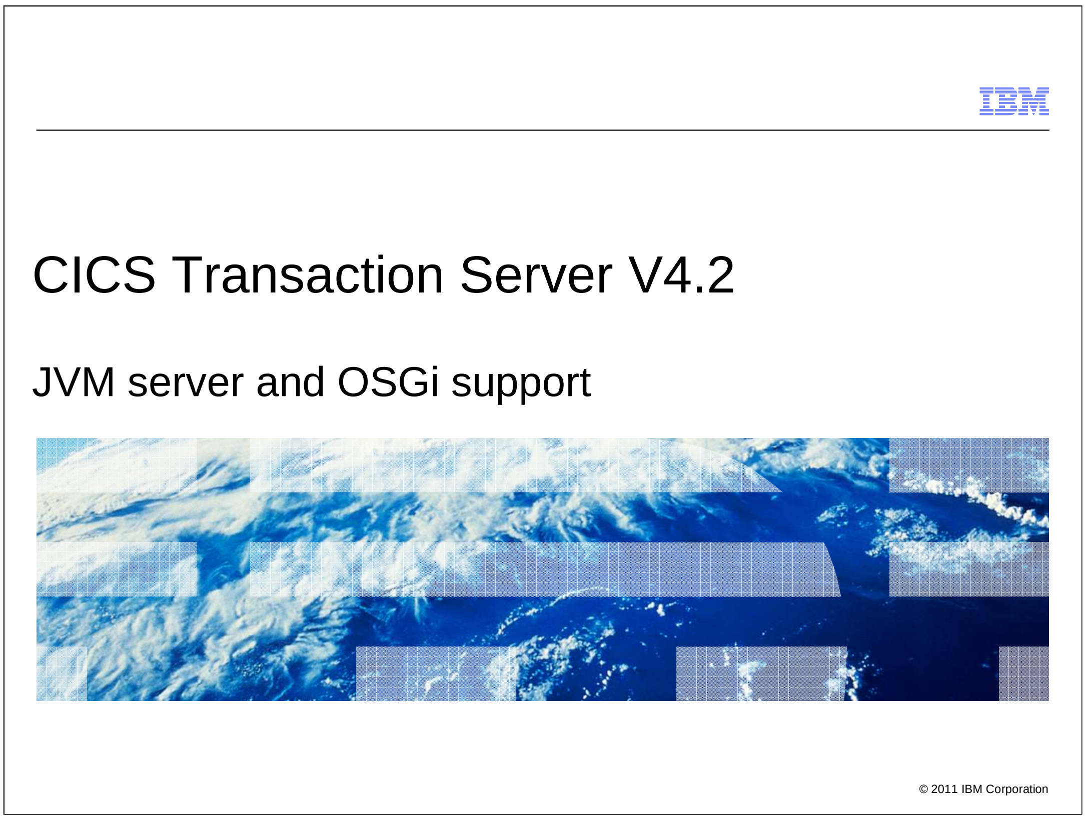


CICS Transaction Server V4.2

JVM server and OSGi support

CICS Transaction Server Version 4.2 has significantly enhanced its Java support to provide a scalable environment that lifts some of the restrictions of running Java applications in CICS regions.

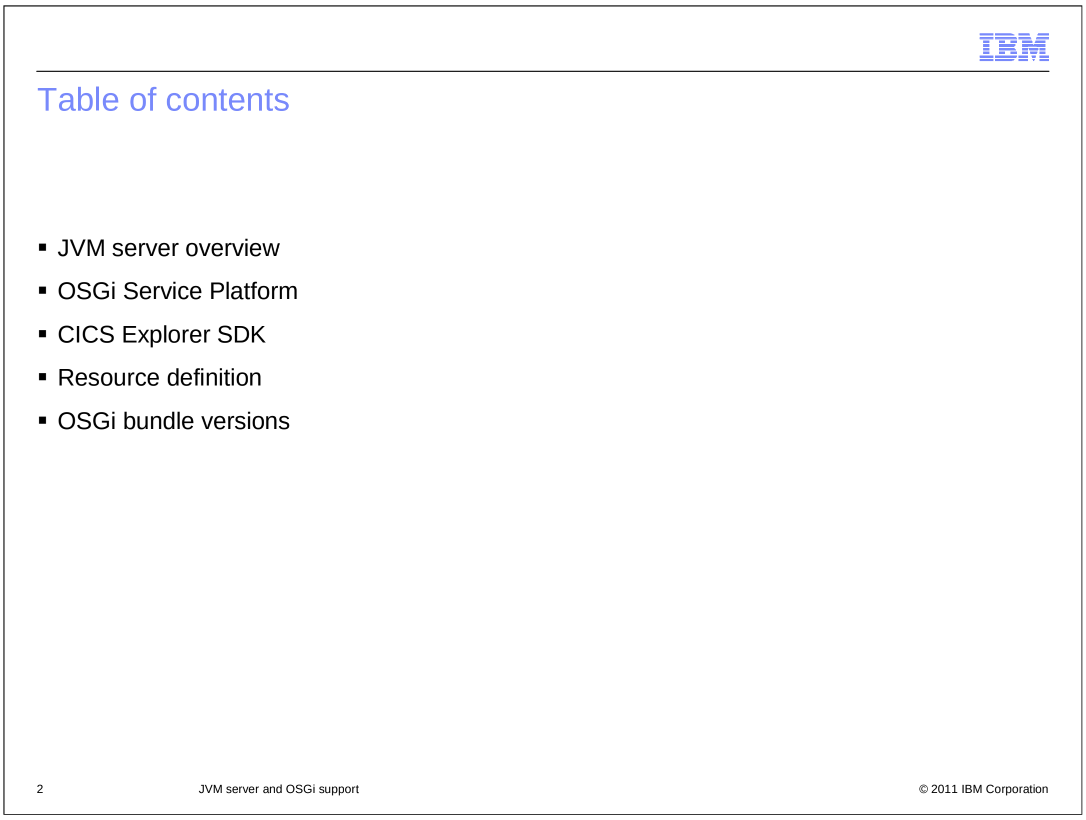
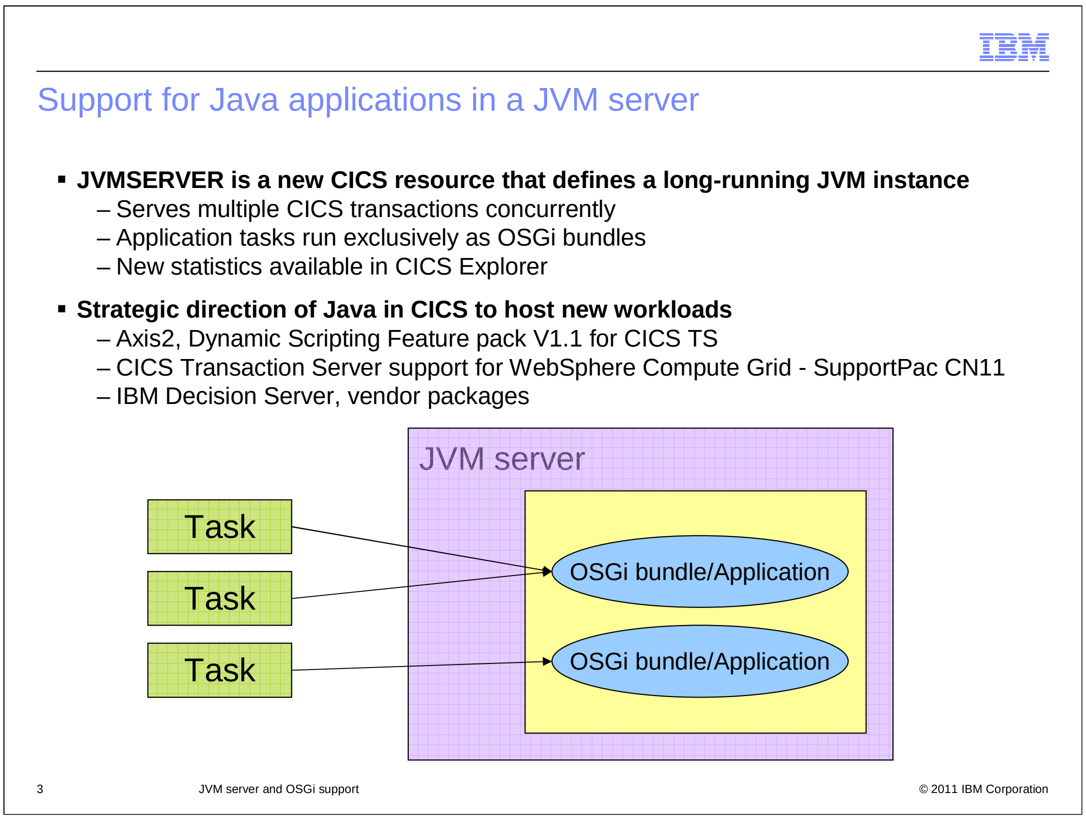


Table of contents

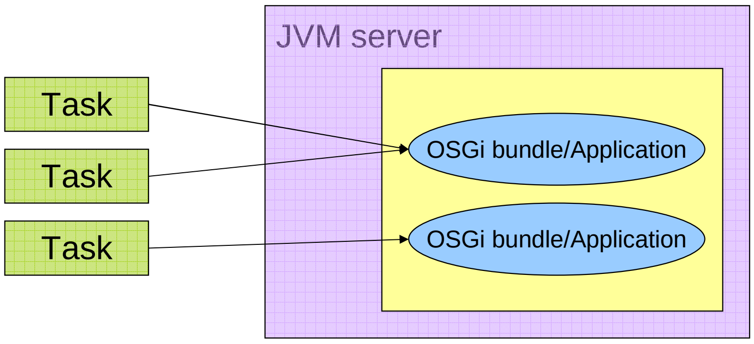
2 JVM server and OSGi support © 2011 IBM Corporation

This module provides an overview of the JVM server, the OSGi Service Platform, and how to develop and deploy Java applications to a JVM server.



Support for Java applications in a JVM server

- **JVMSERVER is a new CICS resource that defines a long-running JVM instance**
  - Serves multiple CICS transactions concurrently
  - Application tasks run exclusively as OSGi bundles
  - New statistics available in CICS Explorer
- **Strategic direction of Java in CICS to host new workloads**
  - Axis2, Dynamic Scripting Feature pack V1.1 for CICS TS
  - CICS Transaction Server support for WebSphere Compute Grid - SupportPac CN11
  - IBM Decision Server, vendor packages

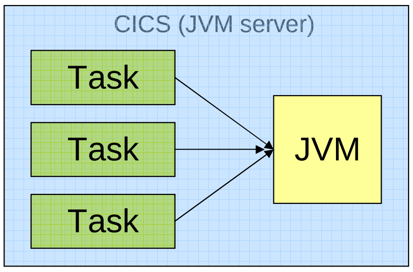


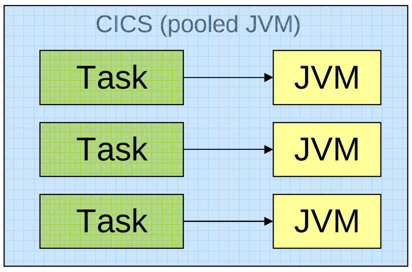
3 JVM server and OSGi support © 2011 IBM Corporation

The JVM server is a runtime environment for running multiple CICS transactions concurrently in the same JVM. It is represented by the JVMSERVER resource. Applications can run in the same JVM server if they are threadsafe and packaged as OSGi bundles. The JVM server contains an OSGi framework to run the OSGi bundles. CICS produces statistics that you can view in CICS Explorer to manage and tune the environment for optimal performance.

The JVM server is the strategic direction for running Java in CICS, including new workloads such as Axis2 for web services and the Dynamic Scripting Feature Pack. The JVM server is also used for the WebSphere Compute Grid supportpac and can be extended to include IBM Decision Server and other external packages.

