

# CI/CS and OSGi

---

Steve Fowlkes – [fowlkes@us.ibm.com](mailto:fowlkes@us.ibm.com)  
Leigh Compton – [lcompton@us.ibm.com](mailto:lcompton@us.ibm.com)

# Trademarks and Copyright

© IBM Corporation 2013, 2015. 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](http://www.ibm.com/legal/copytrade.shtml)

## Abstract

- There's never been a better time to take advantage of Java technology in CICS.
- This topic details CICS support of OSGi and developing OSGi applications
- Liberty profile is covered in a different topic
- Debugging and tuning covered in a different topic

→ For clarification in this presentation, I capitalize BUNDLE when I am referring to a CICS BUNDLE, and not an OSGi bundle

# Agenda

What is OSGi?

CICS and OSGi

- Developing your program
- Defining your program as an OSGi program
- Put your OSGi bundle into a CICS BUNDLE
- Transfer your program to z/OS
- Define your program to CICS

Updating OSGi bundles

# What is OSGi?

*“The OSGi framework is a **module system and service platform for the Java programming language** ...*

*that implements a complete and dynamic component model, something that does not exist in standalone Java/VM environments.*

*Applications or components (coming in the form of bundles for deployment) can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot ”*

Wikipedia

## New Java 11 support

- IBM Semeru Runtime Certified Edition for z/OS, Version 11 fix pack 11.0.15.0 minimum
- Incorporates all changes from Java 9, 10 and 11
- Java language changes can affect applications
- JVM configuration changes can affect deployment
- Application testing is encouraged
- Java 8 remains supported (EOS 2025)
  - Upgrade to Java 11 (EOS 2024) when ready to do so, with a trajectory of other Java LTS versions (e.g. 17, 22)
- <https://www.ibm.com/docs/en/semeru-runtime-ce-z/11?topic=guide-migrating>

## OpenJDK enhancements for Java SE 9, 10, and 11

### JEPs from Java SE 9:

- 102: Process API Updates
- 110: HTTP 2 Client
- 143: Improve Contended Locking
- 193: Variable Handles
- 199: Smart Java Compilation, Phase Two
- 200: The Modular JDK
- 201: Modular Source Code
- 211: Elide Deprecation Warnings on Import Statements
- 213: Milling Project Coin
- 215: Tiered Attribution for javac
- 216: Process Import Statements Correctly
- 217: Annotations Pipeline 2.0
- 219: Datagram Transport Layer Security (DTLS)
- 220: Modular Run-Time Images
- 221: Simplified Doclet API
- 222: jshell: The Java Shell (Read-Eval-Print Loop)
- 223: New Version-String Scheme
- 224: HTML5 Javadoc
- 225: Javadoc Search
- 226: UTF-8 Property Files
- 227: Unicode 7.0
- 229: Create PKCS12 Keystores by Default
- 232: Improve Secure Application Performance
- 235: Test Class-File Attributes Generated by javac
- 236: Parser API for Nashorn
- 238: Multi-Release JAR Files
- 240: Remove the JVM TI hprof Agent
- 241: Remove the jhat Tool
- 244: TLS Application-Layer Protocol Negotiation Extension
- 247: Compile for Older Platform Versions
- 249: OCSP Stapling for TLS
- 251: Multi-Resolution Images
- 252: Use CLDR Locale Data by Default
- 253: Prepare JavaFX UI Controls and CSS APIs for Modularization
- 254: Compact Strings (not enabled by default)
- 255: Merge Selected Xerces 2.11.0 Updates into JAXP
- 256: BeanInfo Annotations
- 257: Update JavaFX/Media to Newer Version of GStreamer
- 258: HarfBuzz Font-Layout Engine
- 259: Stack-Walking API
- 260: Encapsulate Most Internal APIs
- 261: Module System
- 262: TIFF Image I/O
- 263: HIDPI Graphics on Windows and Linux®
- 264: Platform Logging API and Service
- 265: Marlin Graphics Renderer
- 266: More Concurrency Updates
- 267: Unicode 8.0
- 268: XML Catalogs
- 269: Convenience Factory Methods for Collections
- 272: Platform-Specific Desktop Features
- 273: DRBG-Based SecureRandom Implementations
- 274: Enhanced Method Handles
- 275: Modular Java Application Packaging
- 276: Dynamic Linking of Language-Defined Object Models
- 277: Enhanced Deprecation
- 280: Indify String Concatenation
- 282: jlink: The Java Linker
- 283: Enable GTK 3 on Linux
- 285: Spin-Wait Hints
- 287: SHA-3 Hash Algorithms
- 288: Disable SHA-1 Certificates
- 289: Deprecate the Applet API
- 290: Filter Incoming Serialization Data
- 292: Implement Selected ECMAScript 6 Features in Nashorn
- 298: Remove Demos and Samples
- 299: Reorganize Documentation

### JEPs from Java SE 10:

- 286: Local-Variable Type Inference
- 296: Consolidate the JDK Forest into a Single Repository
- 313: Remove the Native-Header Generation Tool (javah)
- 314: Additional Unicode Language-Tag Extensions
- 319: Root Certificates
- 322: Time-Based Release Versioning

### JEPs from Java SE 11:

- 181: Nest-Based Access Control
- 309: Dynamic Class-File Constants
- 318: Epsilon: A No-Op Garbage Collector
- 320: Remove the Java EE and CORBA Modules
- 321: HTTP Client (Standard)
- 323: Local-Variable Syntax for Lambda Parameters
- 324: Key Agreement with Curve25519 and Curve448
- 327: Unicode 10
- 329: ChaCha20 and Poly1305 Cryptographic Algorithms
- 330: Launch Single-File Source-Code Programs
- 331: Low-Overhead Heap Profiling
- 332: Transport Layer Security (TLS) 1.3
- 335: Deprecate the Nashorn JavaScript Engine
- 336: Deprecate the Pack200 Tools and API

# Notes

OSGi technology is the dynamic module system for Java™

“OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java platform.”

The OSGi specifications can be found at...

<http://www.osgi.org/Specifications/HomePage>

# OSGi Bundles



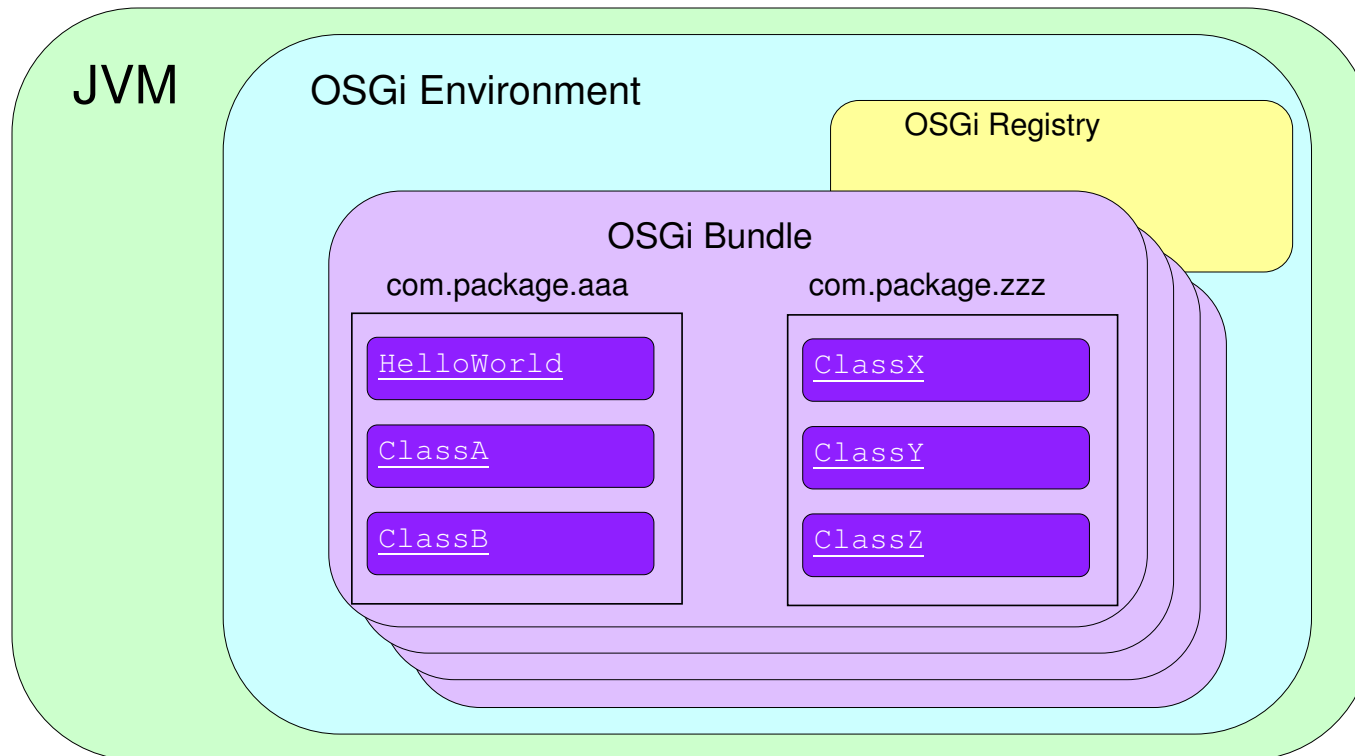
# OSGi Bundles

- OSGi Bundles can be installed, started, stopped, updated, and uninstalled in a running system
- Can run multiple versions of the same module at the same time
- Can contain an ‘activator’ (a class invoked when the OSGi bundle is started)
  - can not run any EXEC CICS commands in an ‘activator’
  - you can use the CICSExecutorService in a OSGi bundle activator to start some work that uses JCICS API
- No global class path, each bundle has its own class loader
- Can register themselves in the service registry
- Can declare and/or discover its dependencies

# Notes:

- JVM server has added the Equinox OSGi Framework into its runtime, allowing applications to be deployed as OSGi bundles. CICS Explorer SDK (based in Eclipse) can be used to create new, or turn existing applications into OSGi bundles. Right click your Java project and select “convert to OSGi Bundle project” (covered towards the end of this presentation)
- You can then create a ‘CICS bundle project’ through the Explorer SDK and place your OSGi bundles into a CICS bundle. Explorer can deploy your CICS bundles into CICS where the OSGi bundleparts are themselves deployed into the OSGi Framework of the JVM server
- With OSGi you don’t end up with a large sprawling classpath - each bundle is self-contained with its own classloader, and has explicit resolvable dependencies. JAR ‘Hell’ is not an issue, you can’t accidentally pick up anyone else’s classes
- OSGi also allows you to install, start, stop, and discard OSGi bundles on the fly while the JVM server is still running (via lifecycle operations on the parent CICS bundle). You can also have parallel versions of the same OSGi bundle – very handy for migrating to newer code
- One other key benefit is that all dependencies between OSGi bundles are explicitly declared. This allows the framework to resolve your bundle and all its dependencies before any code is actually run. Highly reduces CLASSDEFNOTFOUND exception!

# What is an OSGi Bundle?



➡ An OSGi bundle contains one or more Packages, each containing one or more Java classes

# What is an OSGi Bundle? (looking at a single bundle)

The OSGi manifest is created for you by the tooling as part of the OSGi bundle

An OSGi bundle is a Jar file with extra information in the manifest



➡ The manifest describes the content of the bundle

# What is an OSGi Bundle? (Bundle-SymbolicName)

By convention, the bundle's name is the name of the main package in the bundle.

But it is only a naming convention

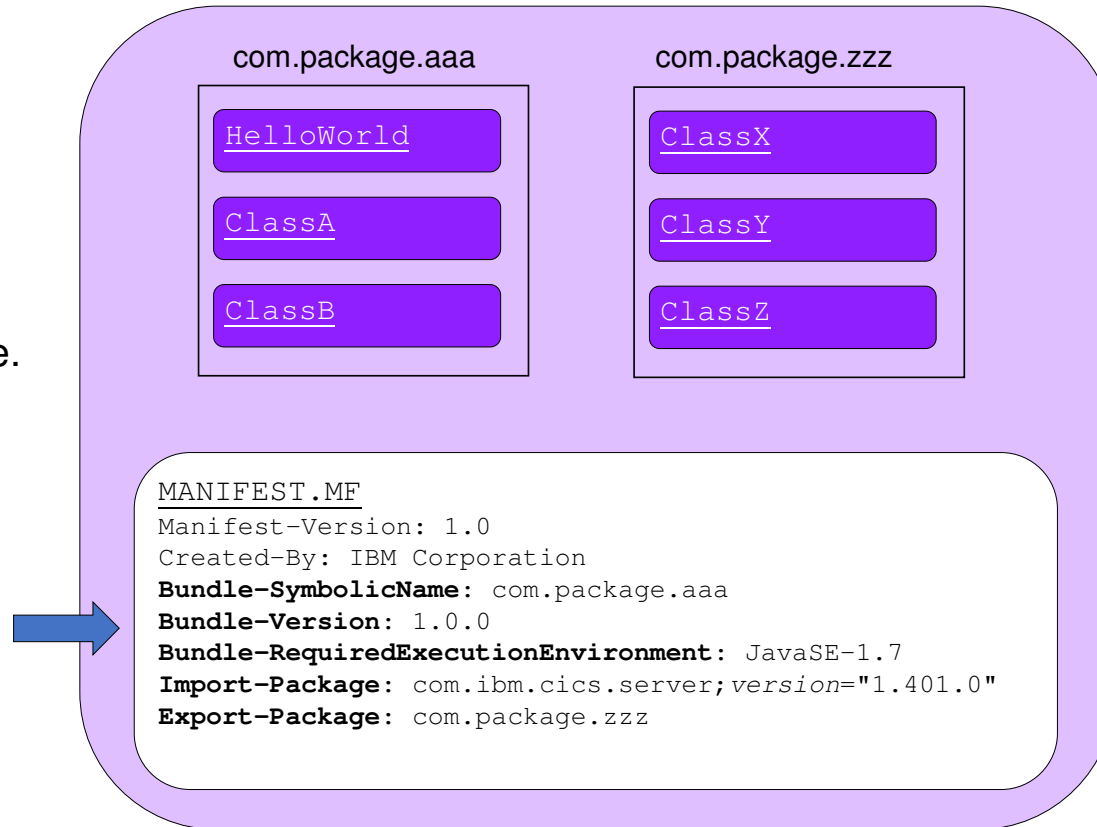


# What is an OSGi Bundle? (Bundle-Version)

The version of this bundle.

