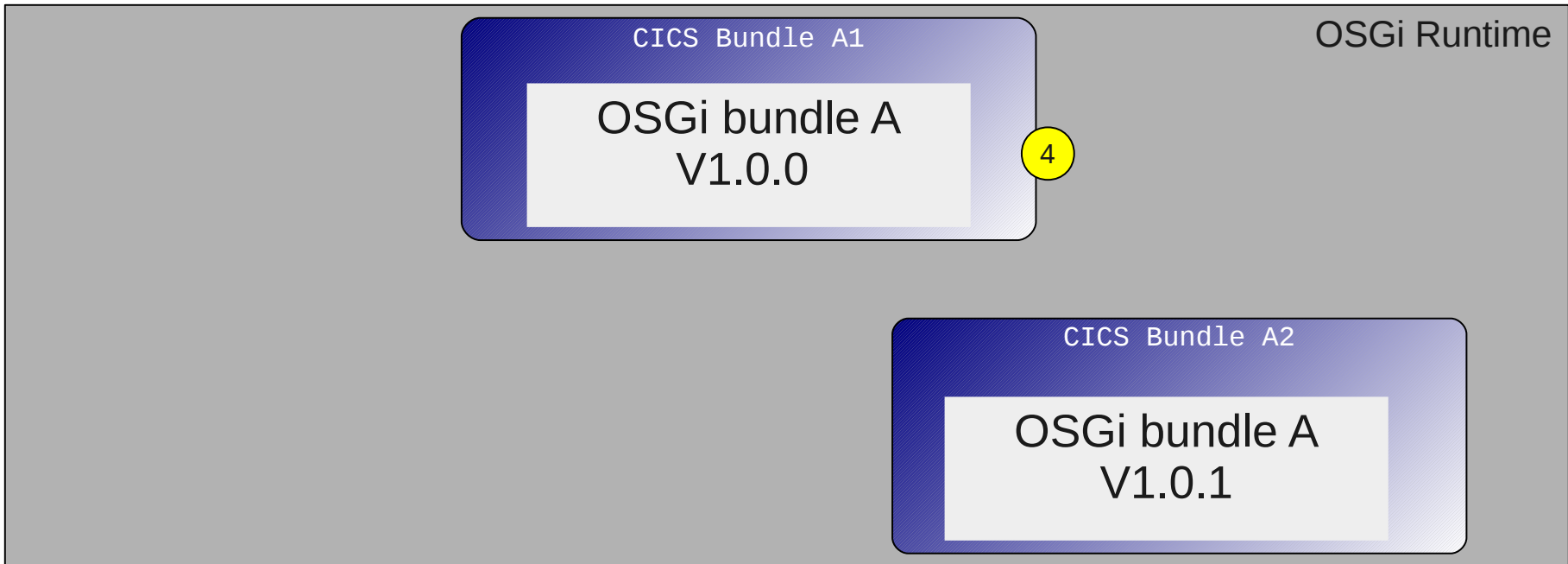
1. Install new CICS Bundle A2 containing new version of OSGi Bundle A
2. Enable CICS Bundle A2
 - OSGi bundle Resolved
 - OSGi Service/Program *Inactive* as duplicates A1