JVM server versus existing Java support





Single JVM – serves many tasks (reduced storage)

Concurrent, multi-threaded, up to 256 threads per JVM server

T8 (CICS key)

MAXTHRDTCBS automatically calculated, up to maximum of 1024 per region

More standard server model (+ data sharing)

Dynamic update and replace of modules

Pool of JVMs – search serves only a single task

Java program isolation

J8 (CICS key), J9 (User key)

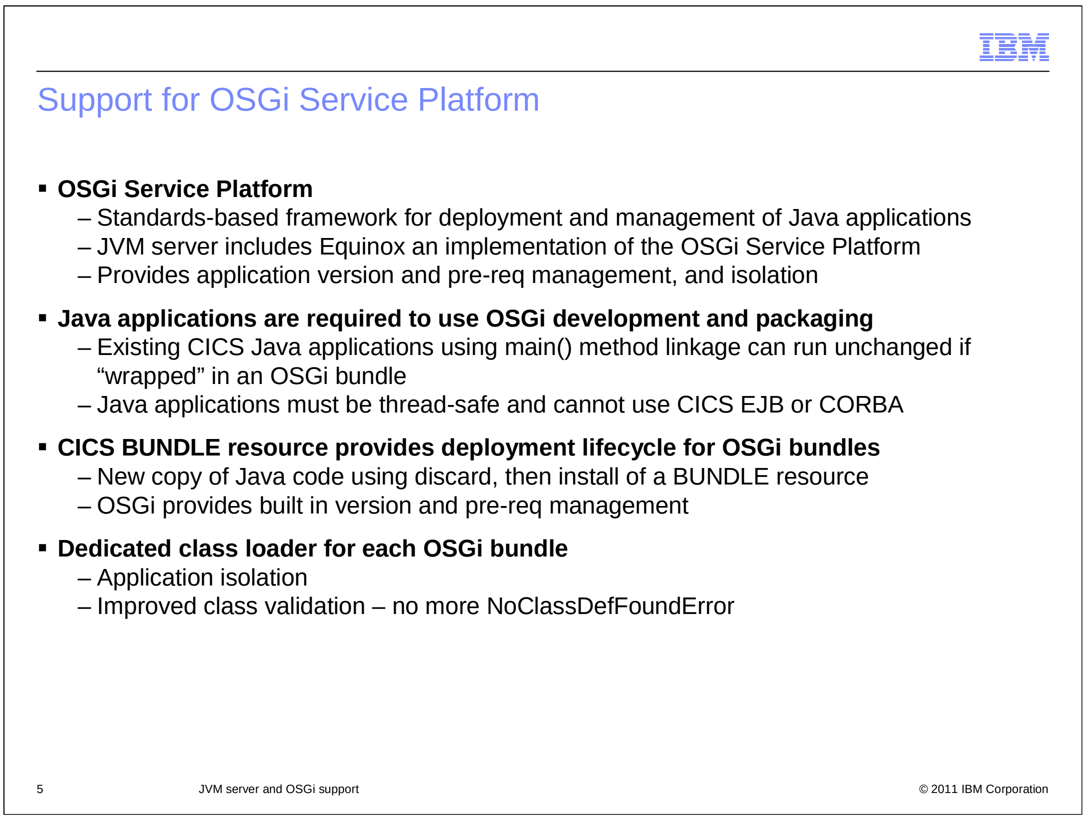
MAXJVMTCBS system initialization parameter

Difficult to share data and state

JVMs must be restarted to effect changes

The JVM server provides many advantages over the existing pooled JVM support. You can use a single JVM to run many tasks, reducing the required storage, as opposed to running many JVMs where each one serves only a single task.

The JVM server provides concurrent, multithreaded support. You can have a maximum of 256 threads in each JVM server. Other differences include the TCBs that are used to run the work and the system initialization parameters that are required. One of the main advantages is that the JVM server is more like a standard server model that can perform data sharing, whereas in a pooled JVM it is difficult to share data and state. Another advantage is that you do not need to restart the JVM server to update and replace application modules. In a pooled environment, you have to restart the JVMs to pick up application changes.



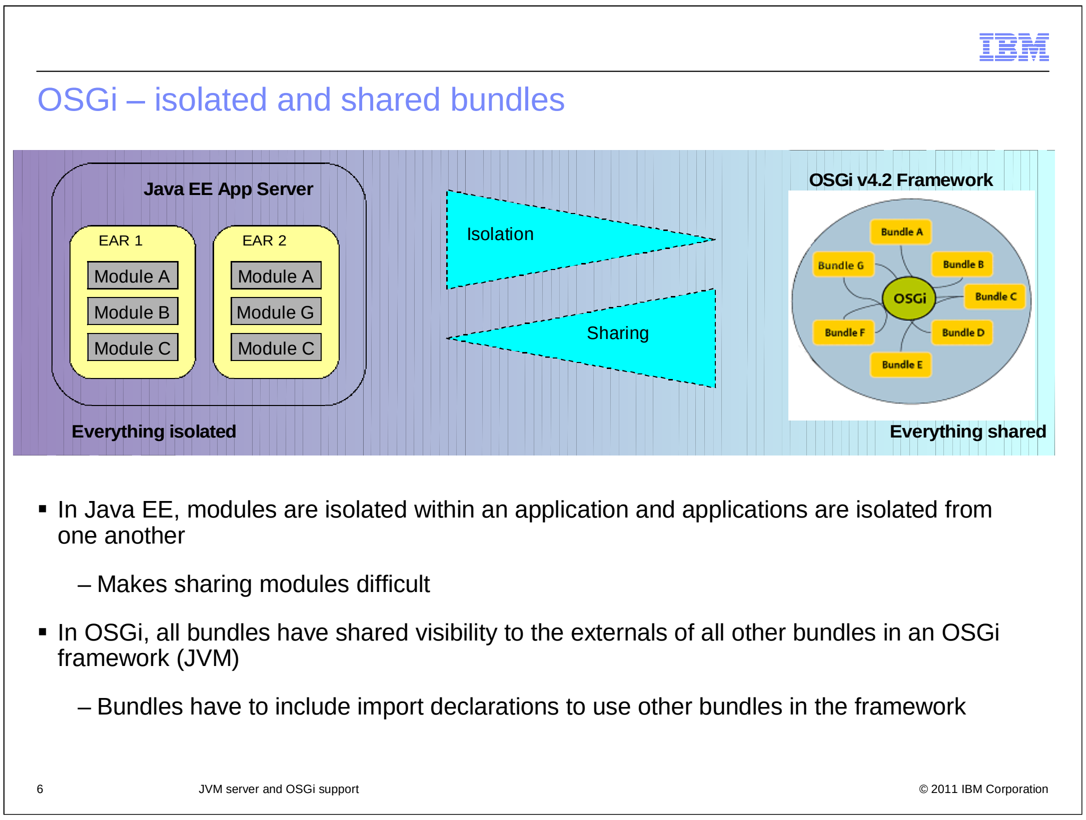
Support for OSGi Service Platform

- **OSGi Service Platform**
  - Standards-based framework for deployment and management of Java applications
  - JVM server includes Equinox an implementation of the OSGi Service Platform
  - Provides application version and pre-req management, and isolation
- **Java applications are required to use OSGi development and packaging**
  - Existing CICS Java applications using main() method linkage can run unchanged if “wrapped” in an OSGi bundle
  - Java applications must be thread-safe and cannot use CICS EJB or CORBA
- **CICS BUNDLE resource provides deployment lifecycle for OSGi bundles**
  - New copy of Java code using discard, then install of a BUNDLE resource
  - OSGi provides built in version and pre-req management
- **Dedicated class loader for each OSGi bundle**
  - Application isolation
  - Improved class validation – no more NoClassDefFoundError

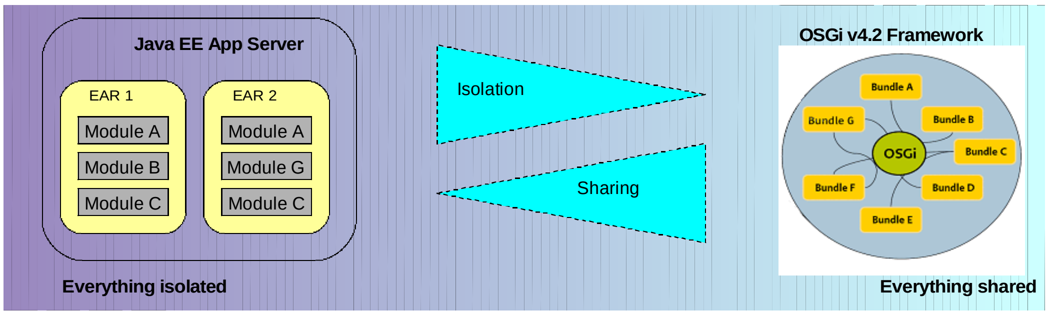
5 JVM server and OSGi support © 2011 IBM Corporation

The OSGi Service Platform provides a standard framework for packaging, deploying, and managing Java applications. The JVM server includes the Equinox implementation of the OSGi Service Platform. OSGi provides advanced application version handling, dependency management, and isolation. To run a Java application in a JVM server, you must use OSGi development and packaging. However, you do not need to make changes to existing applications if they are threadsafe. You can wrap the existing application in an OSGi bundle. You cannot use Enterprise Javabeans or CORBA in a JVM server.

The CICS BUNDLE resource provides lifecycle management of the OSGi bundles, so you can install, disable, and discard a collection of related OSGi bundles together. The OSGi framework resolves dependencies based on the information in the OSGi bundle. Each OSGi bundle also has a dedicated class loader to ensure application isolation and improve class validation.



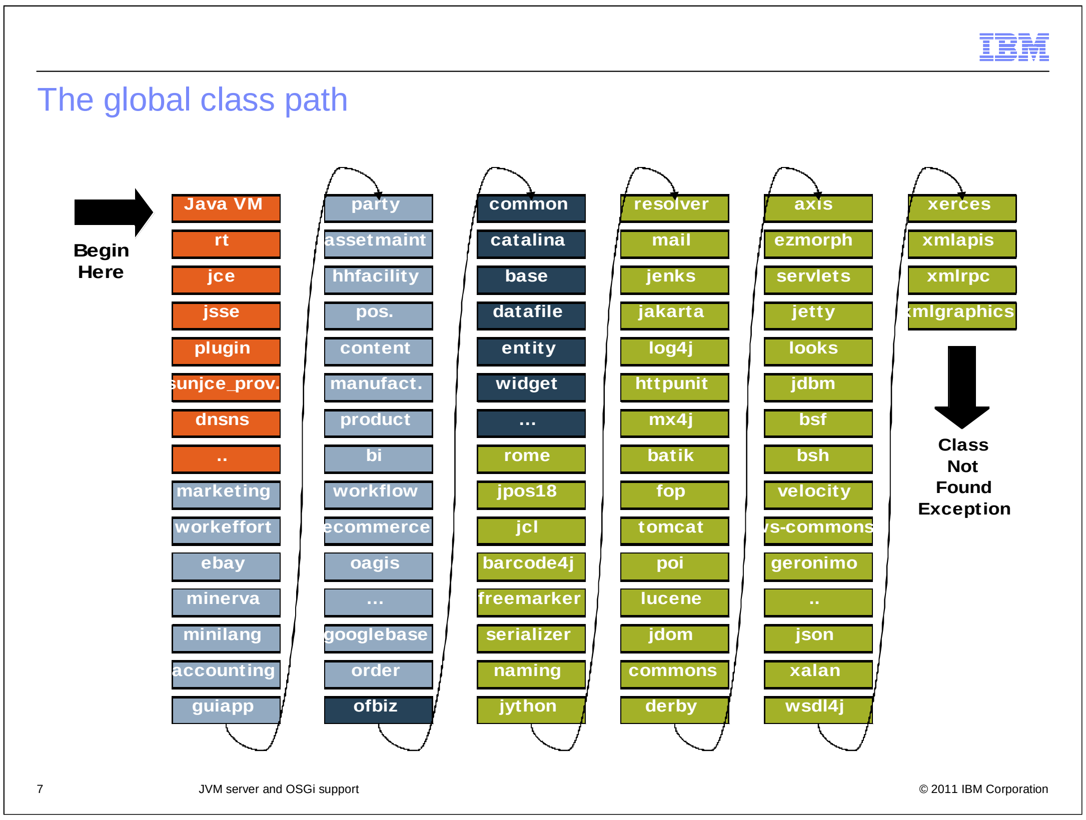
OSGi – isolated and shared bundles



- In Java EE, modules are isolated within an application and applications are isolated from one another
  - Makes sharing modules difficult
- In OSGi, all bundles have shared visibility to the externals of all other bundles in an OSGi framework (JVM)
  - Bundles have to include import declarations to use other bundles in the framework