Use this for your  
versioning.

The same bundle with a  
different version number  
can exist in the same  
OSGi environment



# What is an OSGi Bundle? (RequiredExecutionEnvironment)

The minimum Java level  
that this bundle can run in.



# What is an OSGi Bundle? (Import-Package)

The packages in an OSGi bundle are isolated, except that any class in this bundle has access to any class in the indicated imported package(s).

You can include a range of versioned packages, by using the syntax:



**Import-Package: com.ibm.cics.server;version="[1.401.0,2.0.0]"**



©2019 IBM Corporation



# What is an OSGi Bundle? (Export-Package)

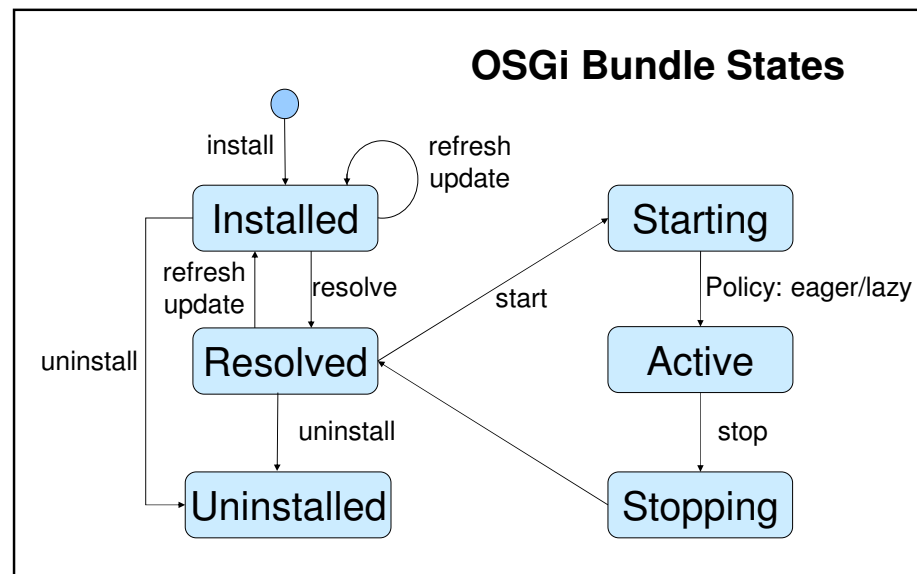
This bundle is making the indicated package(s) available to other OSGi bundles.

You normally don't expose very many classes, so the indicated bundle typically contains only the classes you want other bundles to have access to.



# OSGi Bundle Lifecycle

- OSGi Bundles can be installed, started, stopped, updated, and uninstalled in a running system – they can go active once dependencies are resolved
- You can run multiple versions of the same module at the same time
- They can contain an ‘activator’ (a class invoked when the OSGi bundle is started)
- Dispatches events when the state of the bundle changes
  - Service Events
  - Bundle Events
  - Framework Events
- Can register themselves in the service registry
- Can declare, or discover, its dependencies



# Notes:

- Each OSGi bundle has its own life cycle.
- When the OSGi bundle is installed and once all of the modules' dependencies are resolved, the bundle moves to the resolved state.
- When a resolved OSGi bundle is started, the bundle's activator (if any) is run. Once the activator completes, the module is in the 'active' state and the module packages that it has exported and the services it has registered are available.
- When an active module is stopped. The deactivator is run, and when the deactivator completes, the module is in the resolved status where it can either be refreshed, started, or uninstalled.

# OSGi Bundles in CICS

# OSGi Bundles in CICS

- Application Bundles (note: ‘application bundle’ in OSGi terminology\*\*)
  - Can be self-contained or have dependencies on other bundles
  - Managed by the framework (an app with an unresolved dependency cannot run)
  - To make programs available to CICS the OSGi bundle must declare a CICS main class as an OSGi service (or use [@CICSProgram](#))
  - A CICS PROGRAM definition points to the service
  - An OSGi bundle with common libraries doesn’t have to declare a CICS main class, thus making it available only to other OSGi bundles (that import it)
- Middleware bundles
  - Contain classes to implement system services (e.g. IBM MQ, DB2, or a bundle containing native code that should be loaded once per JVM)
  - Specified in the CICS JVMProfile (the startup file indicating JVM characteristics)
  - Exist for the life of the JVM (cannot update the copy in the running JVM)
  - Cannot contain a CICS main class
- System bundles
  - Manage interaction between CICS and OSGi framework (e.g. JCICS and OSGi bundles)

# Notes:

- Application bundles in this sense is meant to indicate a “Java Application as a set of modular components that can be reused and updated independently, without affecting the availability of applications and the JVM in which they are running.” - from the CICS TS V5.2 Knowledge Center.
- This is in contrast to CICS Application BUNDLES which can contain all of the CICS resources pertaining to a CICS Application.
- Note: this presentation capitalizes BUNDLE when it is talking about a CICS BUNDLE, and uses a lower case bundle when talking about OSGi bundles.

## An OSGi Bundle for CICS (CICS-MainClass)

A reference to this class is placed in the OSGi service registry

Later we will make a CICS PROGRAM entry that references the OSGi service registry entry



## Link to OSGi - @CICSProgram

- An annotation-based approach to developing CICS programs in OSGi
  - Programs can now be defined in-code using the @CICSProgram annotation
- During compilation of an OSGi bundle, the CICS annotation processor is run, detecting methods annotated @CICSProgram, and generates metadata and proxy classes
- When this OSGi bundle is installed into a JVM server
  - CICS scans this metadata and generates a program definition automatically, targeting the proxy class
- When the program is run, the proxy class locates the method annotated @CICSProgram and runs the logic
- This is a mirror of our link to Liberty support, providing a consistent approach
  - Don't worry, we'll continue to support CICS-MainClass

<https://www.ibm.com/docs/en/cics-ts/6.1?topic=loafcp-preparing-osgi-application-be-called-by-cics-program-using-cicsprogram>

```
public class App
{
    @CICSProgram("OSGIPROG")
    public void osgiProgram() throws CicsException
    {
        Task task = Task.getTask();
        Channel channel = task.getCurrentChannel();

        // Business logic here!
    }
}
```



# Development Environment: Create an OSGi Bundle

- In Eclipse (or RDz/RAD), File->New->Project, Plug-in Development->Plug-in Project

Convention: typically same name as the main Jar file

Choose this

Eclipse has used OSGi for several years. Eclipse calls an OSGi project a Plug-in Project

**New Plug-in Project**

Create a new plug-in project

Project name:

☒ Use default location

Location:

Choose file system:

**Project Settings**

☒ Create a Java project

Source folder:

Output folder:

**Target Platform**

This plug-in is targeted to run with:

☐ Eclipse version:

☒ an OSGi framework:

**Working sets**

☐ Add project to working sets

Working sets:

# Notes:

- The “standard” OSGi environment should be chosen so that Equinox specific characteristics are not injected into the bundle, and the bundle remains portable.

# Dev Environment: Manifest Editor in Eclipse

The screenshot displays the Eclipse Manifest Editor for the plug-in `com.ibm.ddw.customer.rest`. The interface is divided into two main sections: **Overview** and **Runtime**.

**Overview Tab:**

- General Information:** This section describes general information about the plug-in.
  - ID:** `com.ibm.ddw.customer.rest`
  - Version:** `1.0.0` (An annotation "Remove .qualifier" points to this field.)
  - Name:** `CustomerRest`
  - Vendor:** `IBM` (An annotation "Typically, no activator" points to this field.)
  - Activator:** (An annotation "Typically, no activator" points to this field.)
  - ☐ Activate this plug-in when one of its classes is loaded
  - ☐ This plug-in is a singleton
- Execution Environments:** Specify the minimum execution environment to run this plug-in.
  - Required Java Level:** `JavaSE-1.6` (An annotation "Required Java Level" points to this field.)

**Runtime Tab:**

- Imported Packages:** Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.
  - `com.ibm.cics.server [1.401.0,2.0.0]`
  - `com.ibm.ddw.customer.datalayouts`
  - `com.ibm.ddw.customer.maintenance`
- Exported Packages:** Enumerate all the packages that this plug-in exposes to clients. All other packages will be hidden from clients at all times.

The bottom of the editor shows a tabbed interface with the following tabs: **Overview**, **Dependencies**, **Runtime**, **Build**, and **MANIFEST.MF**.

# Notes:

- Unless the bundle contains code specific to a certain level of Java (e.g. Java 7 API), then it is not best practice to set a high level of Java, it reduces portability into lower JSE environments unnecessarily.
- "Imported Packages" – when you select a certain level from the menu, you are restricting the imported package to a single version (e.g. `com.ibm.cics.server(1.500.0)` ). To allow better future-proofing and extendibility of the Java application, a version range should be chosen to include API up to the next semantic breakage. i.e. `Import-Package: com.ibm.cics.server;version="[1.401.0,2.0.0)"`

Additionally, a version or version range should be specified in all but extreme cases.

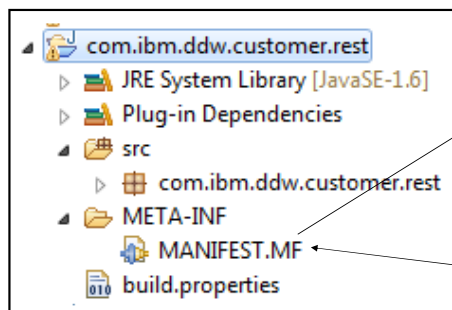
- "Export-Package" – Likewise, it is always best-practice to include the version that is exported.

# Development Environment: OSGi Bundle

- Once we have an OSGi bundle project defined in Eclipse...
- We will now:
  - Encapsulate this OSGi bundle into a CICS BUNDLE
  - Export the CICS BUNDLE to z/OS UNIX System Services
  - Install the CICS BUNDLE into CICS using a BUNDLE definition
  - On the following slides...

---

## Eclipse Bundle Project

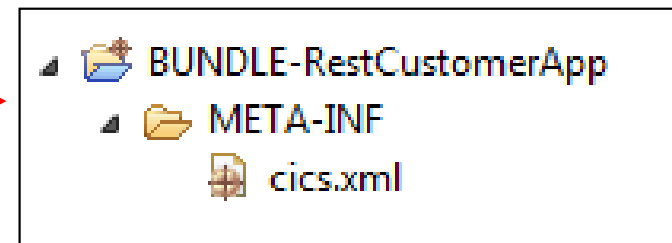
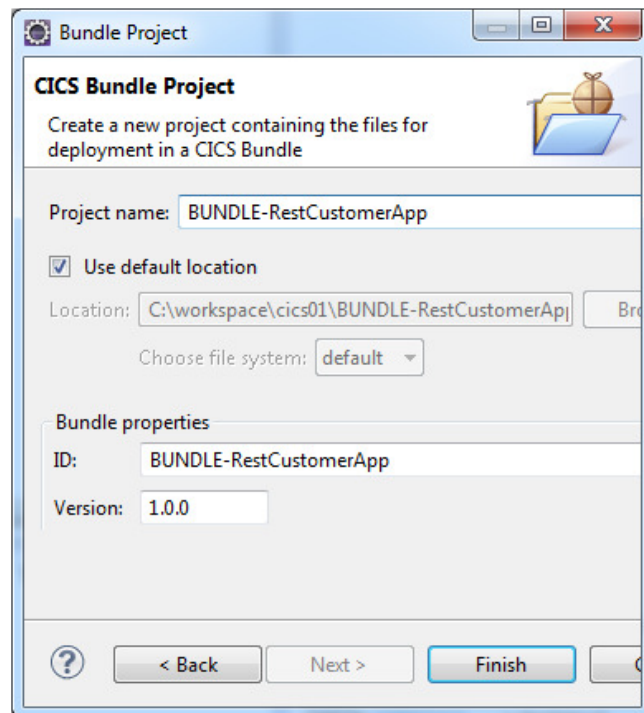


## Manifest:

```
1 Manifest-Version: 1.0
2 Bundle-ManifestVersion: 2
3 Bundle-Name: CustomerRest
4 Bundle-SymbolicName: com.ibm.ddw.customer.rest
5 Bundle-Version: 1.0.0
6 Bundle-Vendor: IBM
7 CICS-MainClass: com.ibm.ddw.customer.rest.CustResJ
8 Bundle-RequiredExecutionEnvironment: JavaSE-1.6
9 Import-Package: com.ibm.cics.server;version="[1.401.0,2.0.0)",
10 com.ibm.ddw.customer.datalayouts,
11 com.ibm.ddw.customer.maintenance
```