OSGi Versioning Procedure – Application Bundles



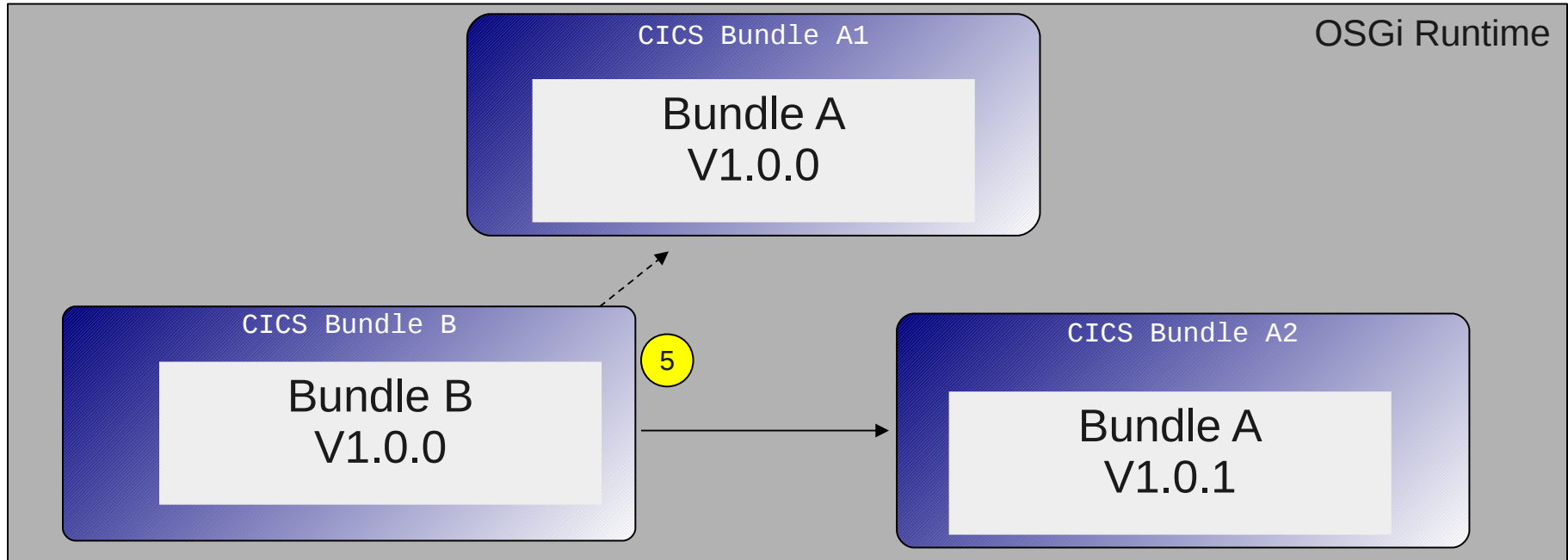
1. Install new CICS Bundle A2
2. Enable new CICS Bundle A2
3. Disable old CICS Bundle A1
 - OSGi Service *Active*
 - New application used for new transactions

OSGi Versioning Procedure – Application Bundles



1. Install new CICS Bundle A2
2. Enable new CICS Bundle A2
3. Disable old CICS Bundle A1
4. Uninstall old CICS Bundle A1

OSGi Versioning Procedure – Library Bundles



1. Install new CICS Bundle A2
2. Enable new CICS Bundle A2
3. Disable old CICS Bundle A1
4. Uninstall old CICS Bundle A1
5. If bundle A is imported from bundle B then recycle (disable/install) bundle B to refresh the classloader for bundle B

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). **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.

Java Web Services with Axis2

Java Web Services with Axis2



- New Java application handler for provider mode
 - Configuration
- New support for provider mode Java application
 - Configuration
 - Application preparation

Provider Mode Java Application Handler



- New application handler written in Java
 - Use is optional
 - Executes in a JVMSERVER
 - 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/>*

Provider Mode Java Application Handler...