6 JVM server and OSGi support © 2011 IBM Corporation

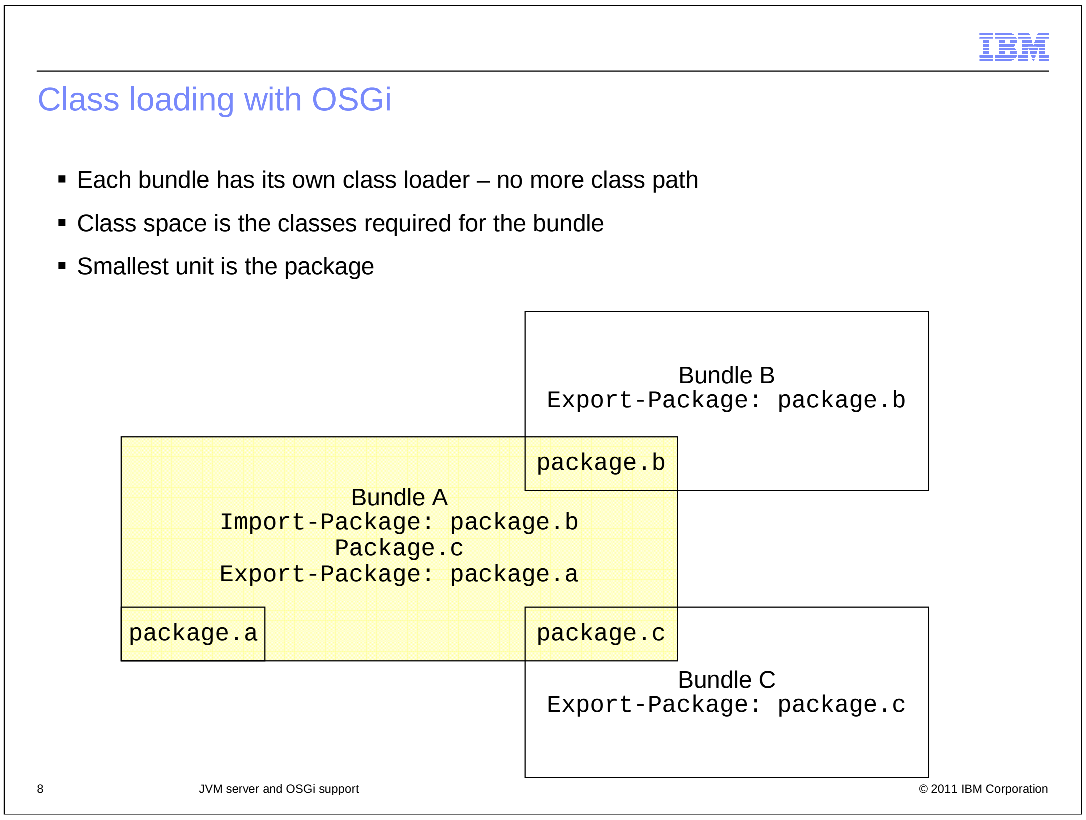
Unlike Java EE applications, OSGi based Java applications allow an extensive and flexible reuse of Java components. Any Java class loaded in an OSGi bundle can be shared to any other bundle residing in the same OSGi framework. In fact, while Java EE ensures isolation between EAR applications loading their classes on isolated class loaders, in the OSGi architecture bundle class loaders can share their classes. The application developer has to explicitly add import statements to use packages from other bundles at run time.



The global class path

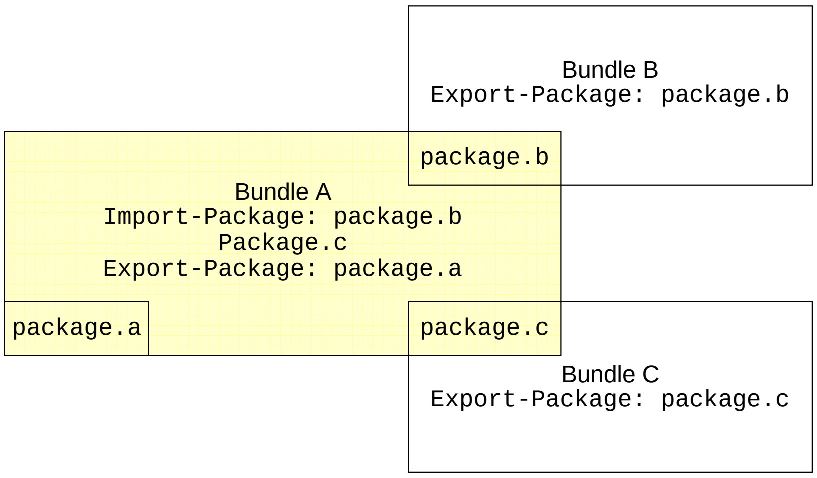
7 JVM server and OSGi support © 2011 IBM Corporation

Java applications that use a class path have three main limitations. Firstly, class loading is expensive because the class loaders architecture is hierarchical and the algorithm to find and load the class is a delegation to the parent class loader. Therefore, classes in the class path that are loaded by the System Class Loader at the bottom of the class loaders hierarchy, are first searched in the Extension Class Loader, then in the Bootstrap Class Loader and then finally in the System Class Loader. Only one version of the same class can be loaded and the order in the class path determines which class is loaded in case of multiple class implementations. Unresolved dependencies between classes are discovered at run time, compromising the application. OSGi solves those limitations. OSGi bundle class loaders are not organized hierarchically and the application immediately determines which bundle class loader to delegate to load the class. Multiple versions of the same class can be shared and referenced and dependencies are both checked and resolved when the bundle is installed.



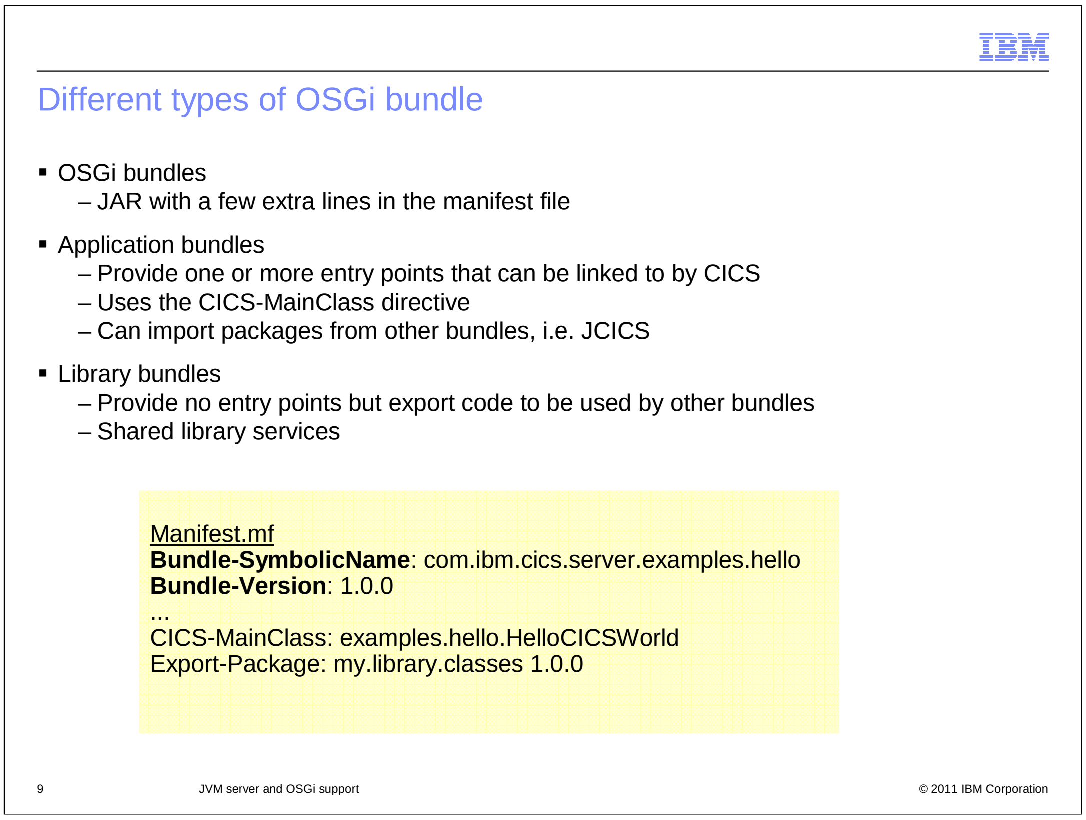
Class loading with OSGi

- Each bundle has its own class loader – no more class path
- Class space is the classes required for the bundle
- Smallest unit is the package


Bundle B
Export-Package: package.b
package.b
Bundle A
Import-Package: package.b
Package.c
Export-Package: package.a
package.a
package.c
Bundle C
Export-Package: package.c

8 JVM server and OSGi support © 2011 IBM Corporation

OSGi bundles use the export package directive in the manifest file to share classes with other bundles and the import package directive to select shared classes from other bundle packages. Export package and import package directives allow application developers to specify a version, or more often a range of versions. Using a range provides more flexibility because the application can export multiple versions of the same class and select which class version is used.



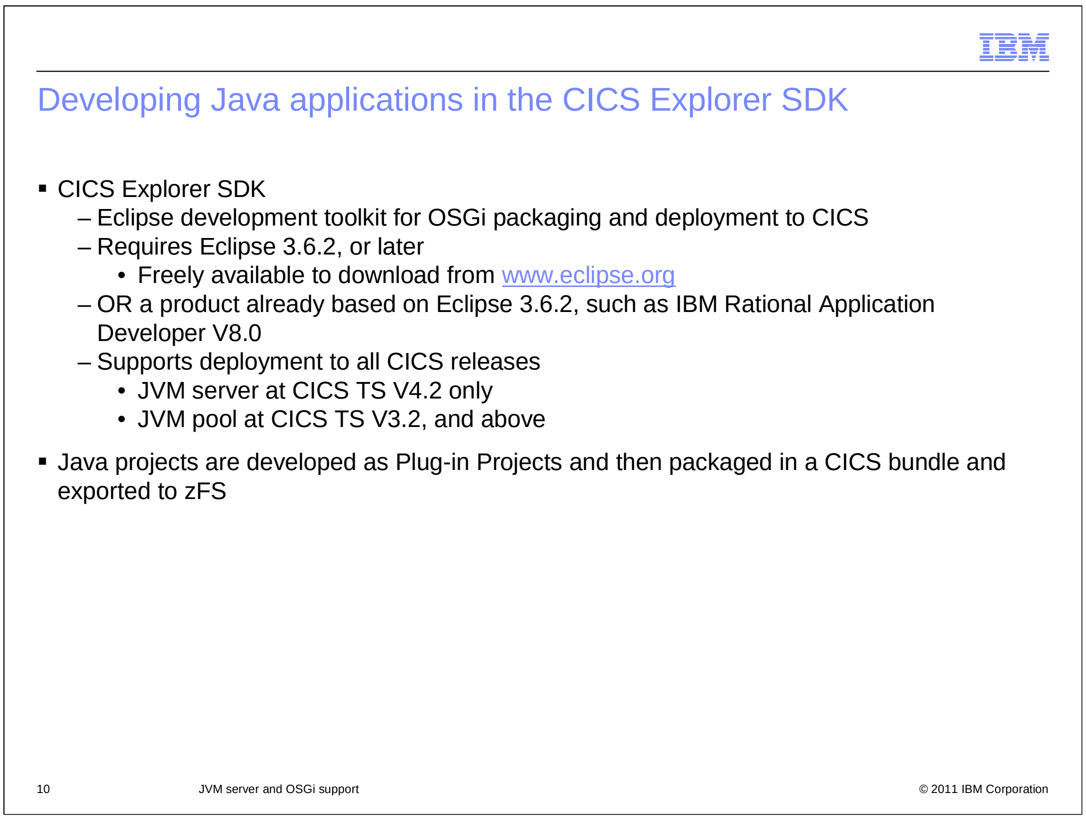
Different types of OSGi bundle

- OSGi bundles
  - JAR with a few extra lines in the manifest file
- Application bundles
  - Provide one or more entry points that can be linked to by CICS
  - Uses the CICS-MainClass directive
  - Can import packages from other bundles, i.e. JCICS
- Library bundles
  - Provide no entry points but 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