# Development Environment: CICS BUNDLE

- File->New->Other, CICS Resources->CICS Bundle Project



On your workstation

# Notes:

- This is an example of a CICS BUNDLE definition

# What is a CICS Bundle?



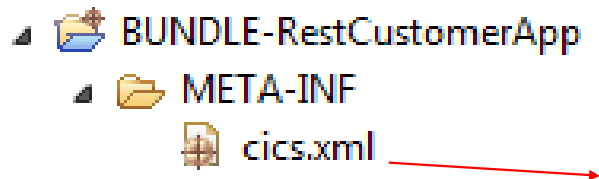
- To define a CICS BUNDLE on your workstation in Eclipse: New->Project, then CICS Resources->CICS Bundle Project (like on the previous slide)
- This says the same as the last slide, it says it in a different way



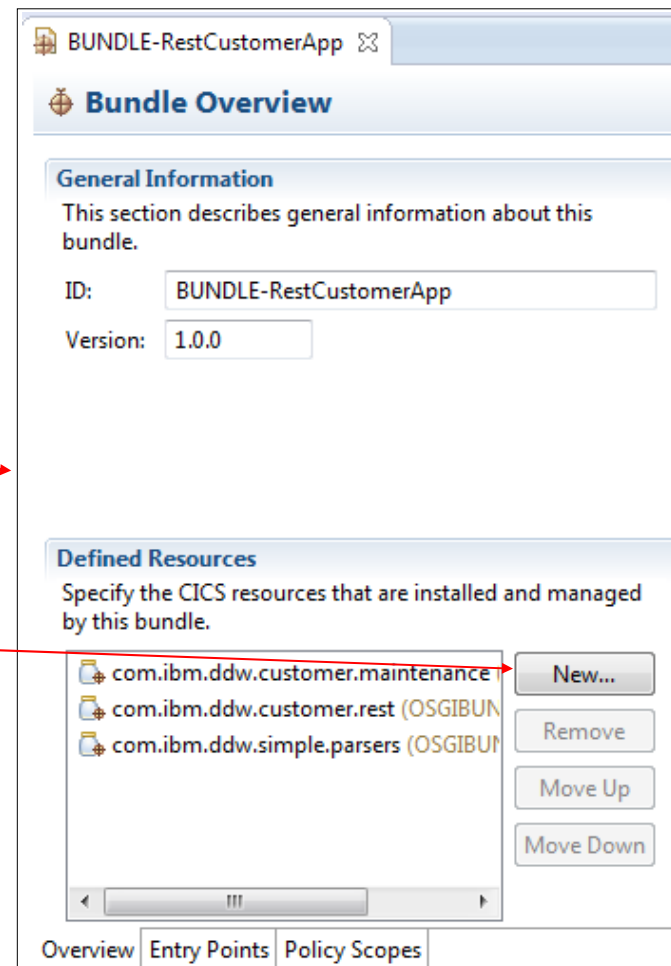
# Notes:

- This is another example of a CICS BUNDLE – it has a META-INF directory containing a manifest (a cics.xml file)

# Development Environment: CICS BUNDLE



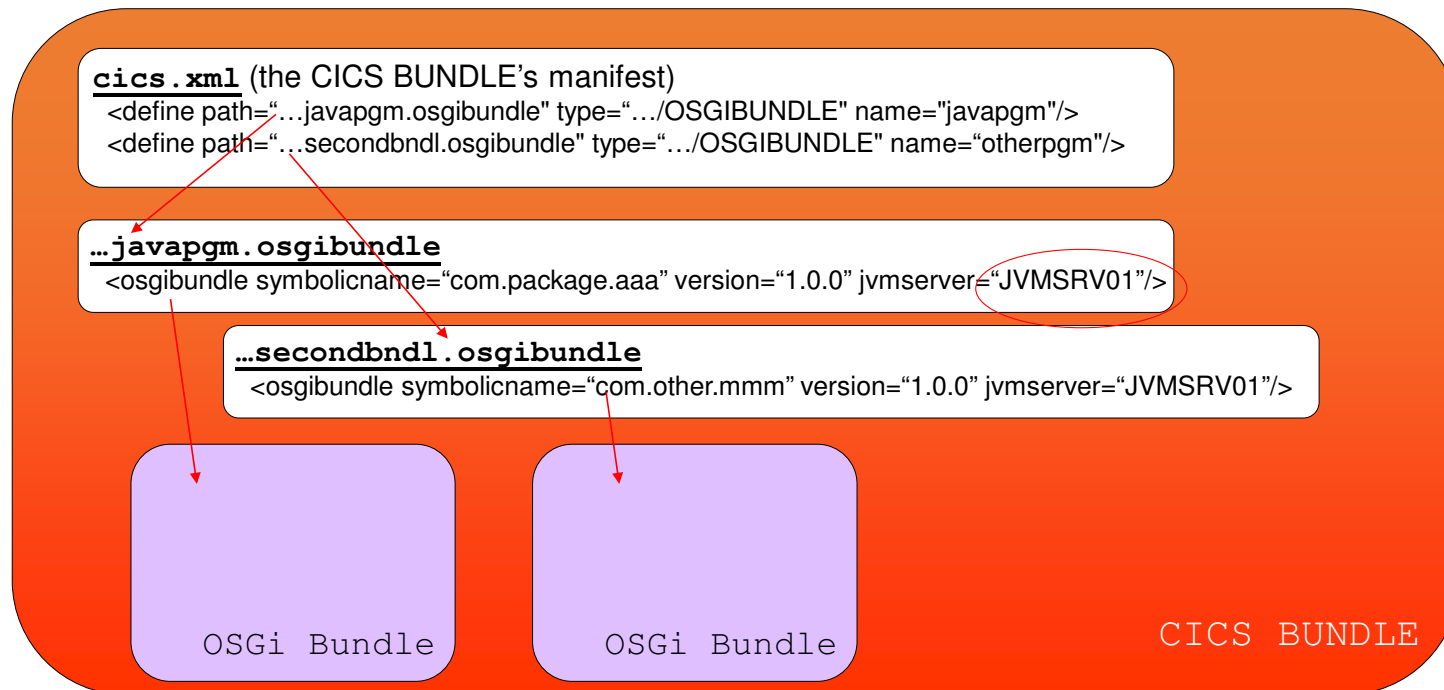
- Double-click the cics.xml file to start a manifest editor
- Click new to add your OSGi project(s)
- When you add an OSGi project, you indicate the name of the JVMSERVER it will run in



# Notes:

- This is a CICS manifest opened in the graphical CICS BUNDLE manifest editor.
- This CICS BUNDLE has some references to OSGi bundles (lower right part of the page). You can add OSGi bundles to a CICS BUNDLE by clicking the ADD button, indicating that you want to add OSGi Bundle Project References.
- After indicating that you want to add OSGi Bundle Project References, a list of OSGi projects/bundles pops up and you can add OSGi bundles to your CICS BUNDLE (one OSGi bundle at a time)
- You must also indicate the name of the JVMSERVER which defines the JVM server in which this OSGi bundle is to run.

# CICS BUNDLE – with two OSGi bundles in it



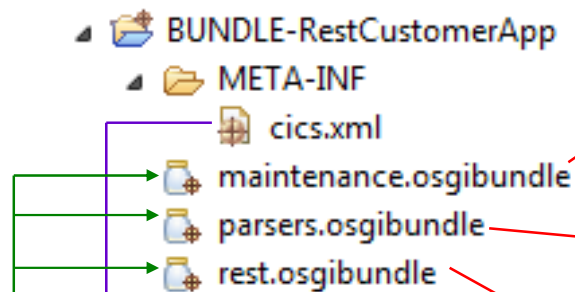
- To insert an OSGi bundle into a CICS BUNDLE
- You can use the technique described on the previous page, or right-click on your CICS BUNDLE, and from the context menu, select New->Include OSGi Project in Bundle

# Notes:

- This slide shows a CICS BUNDLE containing two OSGi bundles. A CICS BUNDLE can contain 1 to n OSGi bundles.
- When adding an OSGi bundle to the CICS BUNDLE, the CICS manifest points to a .osgibundle file containing a reference to the OSGi bundle code. The .osgibundle file also contains the name of the JVMSERVER that defines the JVM server into which this OSGi bundle will be loaded into.

# CICS BUNDLE contains OSGi bundle(s)

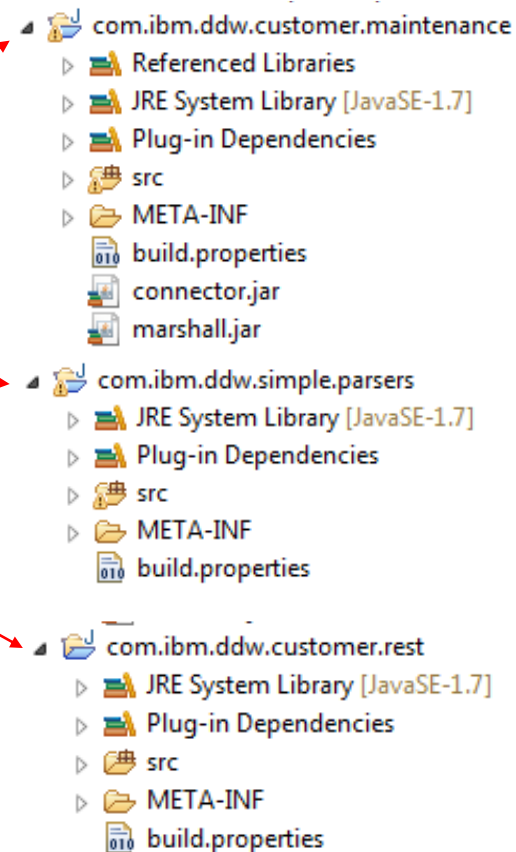
- CICS BUNDLE for Java programs



cics.xml:

```
<define path="maintenance.osgibundle"
      type="../../../OSGIBUNDLE" name="maintenance"/>
<define path="rest.osgibundle"
      type="../../../OSGIBUNDLE" name="rest"/>
<define path="parsers.osgibundle"
      type="../../../OSGIBUNDLE" name="parsers"/>
```

## ■ OSGi bundle projects



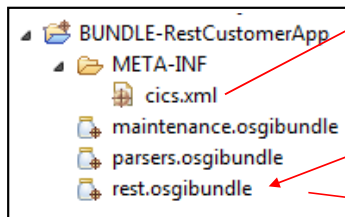
# Notes:

- This slide illustrates the relationship a bit differently.
- A CICS BUNDLE contains
  - A manifest (with references to the .osgibundle files)
  - .osgibundle files (with references to the OSGi bundle projects)
- On your workstation, that's all you see. When you export the CICS BUNDLE to zFS, the OSGi bundles will be Jar'd up and included in the CICS BUNDLE.

# CICS BUNDLE (looking at it a bit differently)

- A CICS BUNDLE project - referencing the OSGi bundle project

## CICS BUNDLE Project



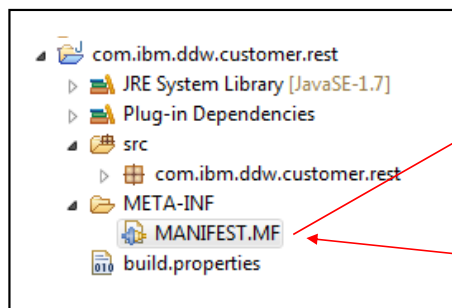
### cics.xml

```
<define path="maintenance.osgibundle" type="...OSGIBUNDLE" name="maintenance"/>  
<define path="rest.osgibundle" type="...OSGIBUNDLE" name="rest"/>  
<define path="parsers.osgibundle" type="...OSGIBUNDLE" name="parsers"/>
```

### rest.osgibundle

```
<osgibundle symbolicname="com.ibm.ddw.customer.rest" version="1.0.0" jvmserver="JVMSRV01"/>
```

## One of the Eclipse Projects in the CICS BUNDLE



### Manifest:

```
1 Manifest-Version: 1.0  
2 Bundle-ManifestVersion: 2  
3 Bundle-Name: CustomerRest  
4 Bundle-SymbolicName: com.ibm.ddw.customer.rest  
5 Bundle-Version: 1.0.0  
6 Bundle-Vendor: IBM  
7 CICS-MainClass: com.ibm.ddw.customer.rest.CustResJ  
8 Bundle-RequiredExecutionEnvironment: JavaSE-1.6  
9 Import-Package: com.ibm.cics.server;version="[1.401.0,2.0.0)",  
10 com.ibm.ddw.customer.datalayouts,  
11 com.ibm.ddw.customer.maintenance
```



# Notes:

- This slide illustrates the relationship a bit differently.
- A CICS BUNDLE contains
  - A manifest (with references to the .osgibundle files)
  - .osgibundle files (with references to the OSGi bundle projects)
- Remember that on your workstation, that's all you see. When you export the CICS BUNDLE to zFS, the OSGi bundles will be Jar'd up and included in the CICS BUNDLE.