- New Java CICS provided application handler

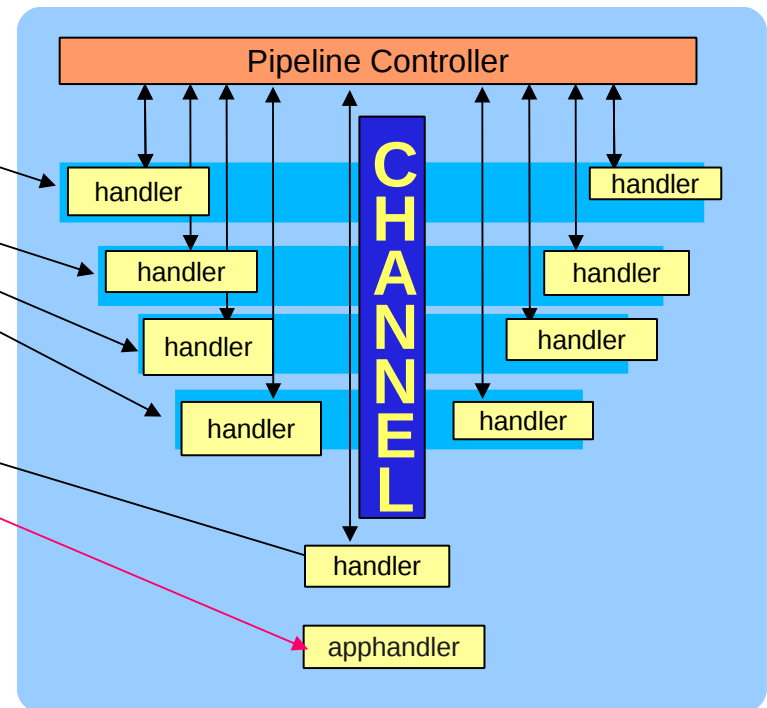
```
<transport>
:
</transport>
```

```
<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>
```



Provider Mode Java Application Handler...



- 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*

Provider Mode Java Application Handler...