9 JVM server and OSGi support © 2011 IBM Corporation

An OSGi bundle is a Java archive file with some additional information in the manifest file. There are two types of OSGi bundles in CICS, application bundles and library bundles. Application bundles are characterized by having the CICS-MainClass directive in the manifest file. The CICS-MainClass directive specifies the list of classes in the bundle whose method “main” can be linked to by CICS using a PROGRAM resource. The references in the CICS-MainClass are registered as OSGi services in the service registry of the OSGi framework. This allows CICS to find the OSGi bundle that contains the class when the application is called at run time. Library bundles do not have classes that can be defined in a PROGRAM resource and made available outside the OSGi framework. Instead a library bundle contains code that is shared by other bundles.



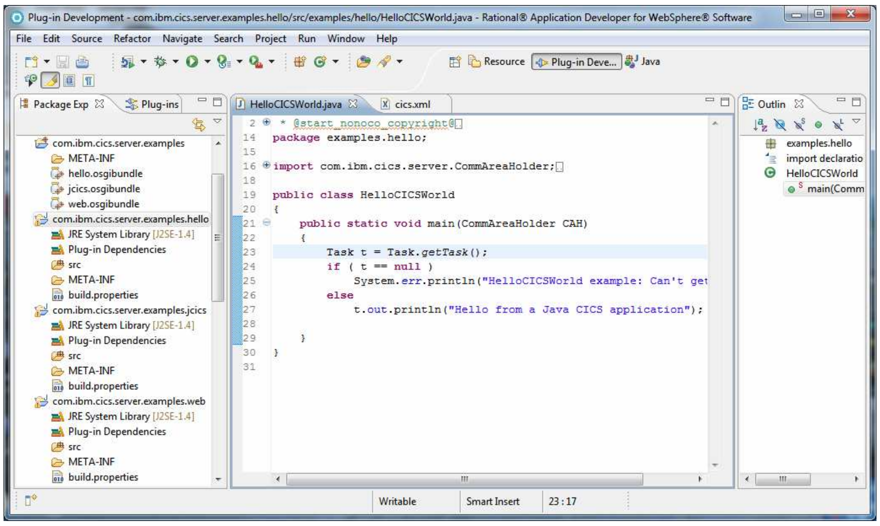
Developing Java applications in the CICS Explorer SDK

- CICS Explorer SDK
  - Eclipse development toolkit for OSGi packaging and deployment to CICS
  - Requires Eclipse 3.6.2, or later
    - Freely available to download from www.eclipse.org
  - OR a product already based on Eclipse 3.6.2, such as IBM Rational Application Developer V8.0
  - Supports deployment to all CICS releases
    - JVM server at CICS TS V4.2 only
    - JVM pool at CICS TS V3.2, and above
- Java projects are developed as Plug-in Projects and then packaged in a CICS bundle and exported to zFS

10 JVM server and OSGi support © 2011 IBM Corporation

To help with developing and deploying OSGi bundles into CICS, the CICS Explorer SDK has been enhanced to provide this support. The CICS Explorer SDK provides application development capabilities and includes support for developing Java applications using the JCICS API and deploying into CICS. You must have an Eclipse Integrated Development Environment at Version 3.6.2 or later. The IDE is freely available from the Eclipse website. Alternatively, you can use a product that is based on Eclipse 3.6.2, such as IBM Rational Application Developer Version 8. The CICS Explorer SDK supports all releases of CICS, so you can develop Java applications for any release. You develop Java projects as plug-in projects in the tool and the package the application in a CICS bundle and export to zFS.

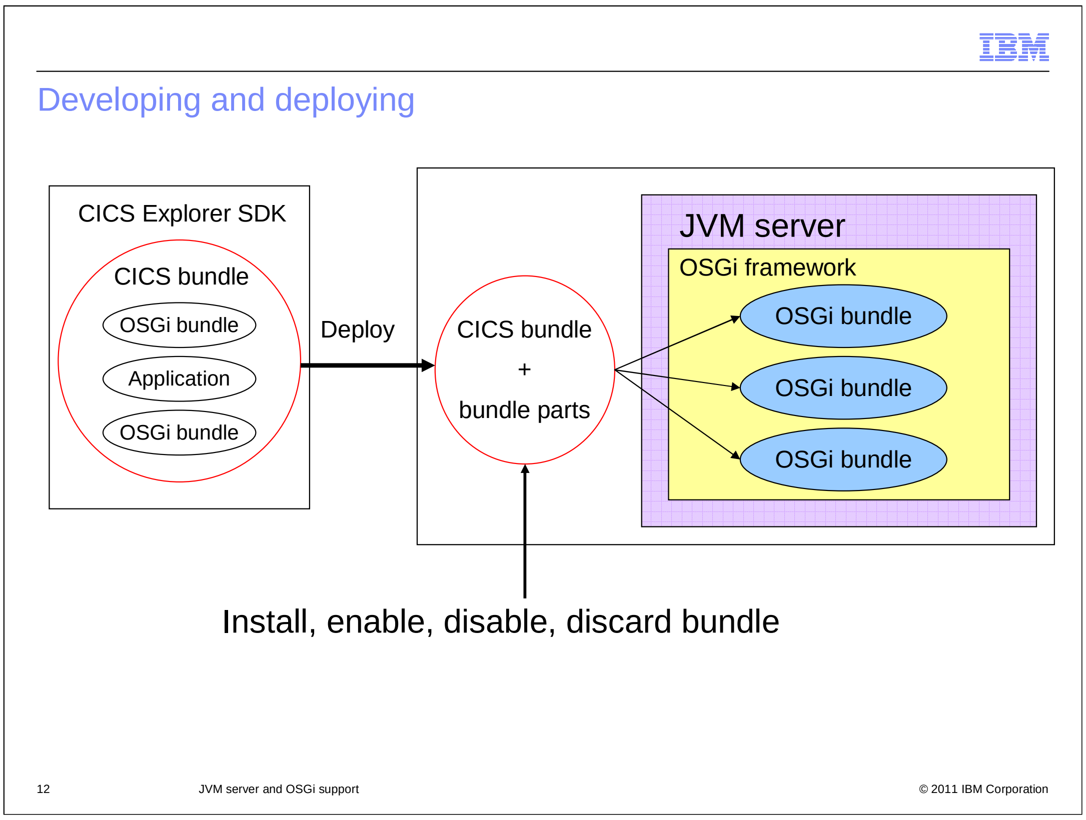
CICS Explorer SDK – for Java development


Plug-in Development - com.ibm.cics.server.examples.hello/src/examples/hello/HelloCICSWorld.java - Rational® Application Developer for WebSphere® Software
File Edit Source Refactor Navigate Search Project Run Window Help
Resource
Plug-in Deve...
Java
Package Exp
Plug-ins
com.ibm.cics.server.examples
META-INF
hello.osgibundle
jcics.osgibundle
web.osgibundle
com.ibm.cics.server.examples.hello
JRE System Library [J2SE-1.4]
Plug-in Dependencies
src
build.properties
com.ibm.cics.server.examples.jcics
com.ibm.cics.server.examples.web
HelloCICSWorld.java
cics.xml
Outlin
examples.hello
import declaratio
HelloCICSWorld
main(Comm
Writable
Smart Insert
23 : 17

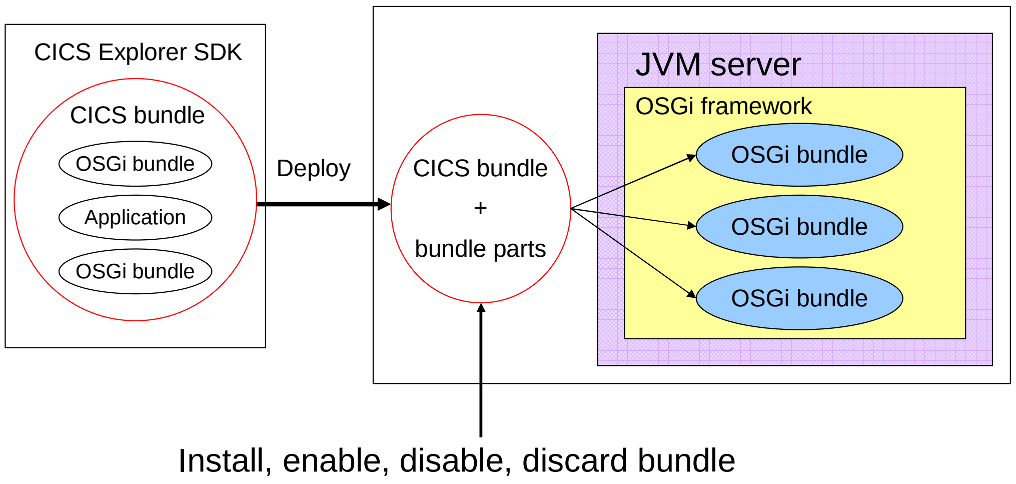
```
 * @start_nonoco_copyright@
package examples.hello;

import com.ibm.cics.server.CommAreaHolder;

public class HelloCICSWorld
{
    public static void main(CommAreaHolder CAH)
    {
        Task t = Task.getTask();
        if ( t == null )
            System.err.println("HelloCICSWorld example: Can't get
        else
            t.out.println("Hello from a Java CICS application");

    }
}
```

- CICS Explorer SDK now provides complete toolkit for developing and deploying CICS Java applications

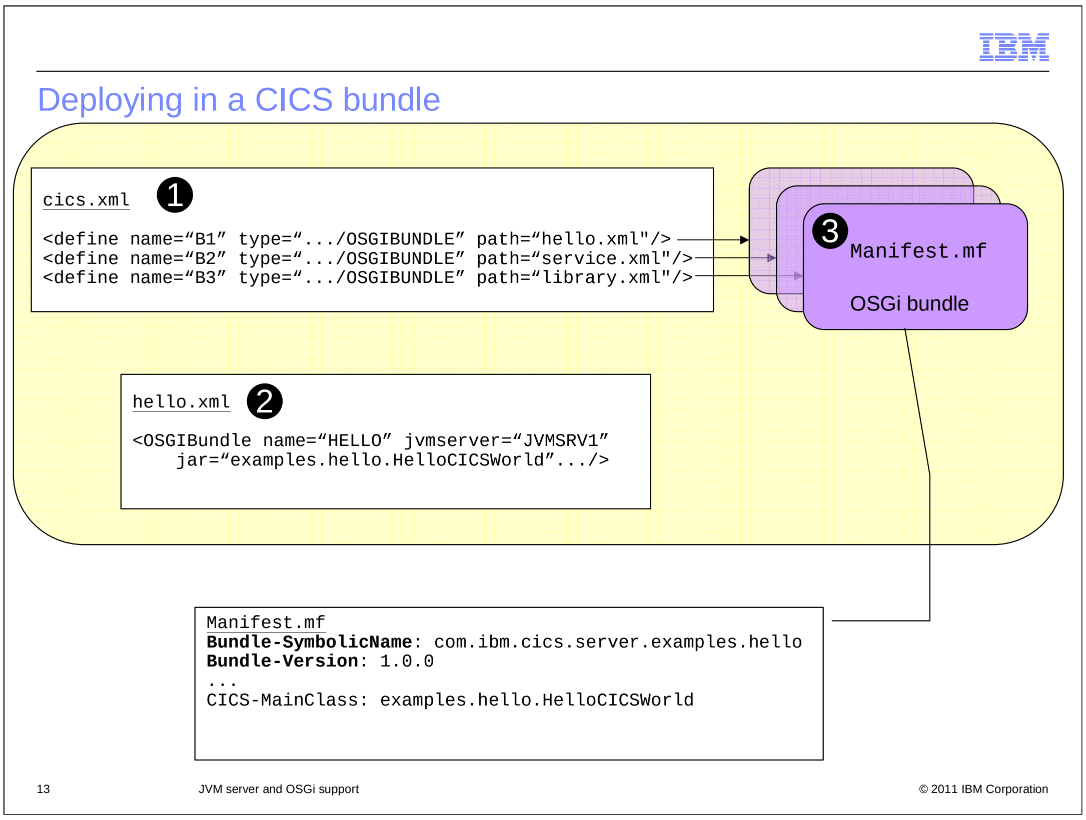
This screen capture shows the CICS Explorer SDK installed in Rational Application Developer with a plug-in project that uses one of the CICS Java samples.