## **CICS BUNDLES** (for OSGi programs on your workstation)

- CICS BUNDLE manifest editor
  - If there is an error in one of the OSGi bundles, the CICS BUNDLE will show an error
- You can't export BUNDLES with errors to z/OS
  - Edit the OSGi project that this BUNDLE refers to, and fix the error
  - If you change one of the OSGi bundles, just export the CICS BUNDLE again

# Notes:

- You can't export a CICS BUNDLE with errors to zFS.
- The CICS BUNDLE will show an error if any of the OSGI bundles that are included in the CICS BUNDLE have an error.

## Export CICS BUNDLE and Install it into CICS

- Export the CICS BUNDLE to z/OS UNIX System Services
  - Right-click on the CICS BUNDLE and from the context menu select 'Export Bundle Project to z/OS UNIX File System'
- Create a BUNDLE definition in CICS that refers to the exported BUNDLE (on the z/OS UNIX File System)
- (Create a JVMSERVER resource – if you don't have one already) (discussed later)
- Install the BUNDLE into CICS
  - The associated OSGi bundle(s) are then installed into the indicated JVM server

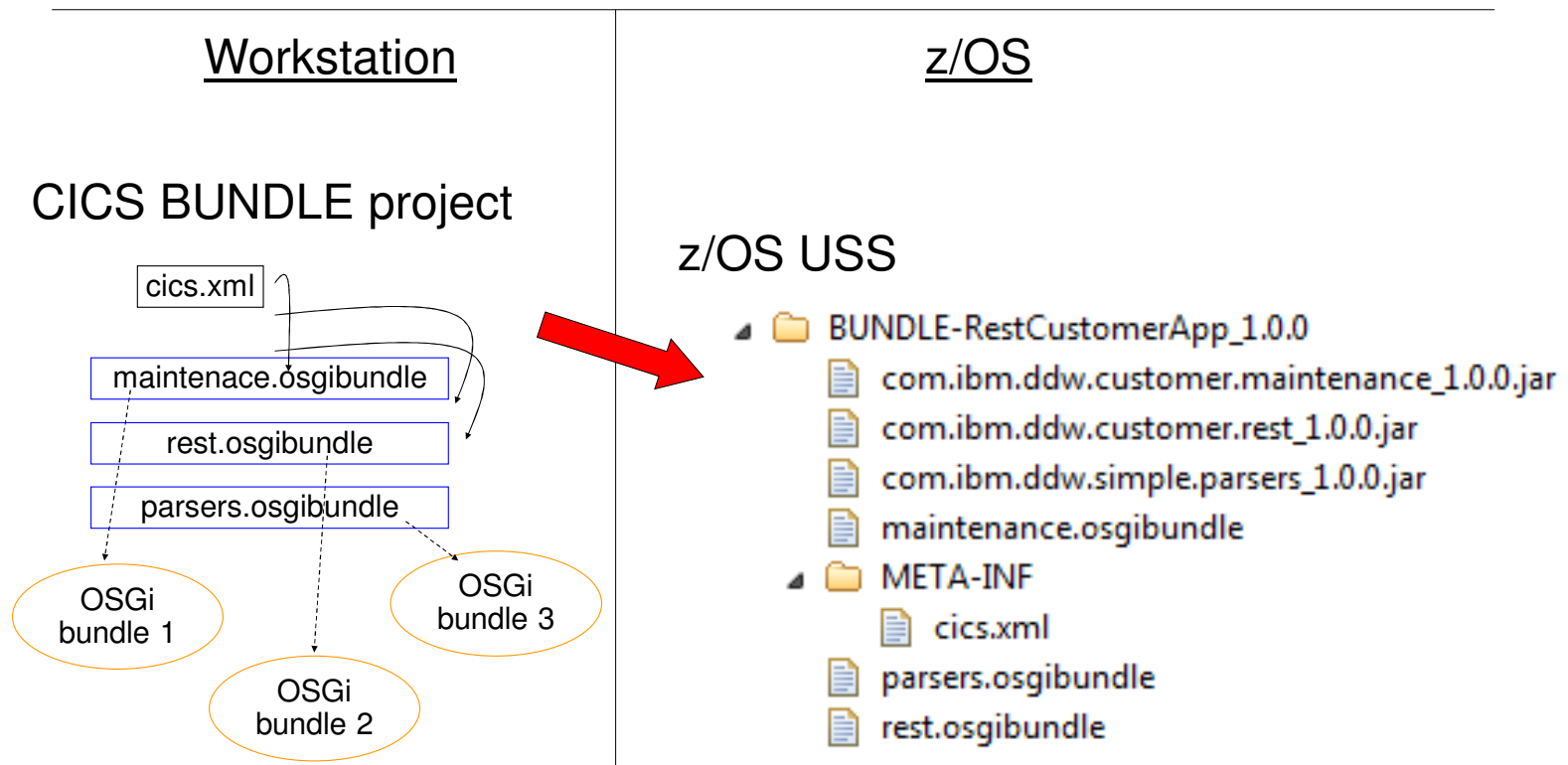
Steps discussed in detail  
on upcoming slides

# Notes:

- This slide indicates the next steps...
- After you export the CICS BUNDLE to zFS, you will have to create a CICS BUNDLE resource definition. The CICS BUNDLE definition references the CICS BUNDLE that you placed on zFS.
- Later we will discuss the JVMSERVER resource definition in CICS that defines a JVM server.
- The next slide will graphically illustrate the CICS BUNDLE on z/FS.

# Export CICS BUNDLE to z/OS UNIX File System

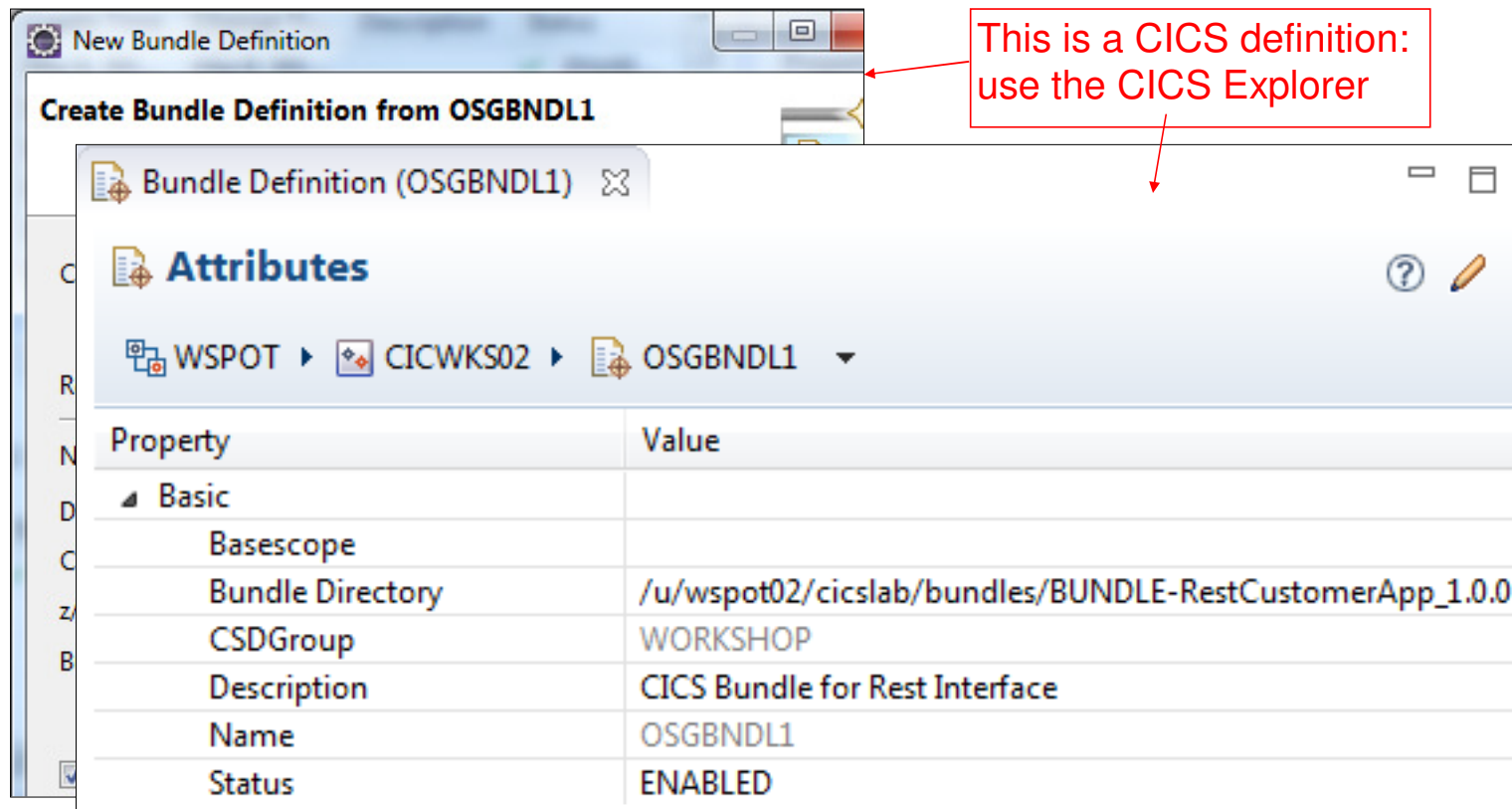
- First, export the CICS BUNDLE to z/OS UNIX System Services



# Notes:

- On your workstation,
  - You have a CICS BUNDLE project.
  - The CICS BUNDLE project contains a CICS manifest (named cics.xml)
  - The cics.xml points to .osgibundle files
  - The .osgibundle files refer to the OSGi bundle projects
- When the CICS BUNDLE is moved to zFS, the current version of the OSGi projects are included. This means that if you make a change to one of the OSGi projects, you don't have to change the CICS BUNDLE. You just have to export the CICS BUNDLE to zFS again.
- Note that if any of the OSGi project/bundles have errors, the CICS BUNDLE will show an error.