- Example pipeline configuration file:

```
<service>
  <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>MYJVM</jvmserver>
      <repository>/u/zem/wsdll/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...



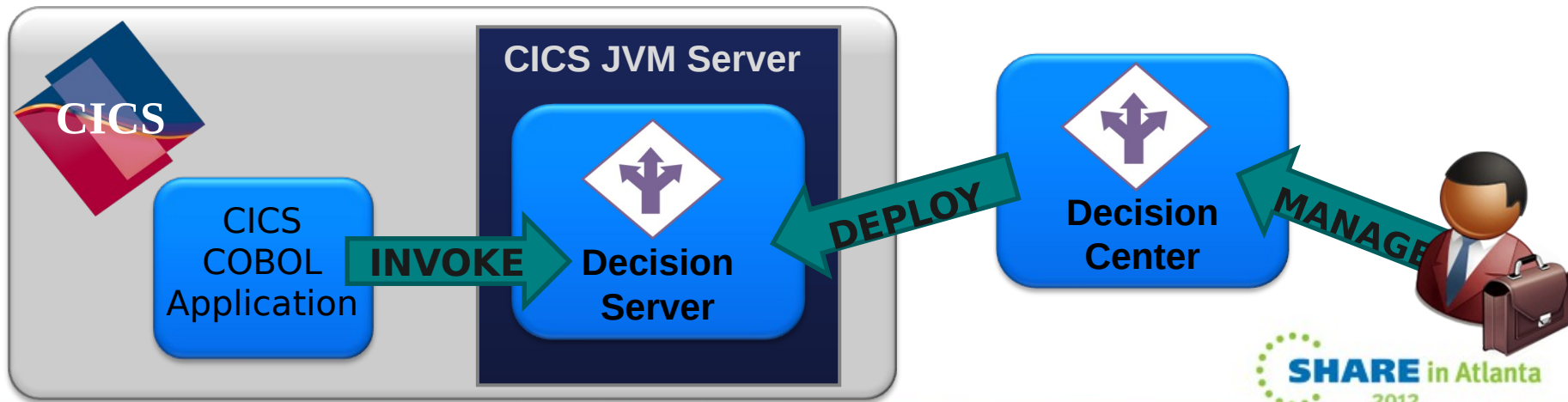
- CICS Services replaced by Java services
 - Axis2 applications interact with CICS with the Axis2 programming model
 - Some CICS services are not applicable
 - *SOAPFAULT CREATE*
 - *WSACONTEXT GET*
 - *DFHWS-OPERATION container*
 - *DFHWS-MEP container*