Developing and deploying



In the CICS Explorer SDK you can create one or many OSGi bundles for an application. Use the CICS Bundle project wizard to create a CICS bundle to deploy the application to CICS. You can install, enable, disable, and discard the CICS BUNDLE resource to manage the lifecycle of the application. The OSGi framework handles the dependencies between the OSGi bundles in the framework.



Deploying in a CICS bundle

```
cics.xml  1

<define name="B1" type=".../OSGIBUNDLE" path="hello.xml"/>
<define name="B2" type=".../OSGIBUNDLE" path="service.xml"/>
<define name="B3" type=".../OSGIBUNDLE" path="library.xml"/>
```

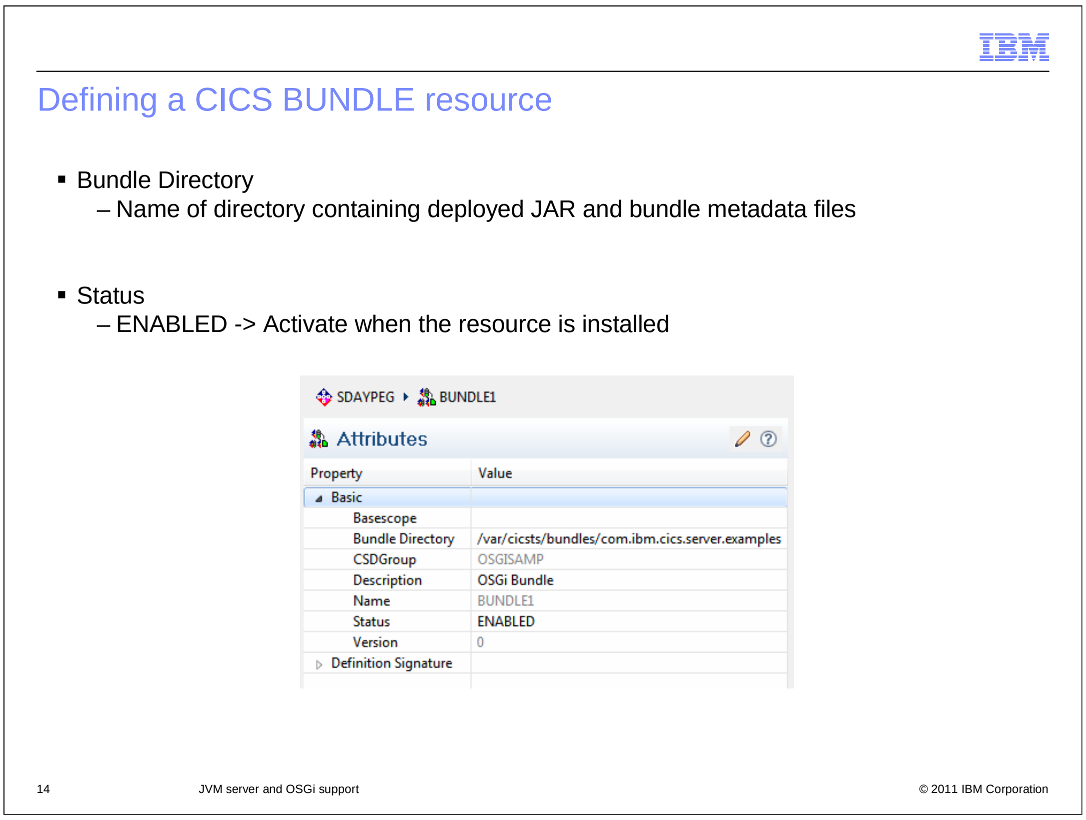
```
hello.xml  2

<OSGIBundle name="HELLO" jvmserver="JVMSRV1"
    jar="examples.hello.HelloCICSWorld".../>
```

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

13 JVM server and OSGi support © 2011 IBM Corporation

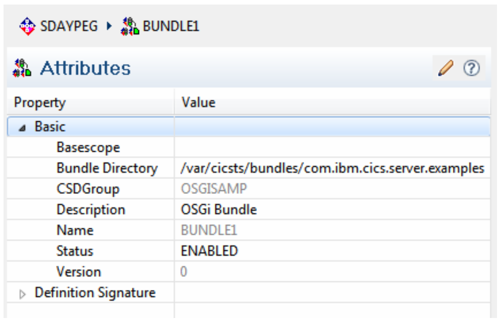
The OSGi implementation in CICS consists of several artifacts that define the metadata of the main resources. The CICS manifest file contains a define XML element of type OSGIBUNDLE for each OSGi bundle that is included in the CICS bundle. The OSGIBUNDLE resource specifies the XML file containing the OSGi bundle specification. The XML file contains the name of the OSGi bundle JAR file and the JVM server where the bundle is going to be deployed.

The OSGi bundle manifest file, included in the jar file, contains information about the bundle and the CICS directive CICS-MainClass, including the list of class to be registered as CICS services.



Defining a CICS BUNDLE resource

- Bundle Directory
  – Name of directory containing deployed JAR and bundle metadata files
- Status
  – ENABLED -> Activate when the resource is installed

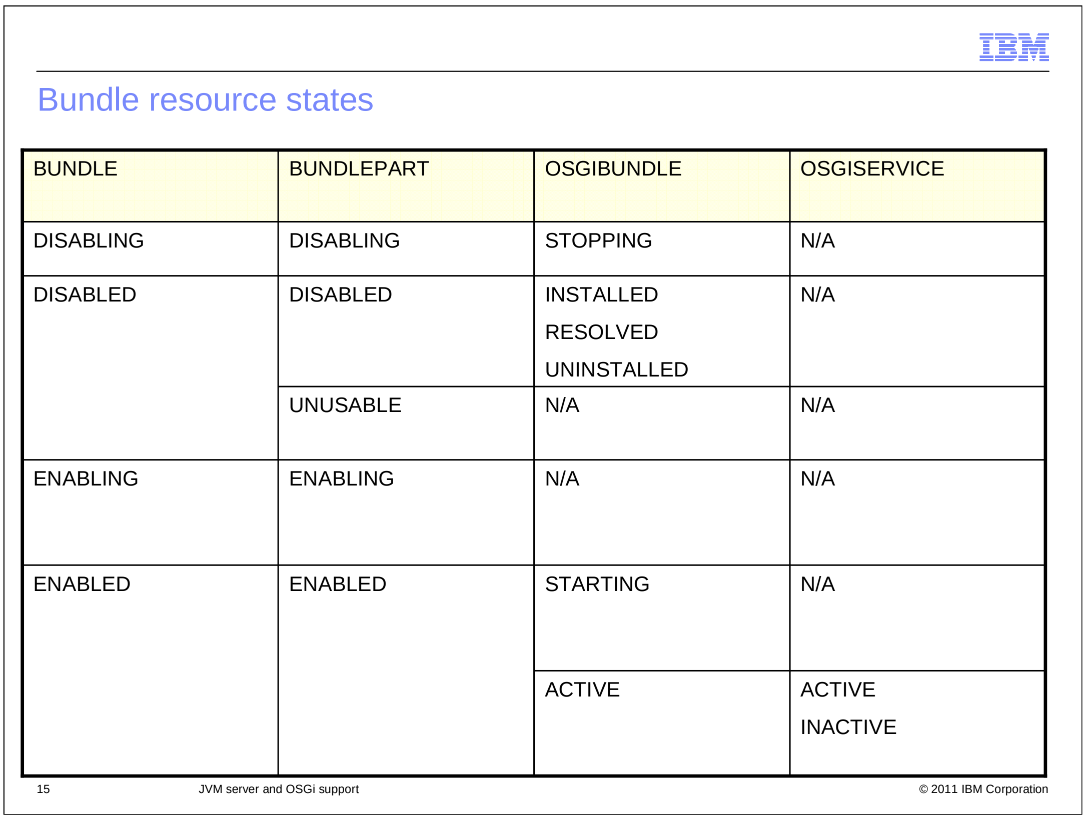

SDAYPEG ▸ BUNDLE1

Attributes

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	

14 JVM server and OSGi support © 2011 IBM Corporation

Create a BUNDLE resource definition specifying the directory of the deployed bundle. The Status attribute defines that CICS will install the resource in an enabled state and try to dynamically create all the required OSGi bundles and services in the designated JVM server.



Bundle resource states

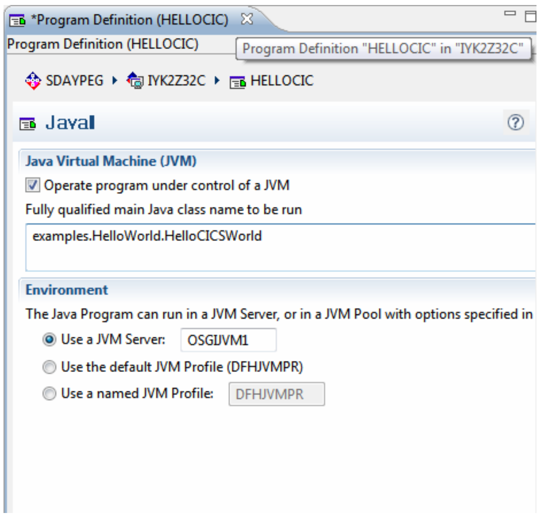
BUNDLE	BUNDLEPART	OSGIBUNDLE	OSGISERVICE
DISABLING	DISABLING	STOPPING	N/A
DISABLED	DISABLED	INSTALLED RESOLVED UNINSTALLED	N/A
	UNUSABLE	N/A	N/A
ENABLING	ENABLING	N/A	N/A
ENABLED	ENABLED	STARTING	N/A
		ACTIVE	ACTIVE INACTIVE

15 JVM server and OSGi support © 2011 IBM Corporation

This table summarizes the states for the BUNDLE resource, its bundle parts, and the associated OSGi bundles and services. When you install the BUNDLE resource, CICS dynamically creates resources for OSGi bundles and services from the bundle part information. If you disable a BUNDLE resource, the bundle parts move into the disabling state and CICS stops the OSGi bundles. When this process finishes, the bundle part is disabled and the OSGi bundles are in the resolved state. The resolved state means that all Java classes that the bundle needs are available and the bundle is ready to be started or has stopped. If you install the BUNDLE resource in a disabled state, CICS installs the OSGi bundles in the JVM server and the OSGi bundles are put in the installed state. If the bundle part is in the unusable state, a problem occurred and CICS was unable to create the OSGi bundle. If you install the BUNDLE resource in an enabled state, CICS installs and starts the OSGi bundles in the framework. If an OSGi bundle is in the starting state, the bundle is being started, the BundleActivator.start method is called but the method has not yet returned. When the bundle has a lazy activation policy, the bundle remains in the starting state until the bundle is activated. When the OSGi bundle is active, the bundle has been successfully activated and is running; its bundle activator start method has been called and returned. If the OSGi service is active, it can be called from CICS. If it is inactive, it means that an OSGi service of the same name is already running in the OSGi framework. It will become active when the other OSGi service is deregistered in the framework.