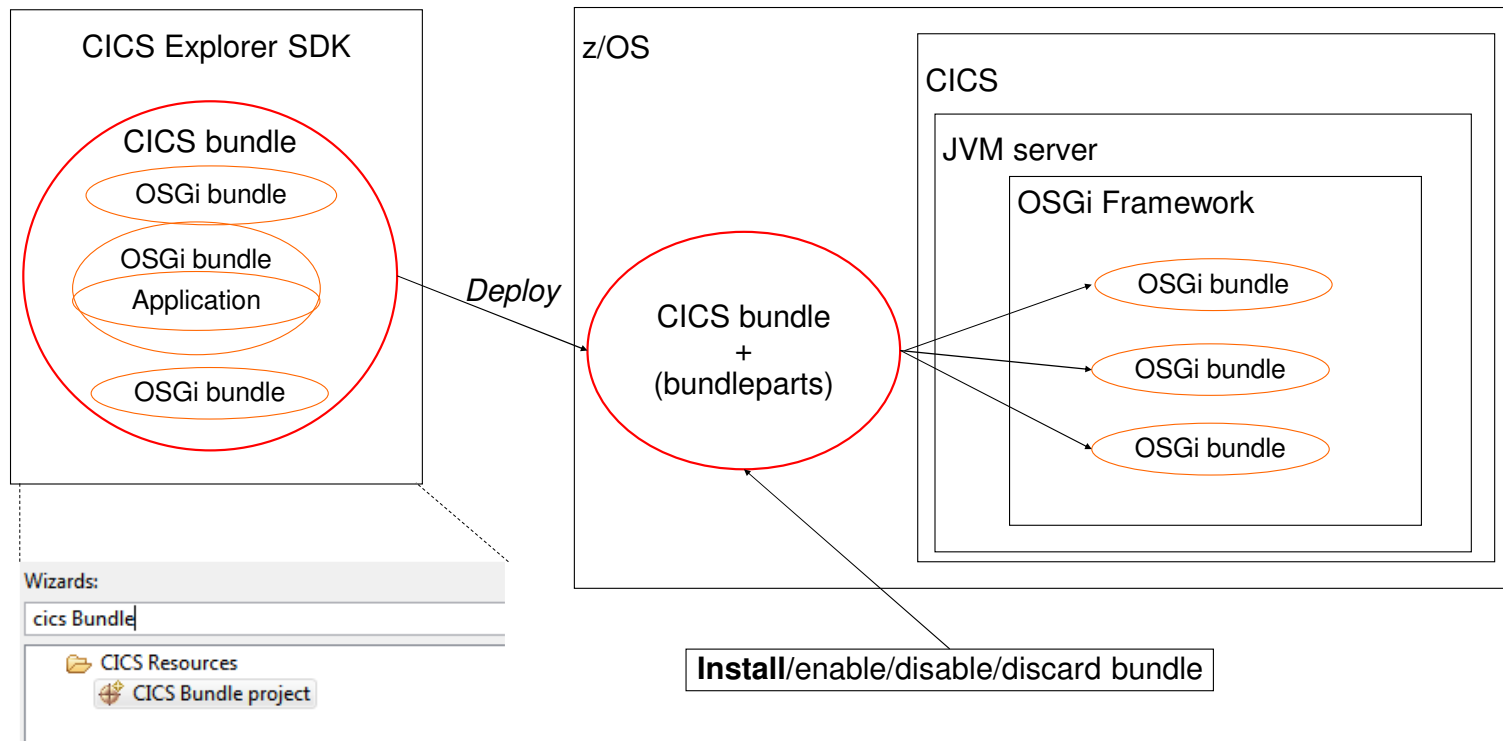
## You need a CICS BUNDLE Definition



When the BUNDLE is installed, the Java bundles go into the JVM

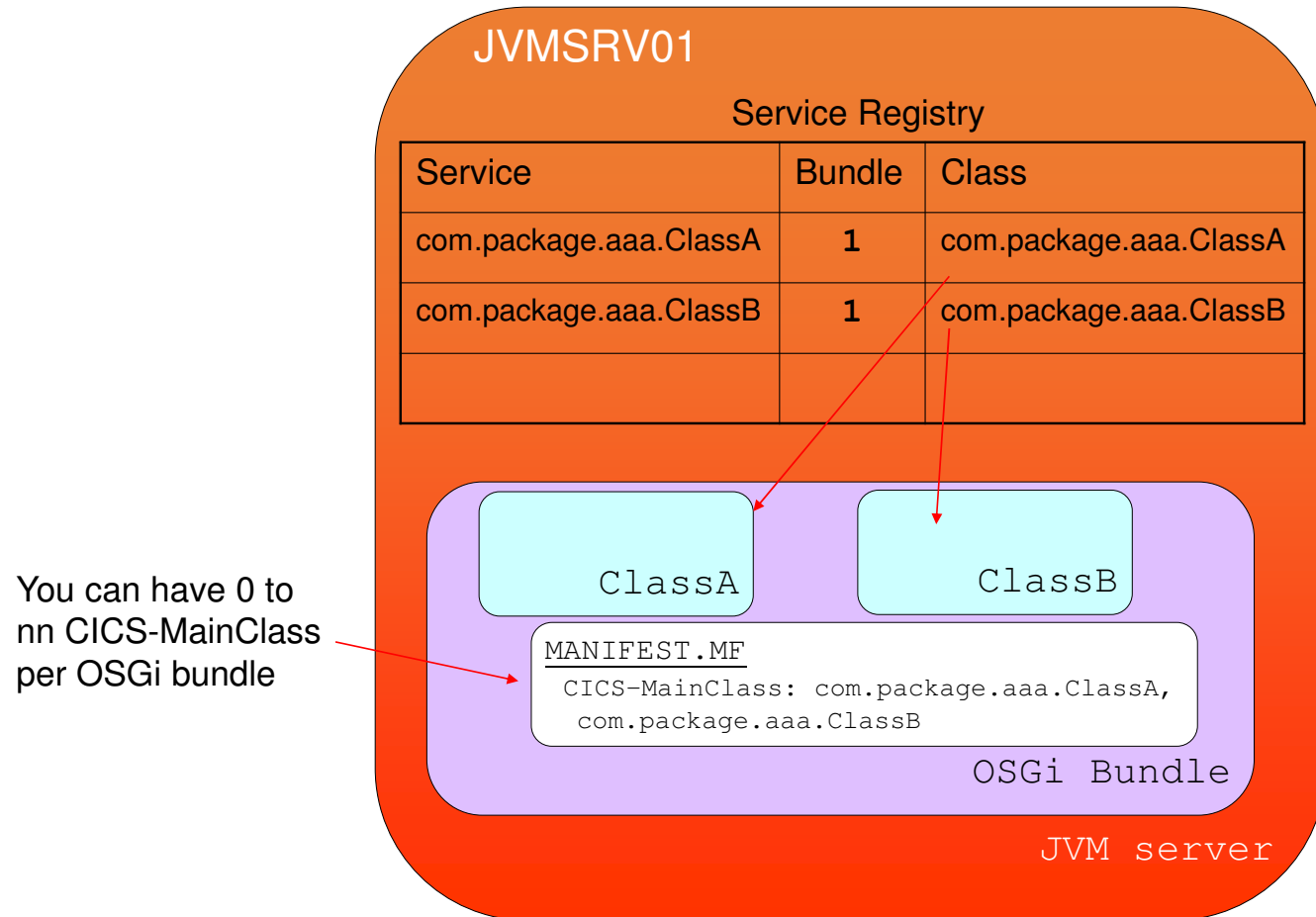


## Java in CICS: Deploying OSGi bundles

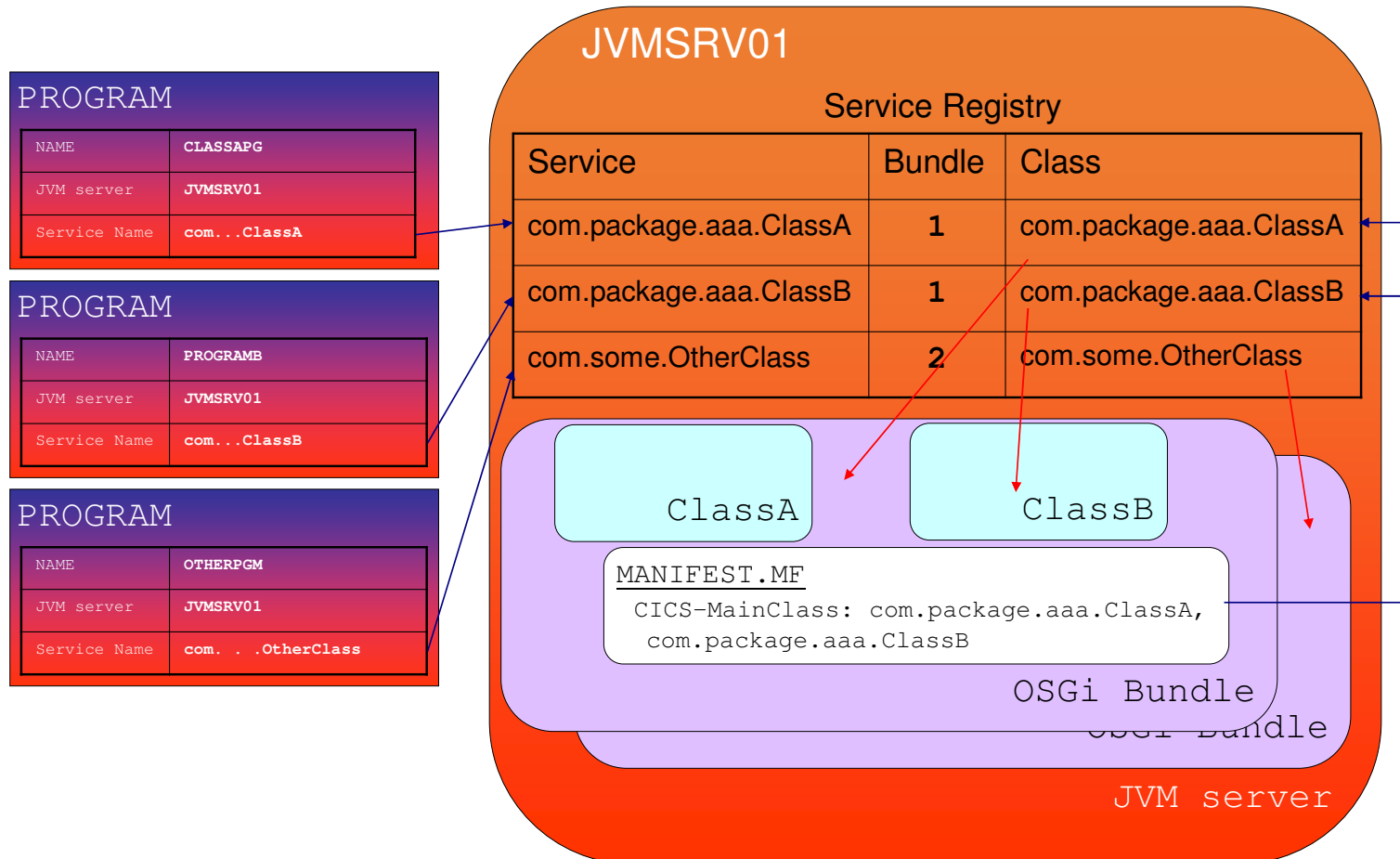


Saying it a bit differently

## OSGi Bundle installed into a CICS JVM server

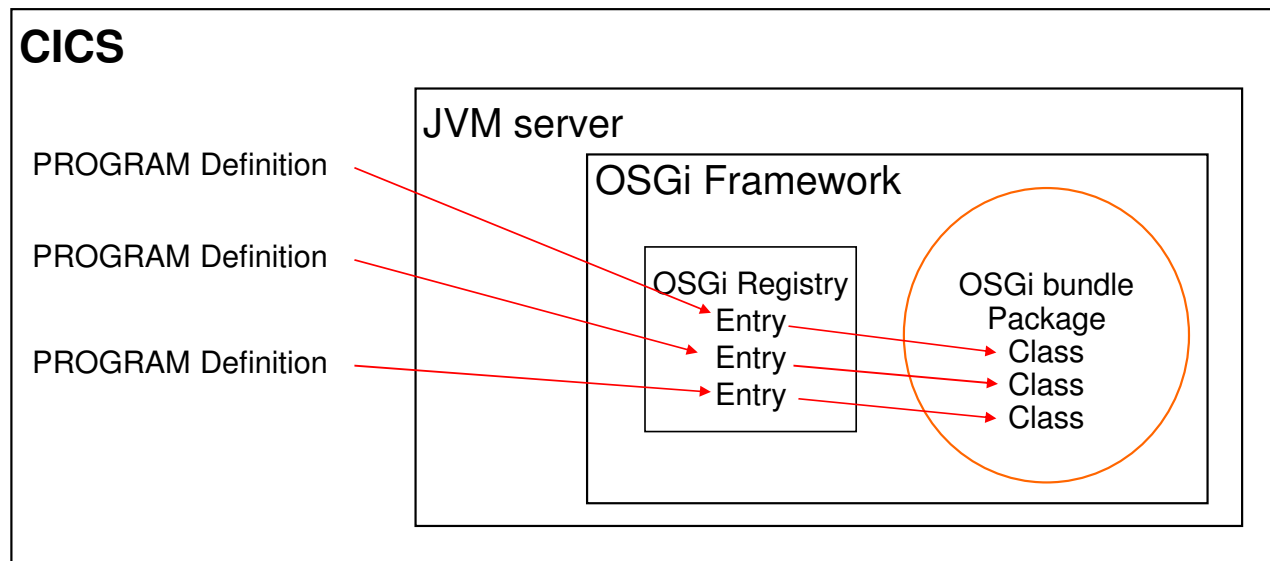


# CICS PROGRAM definitions pointing to OSGi Bundle



# Java in CICS: Same thing in a different illustration

- Execution time: OSGi bundle



A PROGRAM definition points to an entry in the OSGi Registry

The entry was put there with the CICS-MainClass entry in the manifest of the OSGi bundle

There can be 0 or more CICS-MainClass entries in a bundle

## CICS/OSGi BUNDLE/bundle resource states

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

1 – Bundle activation policy = lazy

2 – Transitory state during termination

3 – Inactive OSGI Service if duplicates existing active OSGI service, or Main class is invalid

# OSGi Best Practices to Consider

- Deploy tightly coupled OSGi bundles that comprise an application in the same CICS bundle
  - Deploy these OSGi bundles together in a CICS bundle to update and manage them together
- Avoid creating dependencies between applications
  - Instead, create a common library in a separate OSGi bundle and manage it in its own CICS bundle
  - You can update the library separately from the applications
- Follow OSGi best practices by using versions when creating dependencies between bundles
  - Using a range of versions means that an application can tolerate compatible updates to bundles that it depends on
- Set up a naming convention for the JVM servers
  - Agree on the convention between the system programmers and Java developers

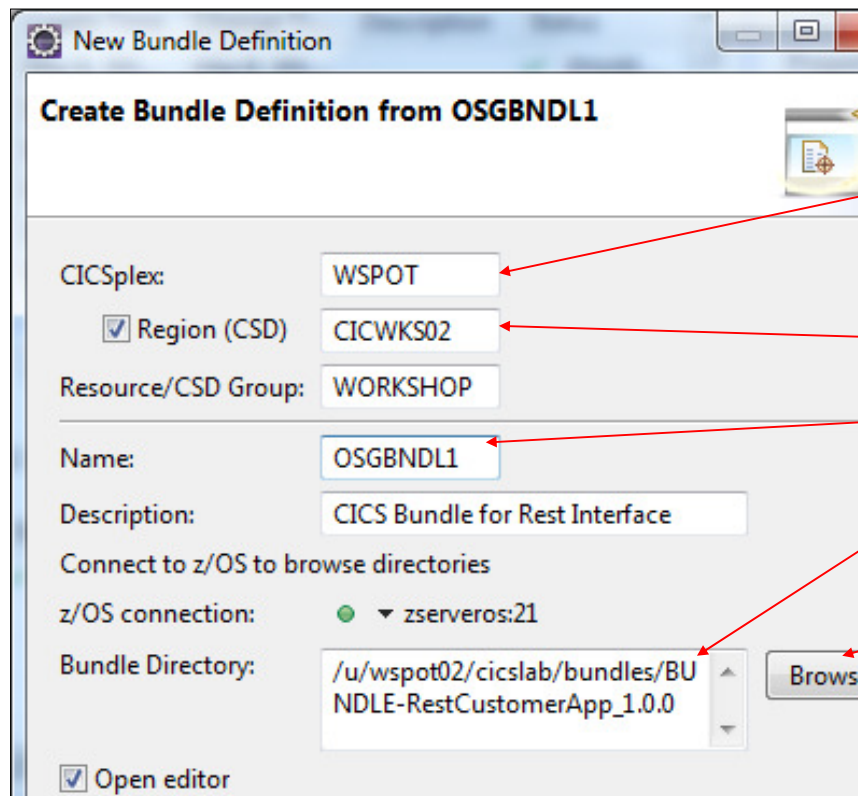
From the CICS Documentation – So a CICS BUNDLE isn't in caps

# From the CICS Explorer

This is what you do from the CICS Explorer

This shows all of the steps, so one step I already showed you will be done again

## From CICS Explorer: Define a CICS BUNDLE



New Bundle Definition

Create Bundle Definition from OSGBNDL1

CICSplex: WSPOT

☒ Region (CSD) CICWKS02

Resource/CSD Group: WORKSHOP

Name: OSGBNDL1

Description: CICS Bundle for Rest Interface

Connect to z/OS to browse directories

z/OS connection: ☒ zserveros:21

Bundle Directory: /u/wspot02/cicslab/bundles/BU NDLE-RestCustomerApp\_1.0.0

☒ Open editor

Browse...