 - *DFHWS-USERID container*
 - *DFHWS-TRANID container*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

WODM Rules Execution Engine in CICS JVM Server

WebSphere Operational Decision Management & CICS

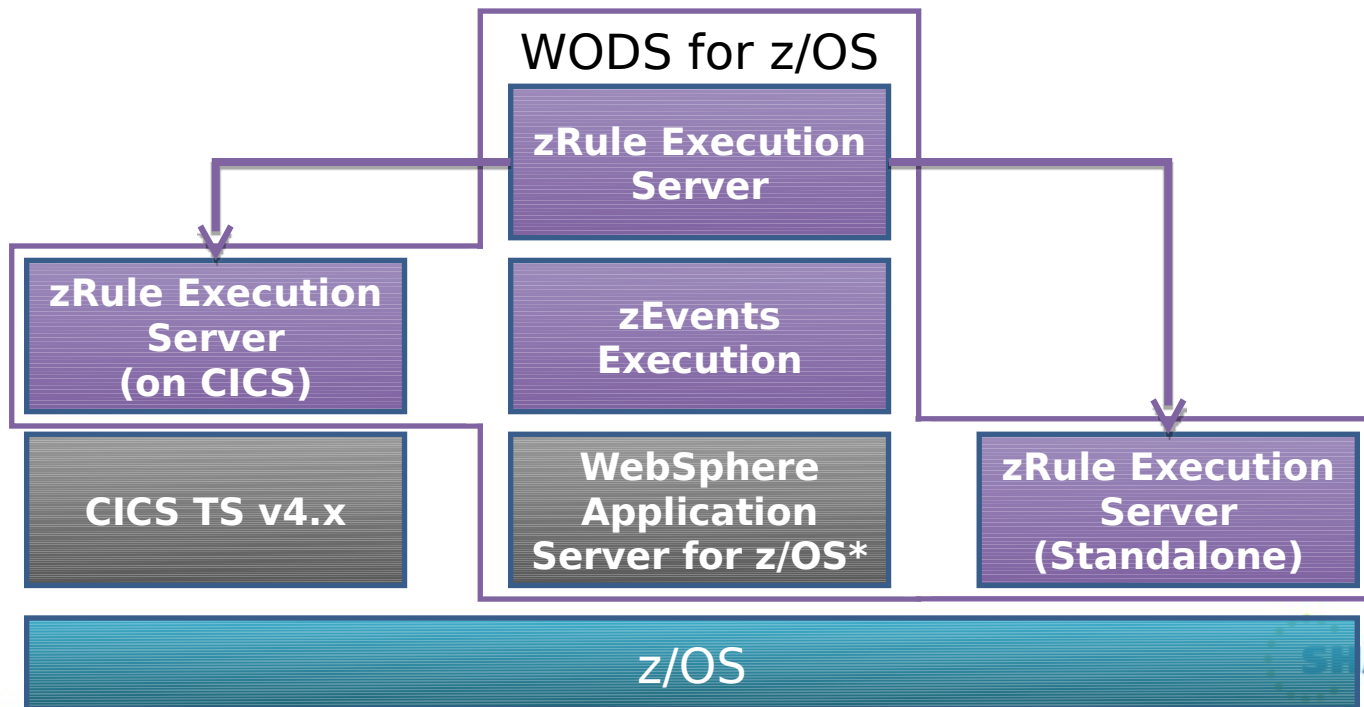
Externalize embedded business rule logic & execute within CICS

- *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



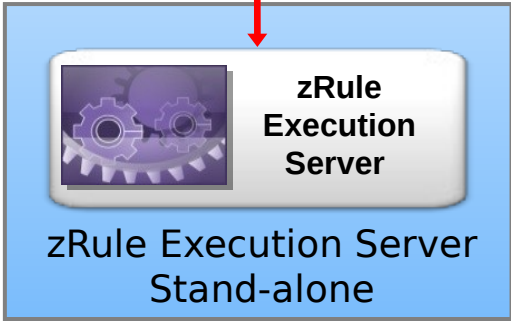
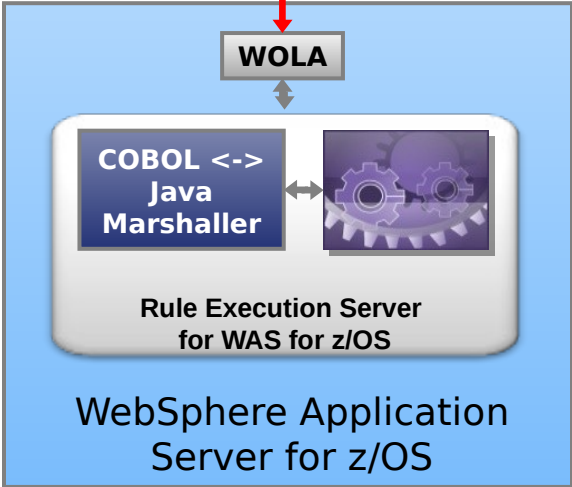
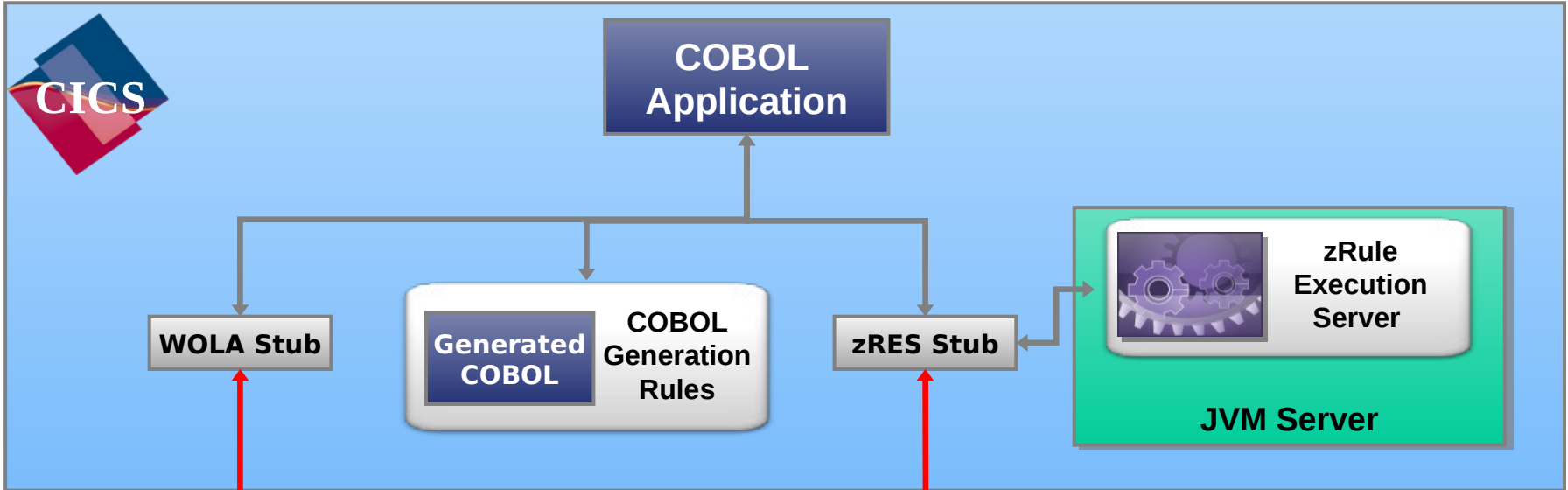