Defining a PROGRAM resource

- 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)

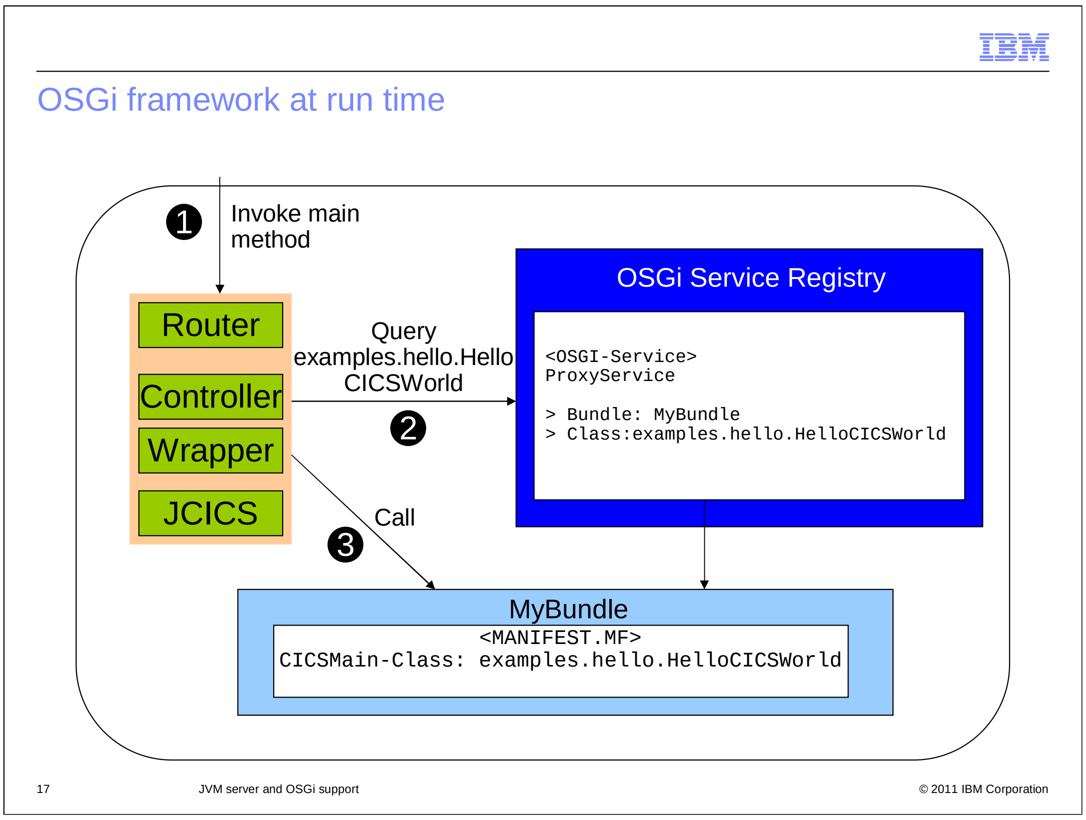

*Program Definition (HELLOCIC)
Program Definition (HELLOCIC) Program Definition "HELLOCIC" in "IYK2Z32C"
SDAYPEG ▸ IYK2Z32C ▸ HELLOCIC

Java

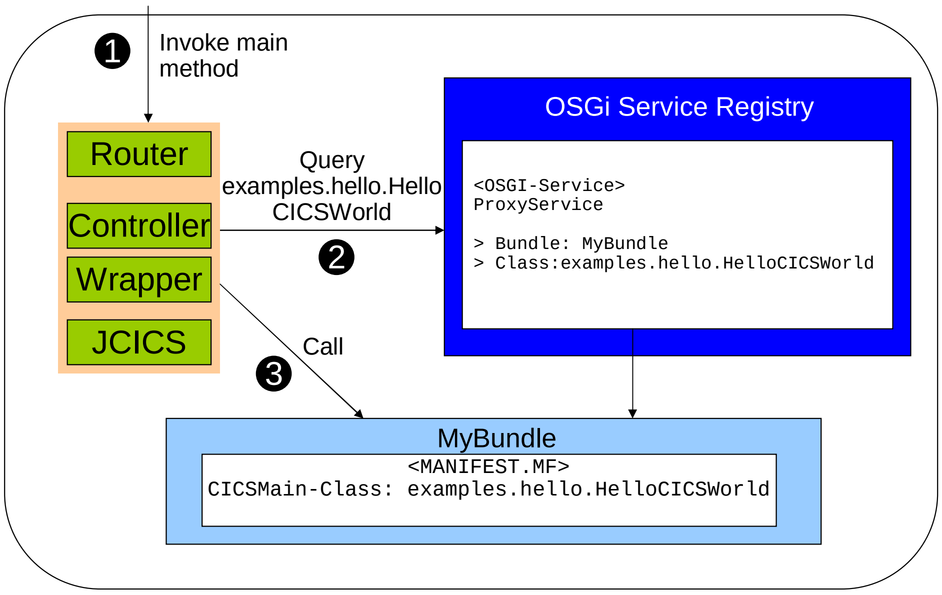
Java Virtual Machine (JVM)
[✓] Operate program under control of a JVM
Fully qualified main Java class name to be run
examples.HelloWorld.HelloCICSWorld

Environment
The Java Program can run in a JVM Server, or in a JVM Pool with options specified in
(●) Use a JVM Server: OSGIJVM1
( ) Use the default JVM Profile (DFHJVMPR)
( ) Use a named JVM Profile: DFHJVMPR

Define a PROGRAM resource for the Java application, specifying the JVM server in which the OSGi bundle is installed. Add the class that was specified in the CICS-MainClass directive. You must also specify that the program is threadsafe and runs in CICS key.



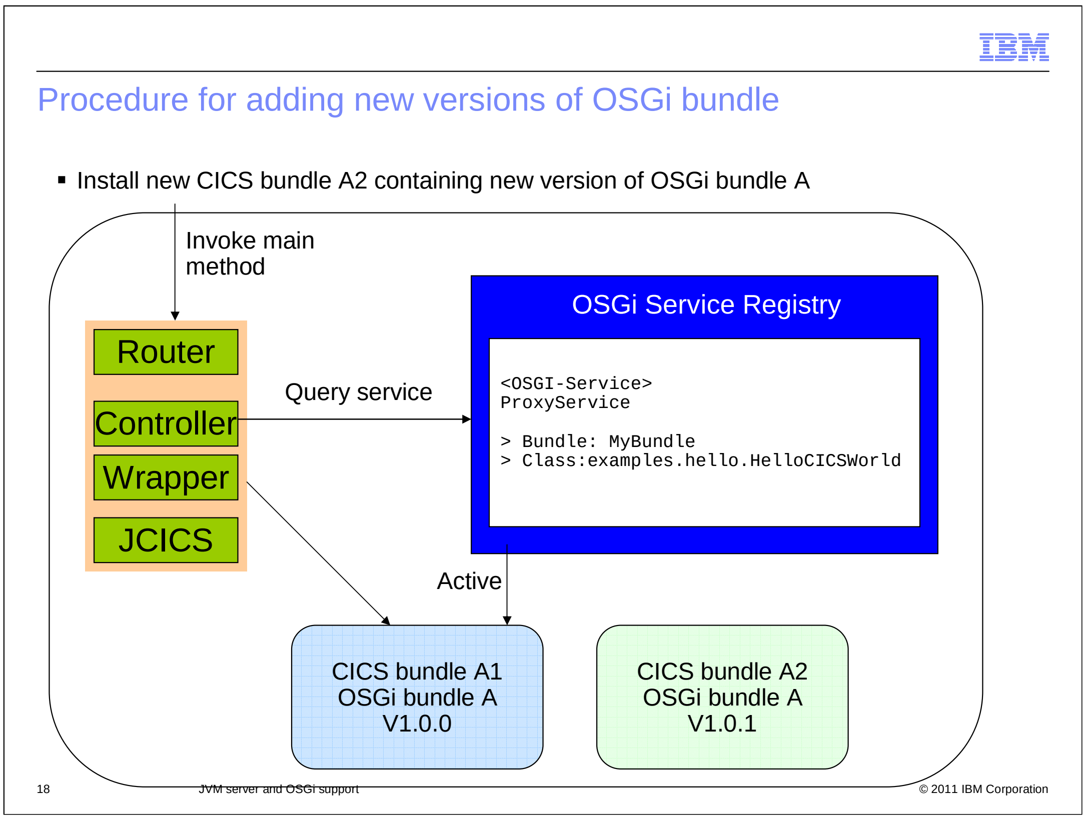
OSGi framework at run time



1
Invoke main method
Router
Controller
Wrapper
JCICS
Query examples.hello.HelloCICSWorld
2
OSGi Service Registry
<OSGI-Service> ProxyService
> Bundle: MyBundle
> Class:examples.hello.HelloCICSWorld
Call
3
MyBundle
<MANIFEST.MF>
CICSMain-Class: examples.hello.HelloCICSWorld

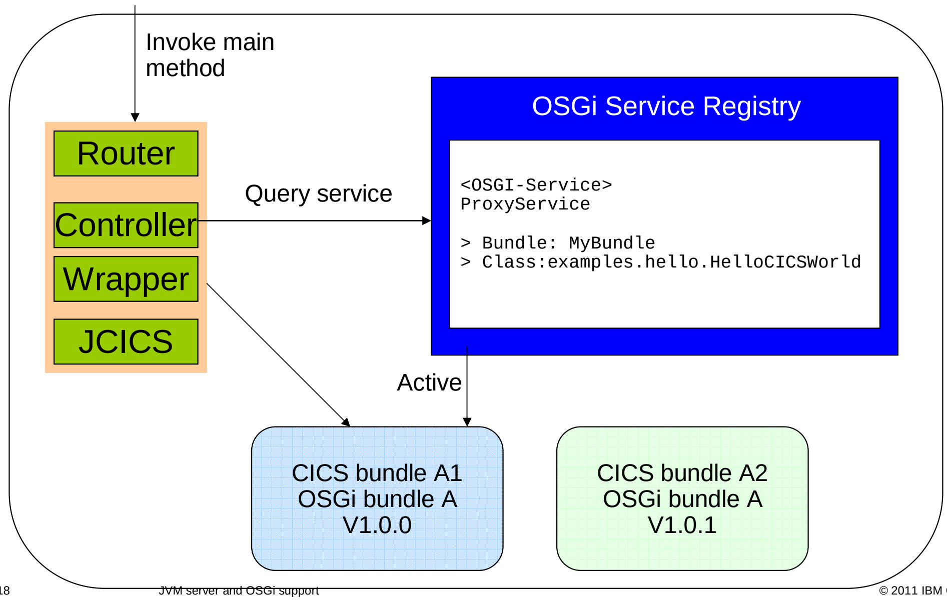
17 JVM server and OSGi support © 2011 IBM Corporation

This diagram shows the architecture of the OSGi framework in the JVM server and the flow when calling a Java program. The router is the component that sits between CICS and Java. It initializes the OSGi framework, stops it, installs system bundles, and routes any OSGi requests to the controller. The controller component handles OSGi bundles installed by CICS bundles. The wrapper component runs main methods of the Java class specified in the PROGRAM resource. When a Java program is linked to, the request goes through the router to the controller. The controller queries the service registry to find the OSGi service that is named as the class in the PROGRAM resource. The classes populating the service registry are those specified in the CICS-MainClass directive of the bundle manifest file and are added to the registry when the OSGi bundle is installed.