You don't 'have' to put your Java program into a CICSplex - you may

CICS Region

Resource Name

BUNDLE Location on USS

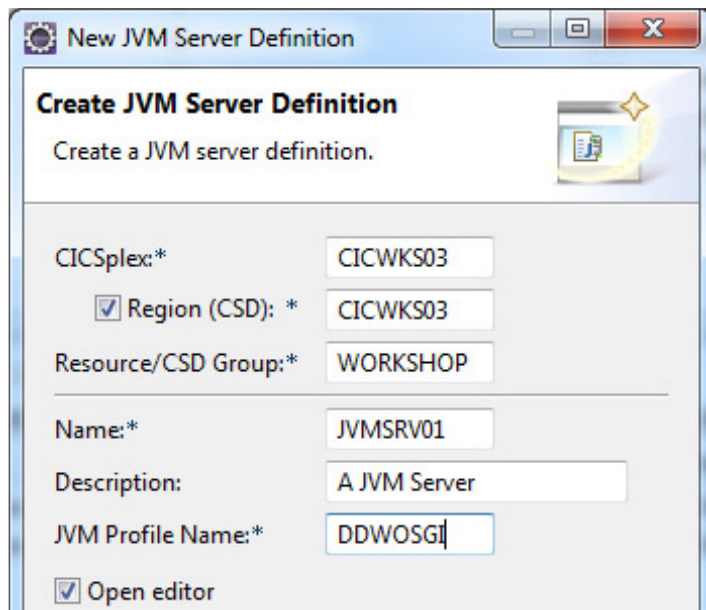
Use the Browse button to locate the BUNDLE directory on USS

We talked about this around slide 48



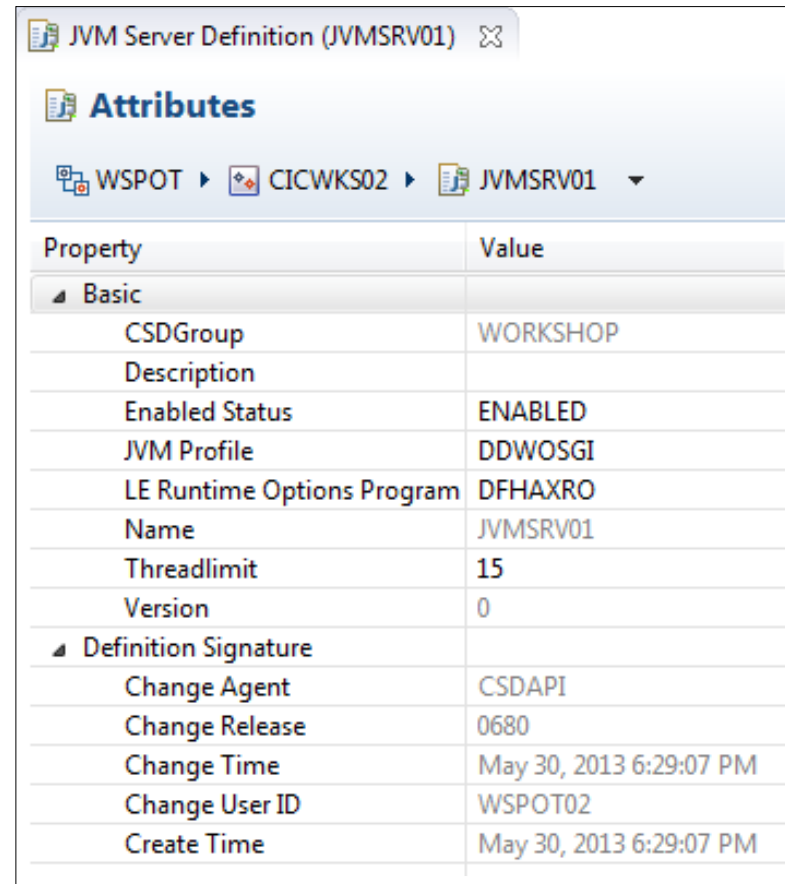
# CICS Explorer – you need a JVM server

- You need an Active JVM server to do the rest – more details later



The dialog box titled "New JVM Server Definition" contains the following fields and options:

- Create JVM Server Definition**  
Create a JVM server definition.
- CICSplex\*: CICWKS03
- ☒ Region (CSD): \* CICWKS03
- Resource/CSD Group\*: WORKSHOP
- Name\*: JVMSRV01
- Description: A JVM Server
- JVM Profile Name\*: DDWOSGI
- ☒ Open editor



JVM Server Definition (JVMSRV01)

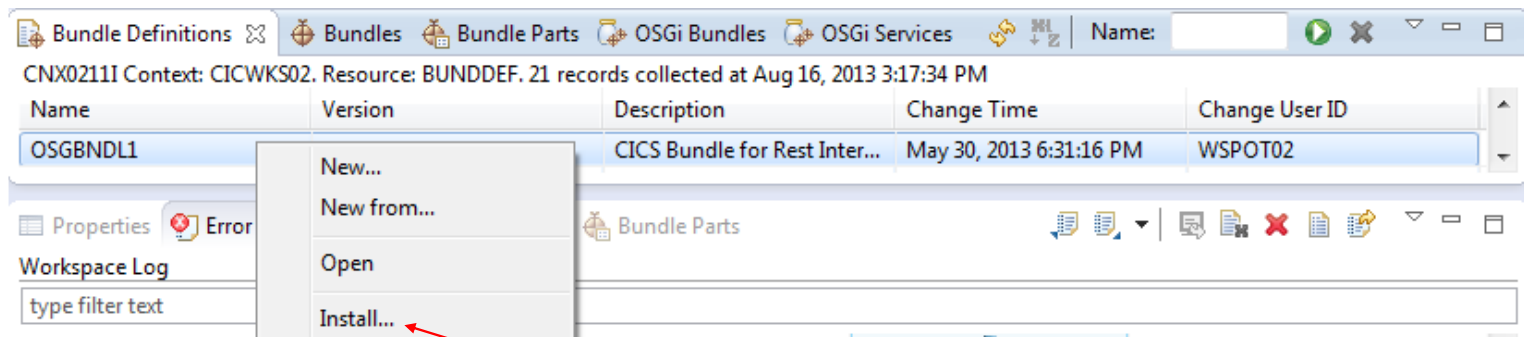
Attributes

WSPOT > CICWKS02 > JVMSRV01

Property	Value
<b>Basic</b>	
CSDGroup	WORKSHOP
Description	
Enabled Status	ENABLED
JVM Profile	DDWOSGI
LE Runtime Options Program	DFHAXRO
Name	JVMSRV01
Threadlimit	15
Version	0
<b>Definition Signature</b>	
Change Agent	CSDAPI
Change Release	0680
Change Time	May 30, 2013 6:29:07 PM
Change User ID	WSPOT02
Create Time	May 30, 2013 6:29:07 PM

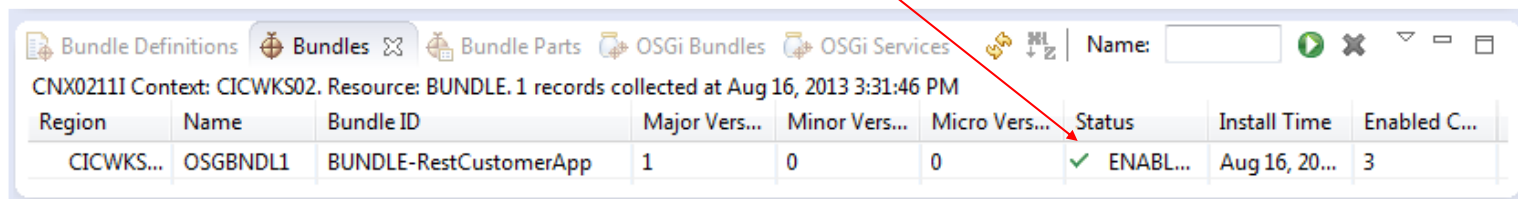
# CICS Explorer: Install the BUNDLE Definition

- After the CICS BUNDLE is on z/OS USS
- After you have defined the CICS BUNDLE (to CICS)
- After the JVM has been defined and installed into CICS
- Then install the BUNDLE in CICS



## CICS Explorer: The BUNDLE is Active

- The BUNDLE is ENABLED



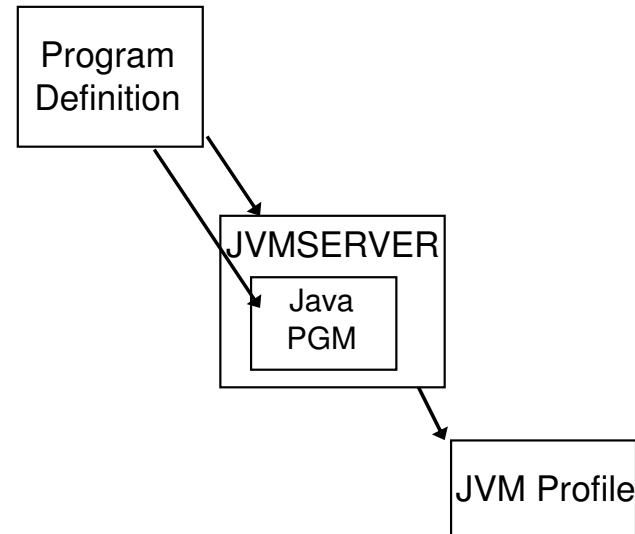
The screenshot shows the 'Bundles' tab in CICS Explorer. The context is 'CNX0211I Context: CICWKS02. Resource: BUNDLE. 1 records collected at Aug 16, 2013 3:31:46 PM'. A table lists the bundle details, with a red arrow pointing to the 'Status' column.

Region	Name	Bundle ID	Major Vers...	Minor Vers...	Micro Vers...	Status	Install Time	Enabled C...
CICWKS...	OSGBNDL1	BUNDLE-RestCustomerApp	1	0	0	✓ ENABL...	Aug 16, 20...	3

- The Java programs are in the JVM server at this point
- We need a PROGRAM definition to execute the Java program
  - Explicit/Implicit
- Then it/they can be executed in CICS?? – we need a few more displays to verify everything is ok (coming up on the next few slides)

# CICS Java PROGRAM Definition

- Program Definition
  - When not using @CICSProgram
  - JVM = YES
  - Specifies OSGI Service entry or alias
    - CICS-MainClass in Manifest
  - Points to JVMSERVER definition
  - Concurrency = Required
  - Execkey = CICS
- JVMProfile (zFS file)
  - Startup characteristics of the JVM
  - Specifies OSGi 'middleware' bundles
  - Java options (e.g. -Xmx=nnM)
  - Some drivers
  - Documented in CICS documentation
  - Covered later



# CICS Java Program Definition

Don't forget to install the  
**PROGRAM** definition

**Create Program Definition**

CICSplex: WSPOT

☒ Region (CSD) CICWKS02

Resource/CSD Group: WORKSHOP

Name: CUSTPGMJ

Description: Java program that works with customers

Program Type

☐ Assembler, C/C++, COBOL, or PL/I

☒ Java

Service Name: com.ibm.sample.cics.xml.java.CustPgmJ

JVM Server: JVMSRV01

☒ Open editor

Finish Cancel

Resource Group

Name to CICS (8-characters)

Comment

This is a Java program

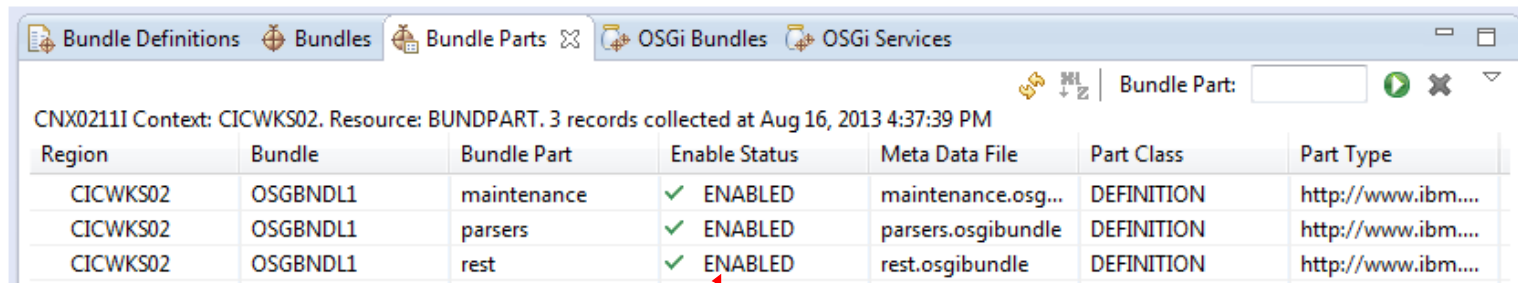
OSGi Service name

Name of the JVM server

A view with all attributes will be  
opened

# CICS Explorer: BUNDLE Parts

- A BUNDLE part is one of the elements in a BUNDLE
- We had 3 OSGi bundles in our CICS BUNDLE, so we have 3 BUNDLE parts



Region	Bundle	Bundle Part	Enable Status	Meta Data File	Part Class	Part Type
CICWKS02	OSGBNDL1	maintenance	✓ ENABLED	maintenance.osg...	DEFINITION	http://www.ibm....
CICWKS02	OSGBNDL1	parsers	✓ ENABLED	parsers.osgibundle	DEFINITION	http://www.ibm....
CICWKS02	OSGBNDL1	rest	✓ ENABLED	rest.osgibundle	DEFINITION	http://www.ibm....