Decision Server for z/OS

- 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 4.x JVMServer environment

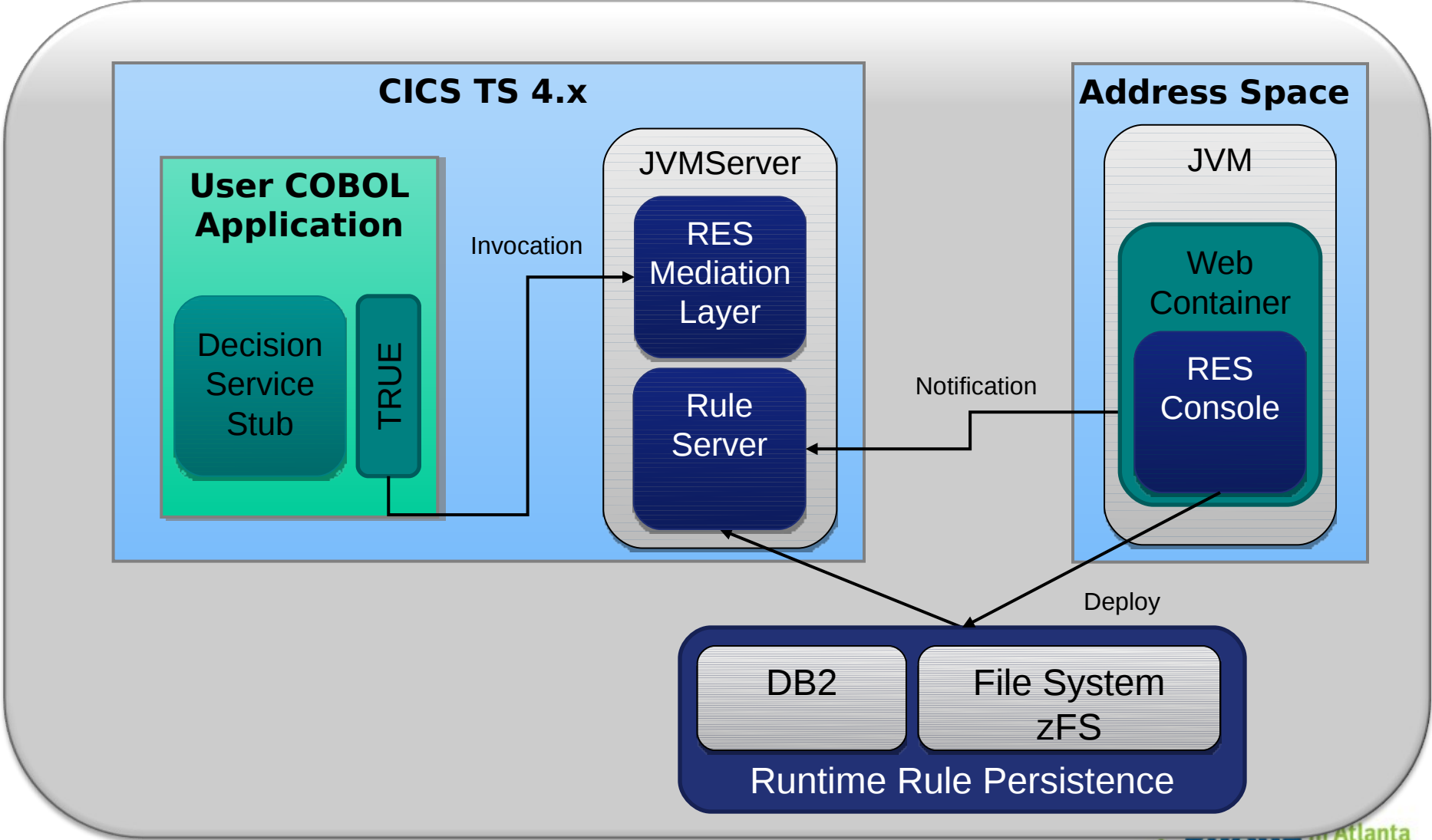


*OEM

Rule invocation options for CICS



zRule Execution Server for z/OS – CICS TS 4.x



Summary



- JVM Options in CICS TS 4.2
 - JVM Pool
 - JVMSERVER
- 64 Bit JVM Support
- OSGi for application management
- Java Web Services with Axis2
- WODM Rules Execution Engine

© IBM Corporation 2012. All Rights Reserved.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.