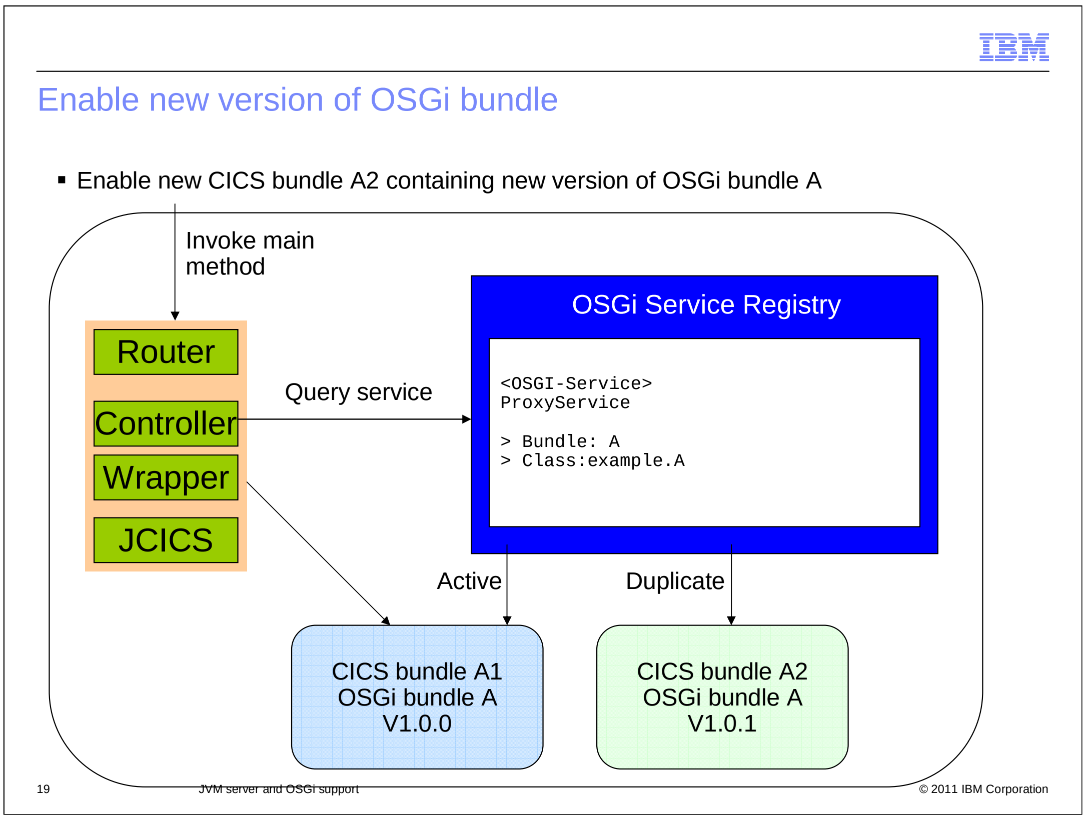


Procedure for adding new versions of OSGi bundle

- Install new CICS bundle A2 containing new version of OSGi bundle A

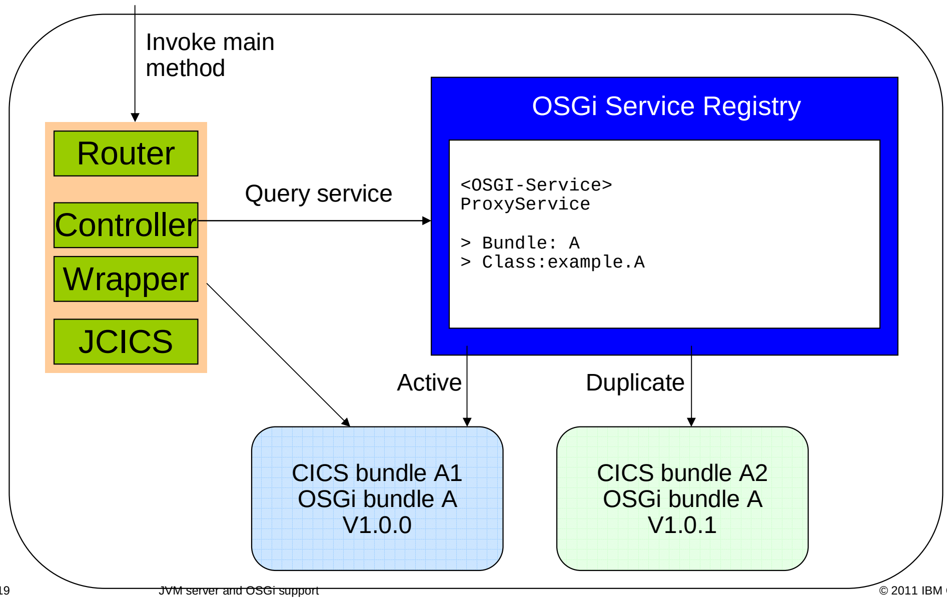

Invoke main method
Router
Controller
Wrapper
JCICS
Query service
OSGi Service Registry
<OSGI-Service> ProxyService
> Bundle: MyBundle
> Class:examples.hello.HelloCICSWorld
Active
CICS bundle A1 OSGi bundle A V1.0.0
CICS bundle A2 OSGi bundle A V1.0.1

One of the advantages of OSGi is that you can deploy new versions of an OSGi bundle into the framework while the existing version is still running. In this example, Version 1.0.0 of an application is deployed in the OSGi framework by a CICS bundle resource called A1. To deploy updates to an application, update the version information in the OSGi bundle and deploy in a new CICS bundle called A2.



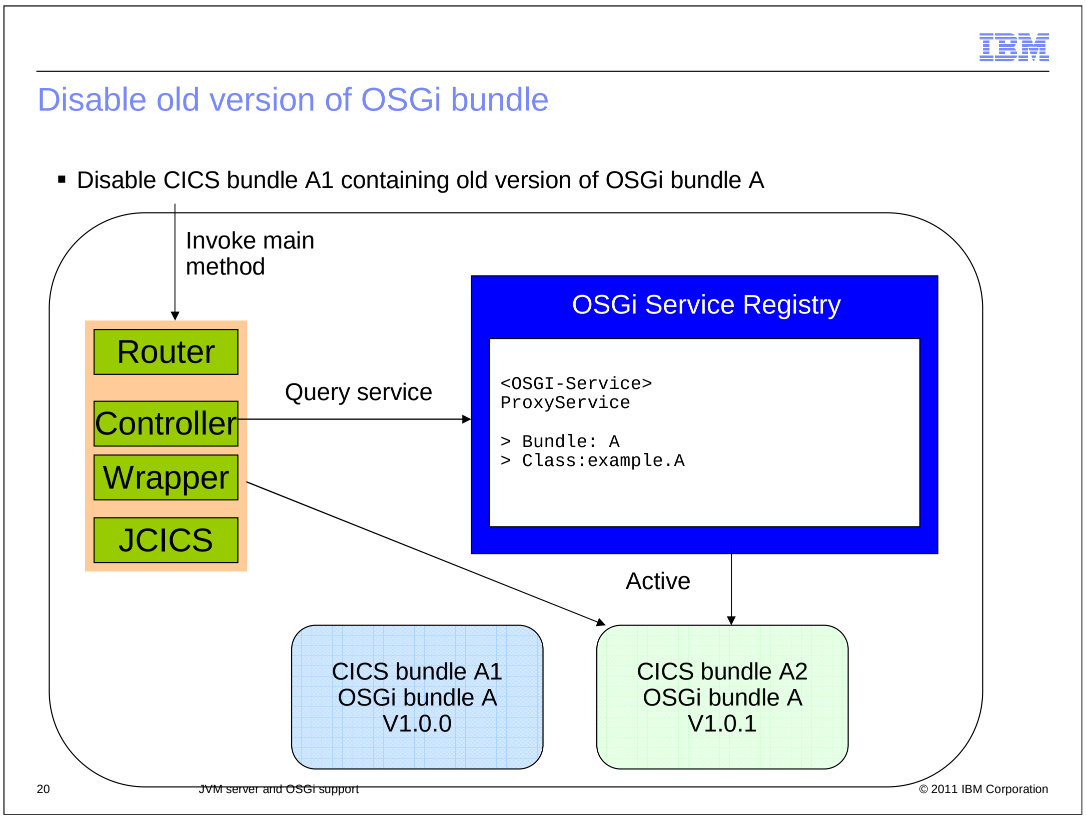
Enable new version of OSGi bundle

- Enable new CICS bundle A2 containing new version of OSGi bundle A



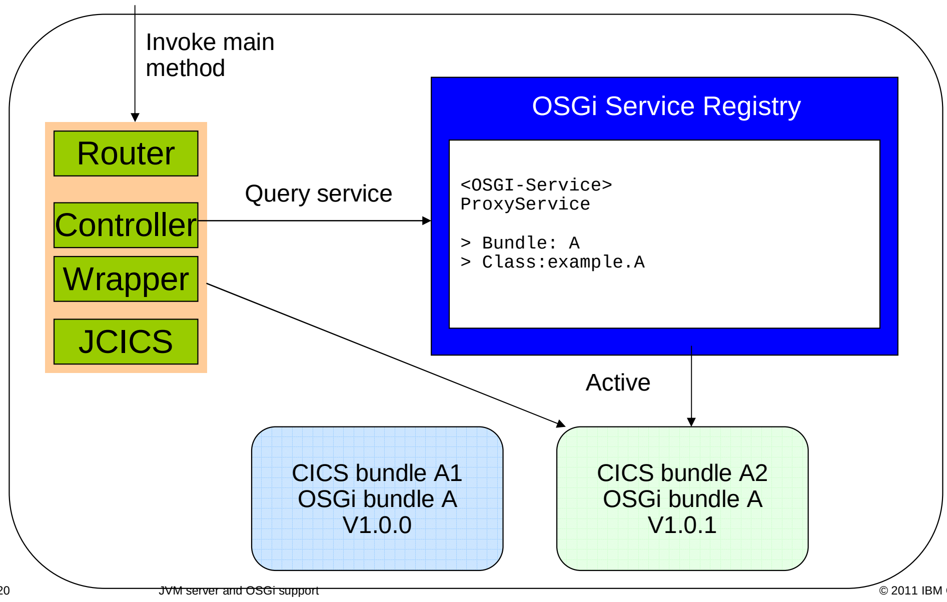
```
<OSGI-Service>
ProxyService

> Bundle: A
> Class:example.A
```

When you enable the BUNDLE resource called A2 the OSGi service bundle is installed in the resolved state. The OSGi service is inactive because it duplicates the A1 resource. CICS continues to use the original version of the OSGi bundle.