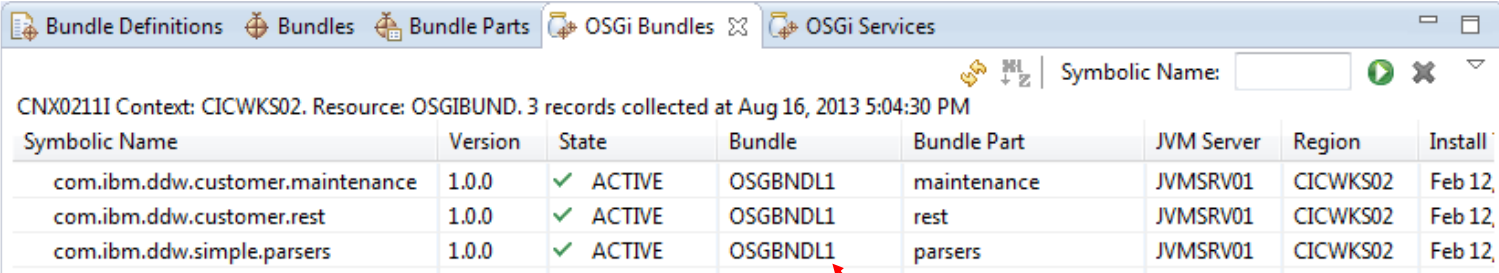
**Must be  
ENABLED**

The name= parameter of the  
cics.xml file for this BUNDLE

Name of the .osgibundle  
file in the manifest in the  
BUNDLE definition

# CICS Explorer: OSGi Bundles

- These are your OSGi bundles
- They need to be in an Active state, or if you used Lazy loading, it will be in a PENDING state until first used
  - Do not use lazy activation policies for OSGi bundles that declare a CICS-MainClass



The screenshot shows the 'OSGi Bundles' tab in CICS Explorer. The title bar includes 'Bundle Definitions', 'Bundles', 'Bundle Parts', 'OSGi Bundles', and 'OSGi Services'. Below the title bar, there is a search bar for 'Symbolic Name' and a status bar indicating 'CNX0211I Context: CICWKS02. Resource: OSGIBUND. 3 records collected at Aug 16, 2013 5:04:30 PM'. The main area contains a table with the following data:

Symbolic Name	Version	State	Bundle	Bundle Part	JVM Server	Region	Install
com.ibm.ddw.customer.maintenance	1.0.0	✓ ACTIVE	OSGBNDL1	maintenance	JVMSRV01	CICWKS02	Feb 12,
com.ibm.ddw.customer.rest	1.0.0	✓ ACTIVE	OSGBNDL1	rest	JVMSRV01	CICWKS02	Feb 12,
com.ibm.ddw.simple.parsers	1.0.0	✓ ACTIVE	OSGBNDL1	parsers	JVMSRV01	CICWKS02	Feb 12,

The symbolic name you used in your bundle manifest

The version in the bundle manifest

Must be Active

Name of the CICS BUNDLE definition

name= parm in the cics.xml BUNDLE manifest

## CICS Explorer: OSGi Services

- There were only two CICS-MainClass
- These must be ACTIVE, or you will get a AJ04 abend when you try to execute the 'program'

The screenshot shows the 'OSGi Services' tab in CICS Explorer. The context is 'CNX0211I Context: CICWKS02. Resource: OSGISERV. 2 records collected at Aug 16, 2013 5:23:19 PM'. The table below lists two services, both in an 'ACTIVE' state. Red arrows point from explanatory text boxes to specific columns in the table.

Service Name	Bundle Part	OSGi Bundle	Version	Service St...	Bundle
com.ibm.ddw.customer.maintenance.CustPgmJ	maintenance	com.ibm.ddw.customer.maintenance	1.0.0	ACTIVE	OSGBNDL:
com.ibm.ddw.customer.rest.CustResJ	rest	com.ibm.ddw.customer.rest	1.0.0	ACTIVE	OSGBNDL:

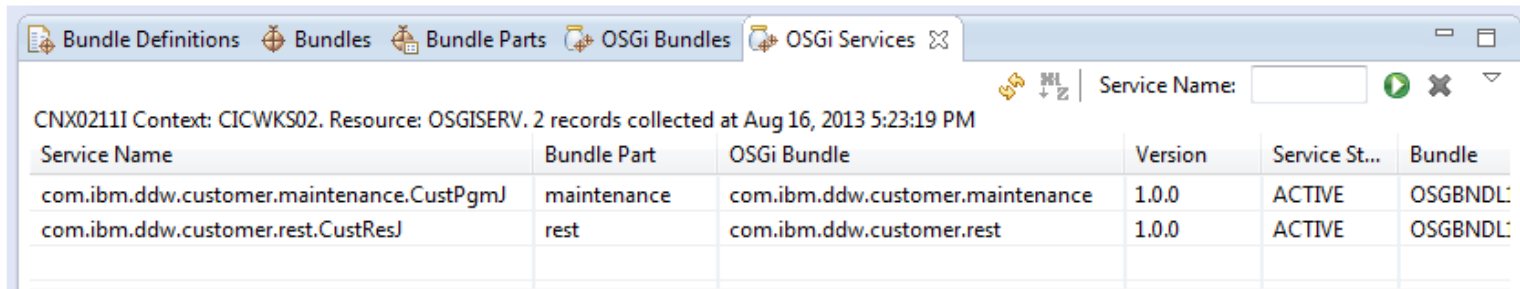
Annotations (in red boxes):

- The value on the CICS-MainClass**: Points to the 'Service Name' column.
- PROGRAM definition must have this, else AJ04 abend**: Points to the 'Bundle Part' column.
- name= parm in the cics.xml BUNDLE manifest**: Points to the 'OSGi Bundle' column.
- OSGi bundle name**: Points to the 'OSGi Bundle' column.
- Must be Active**: Points to the 'Service St...' column.
- Name of the CICS BUNDLE definition**: Points to the 'Bundle' column.



# CICS Explorer

- There are probably more columns of information



The screenshot shows the CICS Explorer interface with the 'OSGi Services' tab selected. The window title bar includes tabs for 'Bundle Definitions', 'Bundles', 'Bundle Parts', 'OSGi Bundles', and 'OSGi Services'. Below the tabs, there is a search bar labeled 'Service Name:' and a status bar indicating 'CNX0211I Context: CICWKS02. Resource: OSGISERV. 2 records collected at Aug 16, 2013 5:23:19 PM'. The main area displays a table with the following data:

Service Name	Bundle Part	OSGi Bundle	Version	Service St...	Bundle
com.ibm.ddw.customer.maintenance.CustPgmJ	maintenance	com.ibm.ddw.customer.maintenance	1.0.0	ACTIVE	OSGBNDL:
com.ibm.ddw.customer.rest.CustResJ	rest	com.ibm.ddw.customer.rest	1.0.0	ACTIVE	OSGBNDL:

You can add more columns or remove columns to/from any CICS Explorer resource display. They all work the same.

## CICS Explorer vs. CEMT

- With CEMT:
  - You can display a BUNDLE definition
- With the CICS Explorer
  - You can display a BUNDLE definition
  - You can also display:
    - BUNDLE parts
    - OSGi bundles
    - OSGi Services –
      - Needed to verify the CICS-MainClass
      - And that it is ACTIVE
  - You can see the level of OSGi bundles

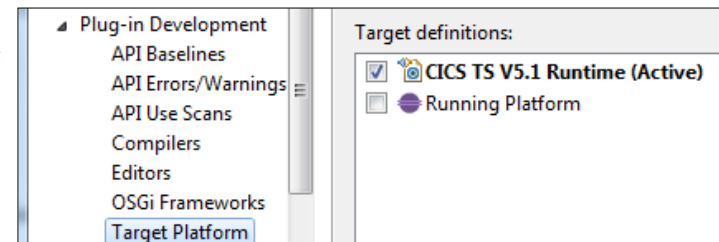
# Development Summary

# CICS Explorer SDK – Development - Summary

## 1. Install CICS Explorer SDK on your desktop

## 2. Set Target Platform

- Sets JCICS level
- **Needs** to be done **once** per workspace
- Window -> Preferences...  
Plug-in Development-> Target Platform...  
From Target Platform page click the Add button  
From the Target Definition click Template and use the pull-down to select CICS TS V5.x Runtime



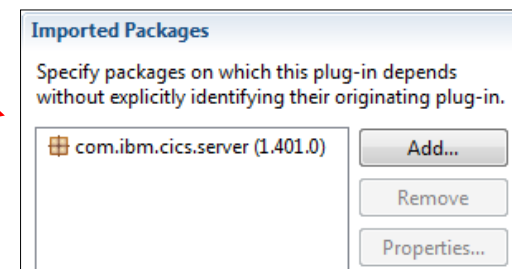
## 3. Create New OSGi Project

- New -> Plug-in Project (OSGi framework=standard)

Develop

## 4. Provide access to JCICS package

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

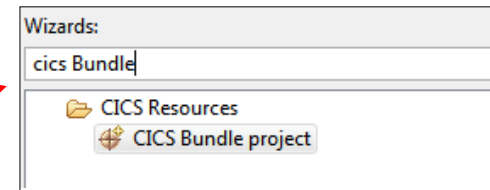


## 5. Use the workbench to import or create Java class

# CICS Explorer SDK - Development - Summary

## 6. Create CICS BUNDLE

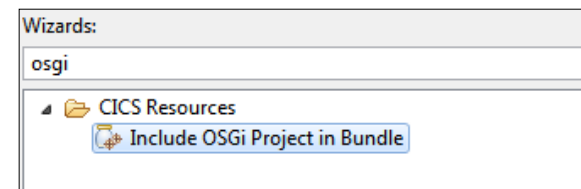
- New -> CICS Bundle Project



## 7. Add your OSGi bundle to your CICS BUNDLE

- New -> Include OSGi Project in Bundle

Encapsulate OSGi  
bundles into a CICS  
BUNDLE



# CICS Explorer SDK – Development - Summary

## Export to z/OS

### 8. Connect CICS Explorer to z/OS

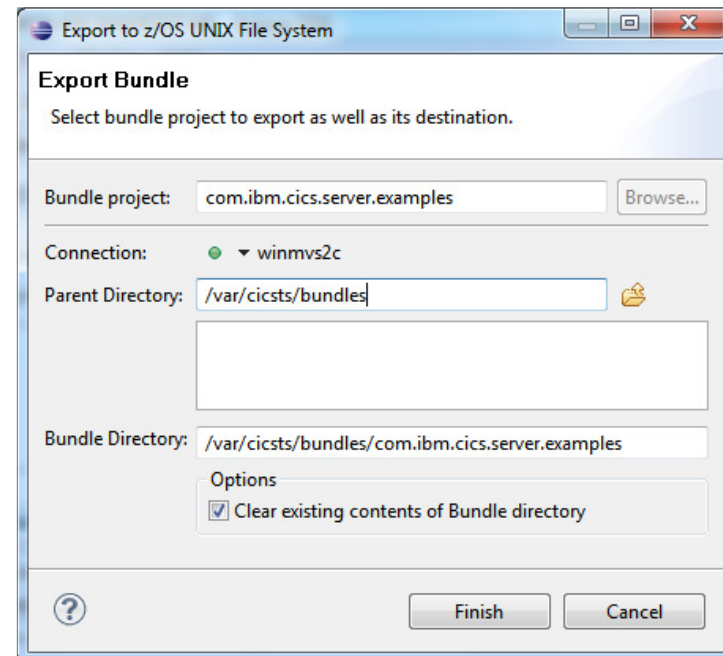
- Windows->Open Perspective  
->z/OS or RSE