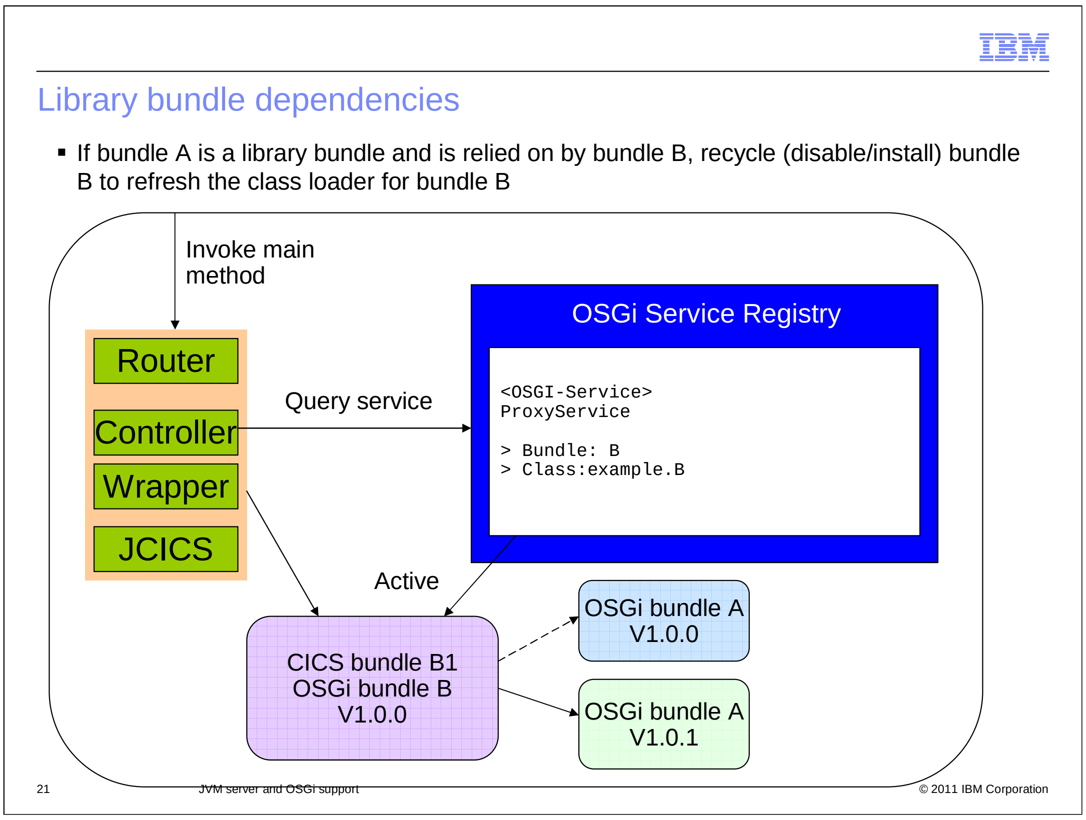


Disable old version of OSGi bundle

- Disable CICS bundle A1 containing old version of OSGi bundle A



Disable the original version of the application. The OSGi service for V1.0.1 of the OSGi bundle becomes active. The new version of the application is used for all subsequent transactions. You can discard bundle A1 to completely remove it from the OSGi framework.



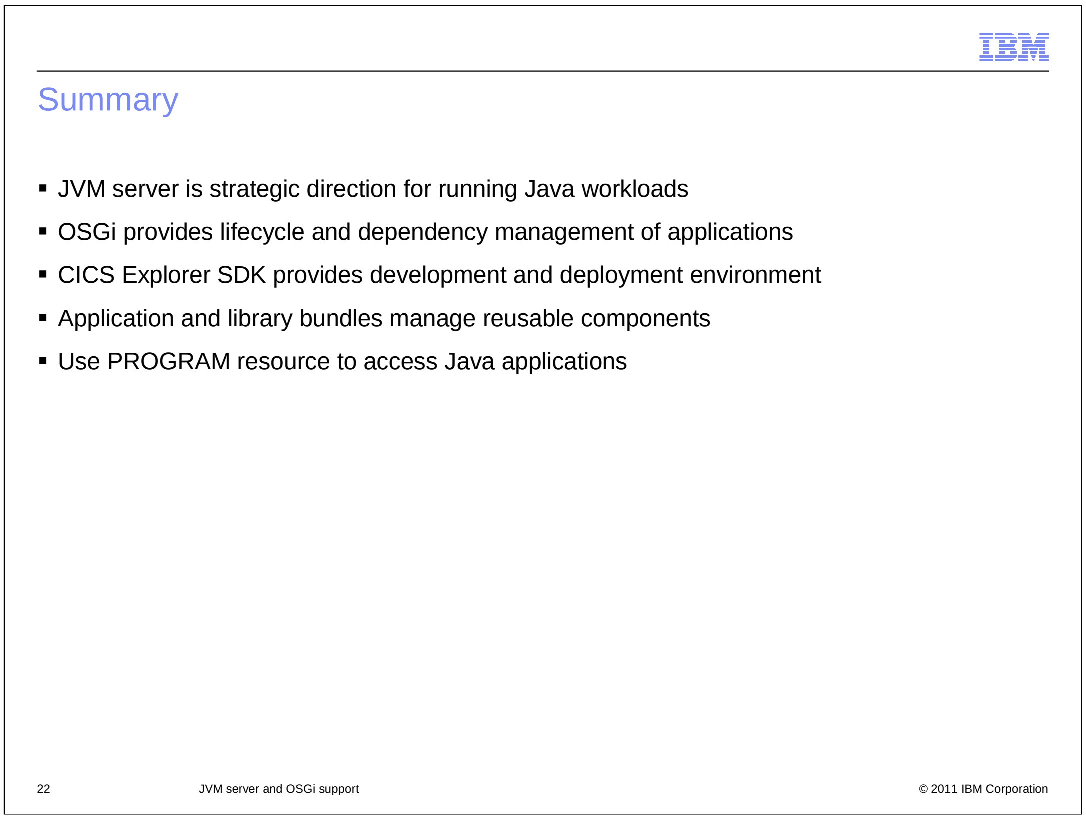
Library bundle dependencies

- If bundle A is a library bundle and is relied on by bundle B, recycle (disable/install) bundle B to refresh the class loader for bundle B

```
<OSGI-Service>
ProxyService

> Bundle: B
> Class:example.B
```

21 JVM server and OSGi support © 2011 IBM Corporation

If bundle A is a library bundle and is imported by OSGi bundle B, you must recycle the bundle to refresh the class loader of bundle B. To do this, disable and enable the BUNDLE resource B1.

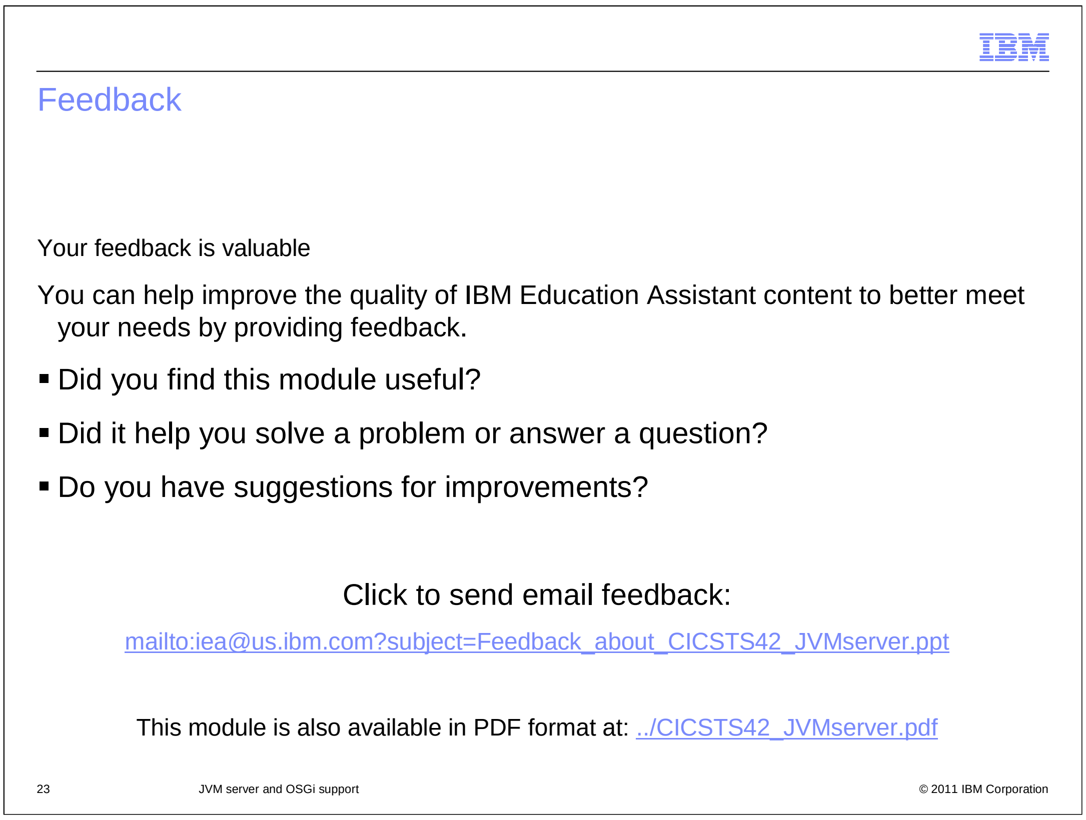


Summary

- JVM server is strategic direction for running Java workloads
- OSGi provides lifecycle and dependency management of applications
- CICS Explorer SDK provides development and deployment environment
- Application and library bundles manage reusable components
- Use PROGRAM resource to access Java applications

22 JVM server and OSGi support © 2011 IBM Corporation

In summary, the JVM server is the strategic runtime environment for Java workloads in CICS. The OSGi packaging and framework provides lifecycle and dependency management for Java applications. No class path is required and you do not have to restart the JVM server to introduce new applications or updates to those applications. The CICS Explorer SDK provides the support to develop and deploy Java applications into any release of CICS. Separating application and library bundles means you can more easily reuse application components. Java applications can be accessed from outside the JVM server by using a PROGRAM resource.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

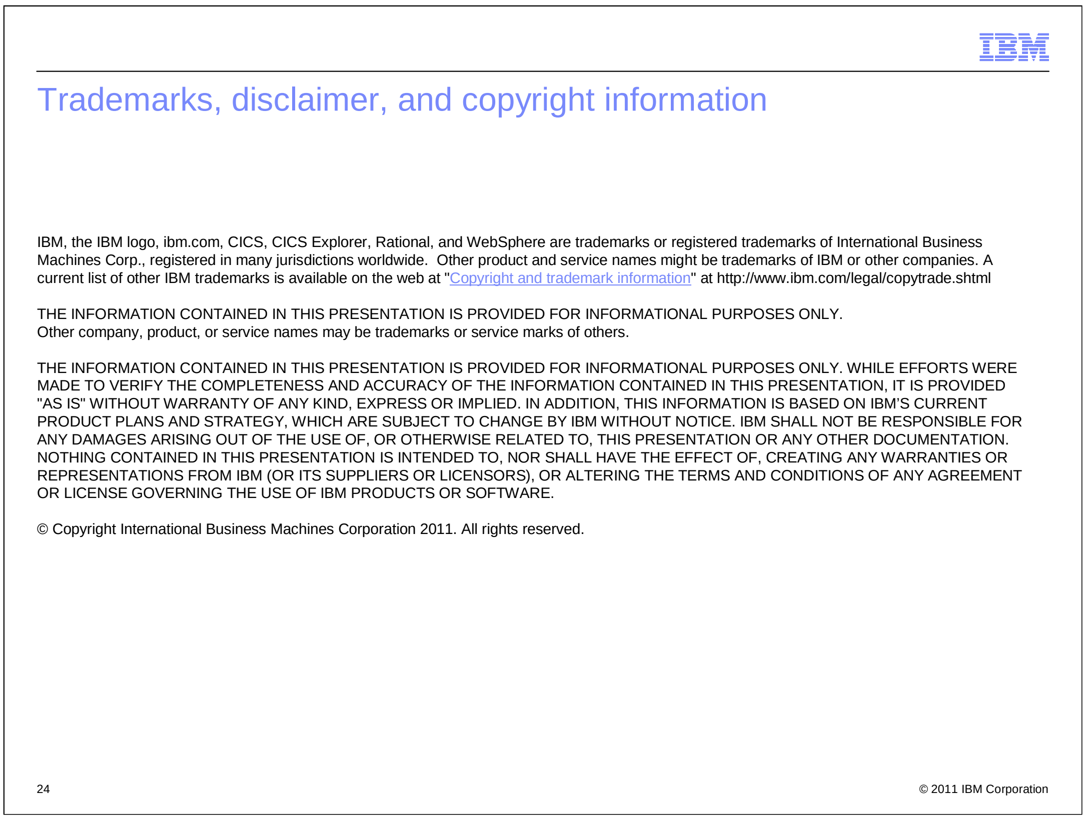
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_CICSTS42_JVMserver.ppt

This module is also available in PDF format at: ../CICSTS42_JVMserver.pdf

23 JVM server and OSGi support © 2011 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information

24 © 2011 IBM Corporation