### 9. Export CICS BUNDLE to USS

- From workstation to  
z/OS UNIX File System

### 10. Provide CICS region id read access

- mkdir /var/cicsts/bundles
- chmod 750 /var/cicsts/bundles

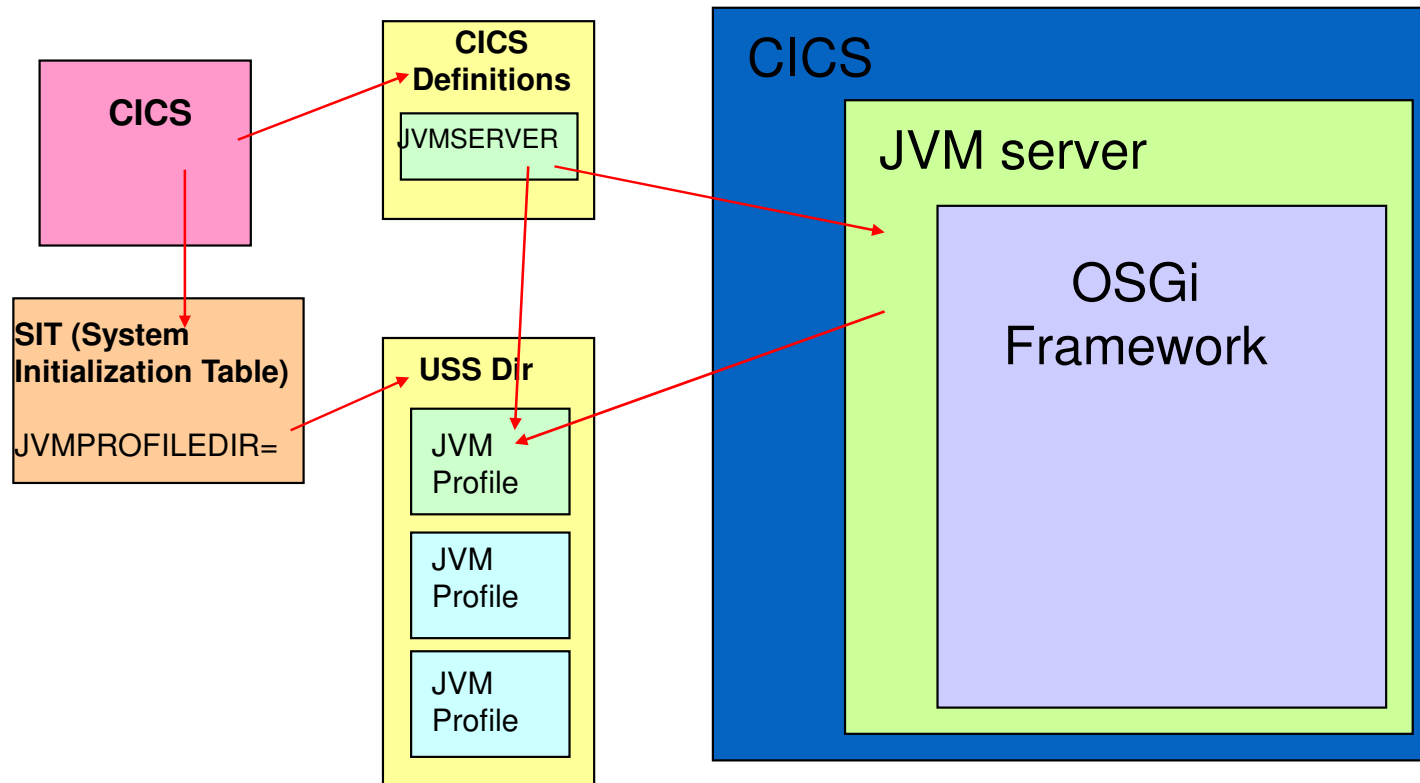


Then you are ready  
to install into CICS

# JVM Server

# CICS JVM Environment

- The CICS JVM for OSGi





# JVMProfile for JVM server

## Must use

- CICS TS (any supported release)
  - Java V8 64-bit

‘gencon’ is default GC policy (more on GC in a different topic)

```
#####  
# Selected parts of a JVMProfile  
#####  
JAVA_HOME=/usr/lpp/java/J8.0_64  
WORK_DIR=.  
#STDIN=  
#STDOUT=  
#STDERR=  
#JVMTRACE=  
  
#### OSGi items  
# OSGI_BUNDLES= (used for ‘middleware’ bundles)  
#OSGI_STARTUP_TIMEOUT= 60 (startup timeout)  
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream  
  
#### JVM Options  
-Xgcpolicy:gencon  
-Xms16M  
-Xmx32M
```

## JVM profile for OSGi (1 of 4)

- Sample is DFHOSGI
- OSGI\_BUNDLES= - middleware OSGi bundles
- OSGI\_FRAMEWORK\_TIMEOUT=nn - timeout for OSGi infrastructure
- JAVA\_HOME=/usr/lpp/java/J8.0\_64/
- Options starting with
  - -D are JVM system properties
  - -X are treated as JVM command-line options
  - Options starting with '-' are passed to the JVM with being parsed by CICS
- Symbols
  - &APPLID; - represents the APPLID of this region
  - &DATE; - the current date in the format Dyymmdd
  - &JVMSERVER; - name of the JVMSERVER (unique output of dumps)
  - &TIME; - JVM start time in the format Thhmmss
- LIBPATH\_SUFFIX=
  - Library files that are to be added after the LIBPATH that CICS builds

## JVM profiles for OSGi (2 of 4)

- JAVA\_DUMP\_TDUMP\_PATTERN=
  - Can use symbols (&APPLID, etc)
  - Name of Java TDUMP if one is written, in the event of a JVM abend
- JVMTRACE={&APPLID;.&JVMSERVER;.Dyyyymmdd.Thhmmss.dfhjvmtrc|file\_name}
- PRINT\_JVM\_OPTIONS={YES|NO}
  - If yes, the JVM startup options are also printed to SYSPRINT, including those not visible in JVM profile
- WORK\_DIR={.|directory\_name}
  - The working directory for this JVM
  - A '.' says 'use CICS home directory'
  - If WORK-DIR is omitted, /tmp is used
- STDOUT={&APPLID;.&JVMSERVER;.Dyymmdd.Thhmmss.dfhjvmout|file\_name}
- STDERR={&APPLID;.&JVMSERVER;.Dyymmdd.Thhmmss.dfhjvmerr|file\_name}
- STDIN={file\_name} (very rarely used)
- USEROUTPUTCLASS=
  - Name of a routine to process stdout and stderr messages
- TZ – the local timezone

## JVM profiles for OSGi (3 of 4)

- -Xrunjdwp or -agentlib:runjdwp
  - Specifies whether debugging support is enable in this JVM
  - E.g. -agentlib:jdwp=transport=dt\_socket,server=y,address=8000,suspend=n
- -Xms
  - Specifies the initial size of the heap
- -Xmx
  - Specifies the maximum size of the heap
- -Xscmx
  - Specifies the size of the share class cache
- -Xshareclasses
  - The JVM connects to a cache or creates one if it doesn't exist
- -Xhealthcenter or -agentlib:healthcenter
  - The IBM HealthCenter can be attached at the specified port

## JVM Profile for OSGi (4 of 4)

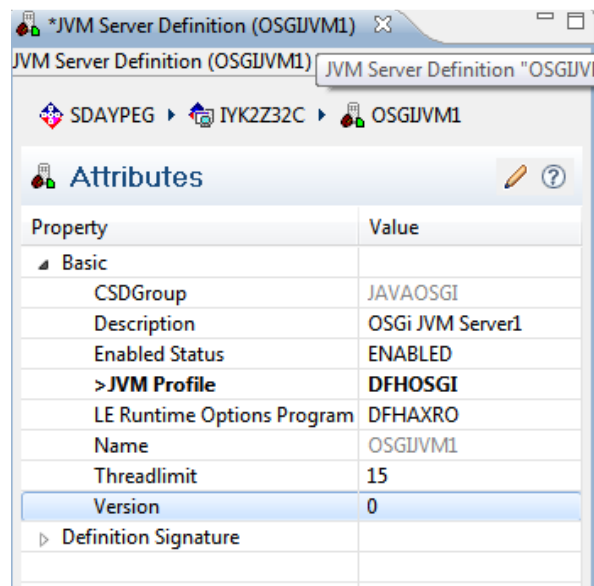
- WORK\_DIR contains:
  - Log files
    - <applid><jvmserver>.dfhjvmerr
    - <applid><jvmserver>.dfhjvmout
    - <applid><jvmserver>.dfhjvmtrc
    - <applid>/<jvmserver>/configuration
      - OSGi cache directory
      - OSGi log

# JVMSERVER Definition

# JVM servers

- Max number of T8 TCBs - CICS TS V5 = 2000
- Max T8 TCB/threads per JVM server = 256
- Applications must be deployed as OSGi bundles
  - **Except** for Axis2 programs (not OSGi) and Liberty programs (optional)
- It is possible to use multiple JVM servers per region
  - To simplify administration (i.e different brands)
  - Going beyond the bounds of what a single JVM can provide (256 threads)
  - If using conflicting drivers i.e different middleware native libraries.
  - If using non-OSGi environments (such as Axis2 and OSGi user applications)
  - If using performance sensitive applications that are sensitive to long GC times freeing heap from other applications
  - If using Operation Decision Management (can run in its own JVM)

# Defining a JVM server



- JVM Profile
  - JVM profile in HFS/zFS in JVMPROFILEDIR (in the SIT)
  - DFHOSGI is default
- LE Runtime Options
  - LE storage options
  - Defaults to DFHAXRO
- Threadlimit
  - Max number of T8 threads in JVM server
  - Default is 15, minimum=1, max=256

Note: DB2 in CICS TS V5 requires no TCB switch

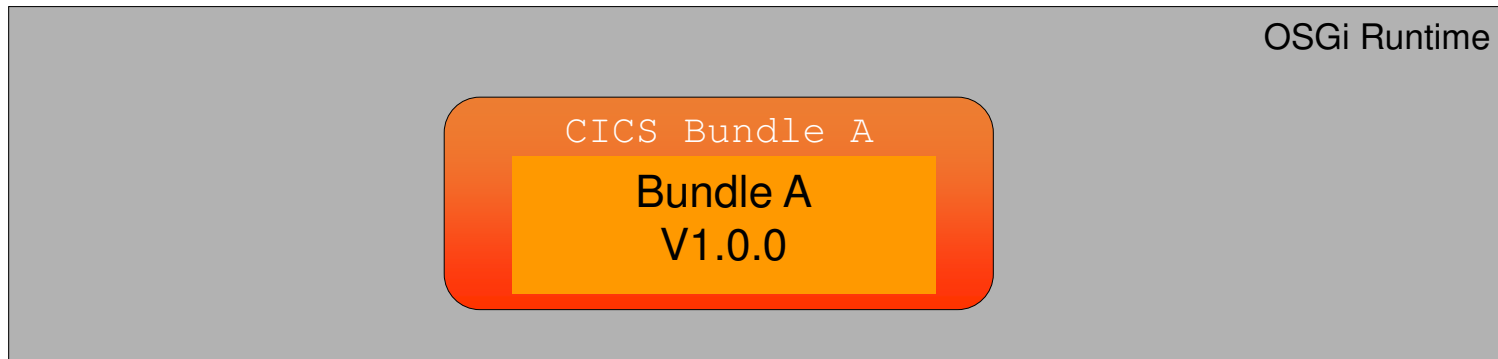


# Updating OSGi Bundles

# OSGi Versioning

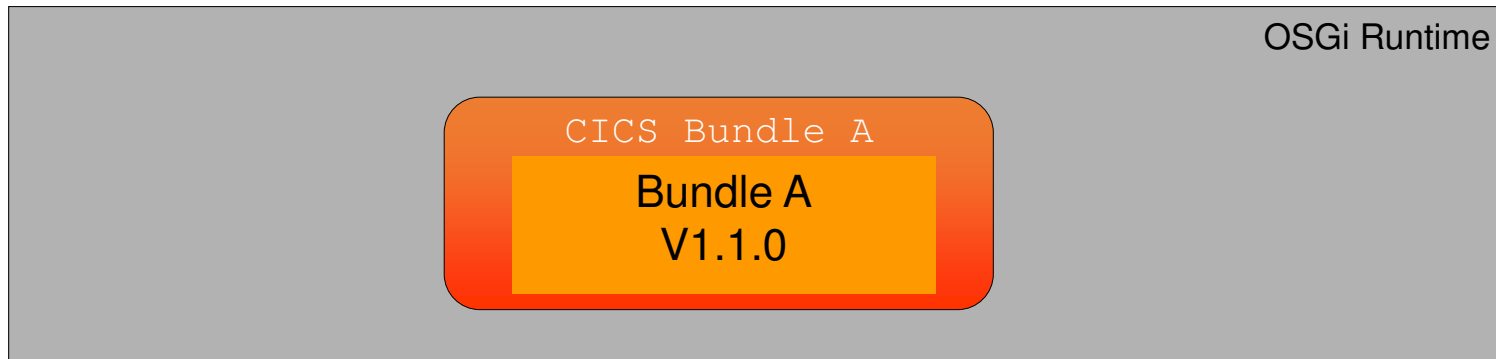
- Old way:
  - **Restart:** Redeploy JAR or class, and restart JVM (or CICS)
- New options:
  - **SET BUNDLE** COPY(PHASEIN) support
    - Enables registration of a new OSGi bundle
  - **Refresh** JAR and redeploy BUNDLE
    - Full disable/delete/install/enable lifecycle on the BUNDLE
  - **Version** JAR and deploy new BUNDLE and disable old BUNDLE
    - Ability to toggle between 2 OSGi bundles in same JVM
  - **Alias:** Version JAR and add new alias for service, deploy new BUNDLE and add a new PROGRAM definition
    - Run two different versions of JAR in same JVM

## OSGi Initial install



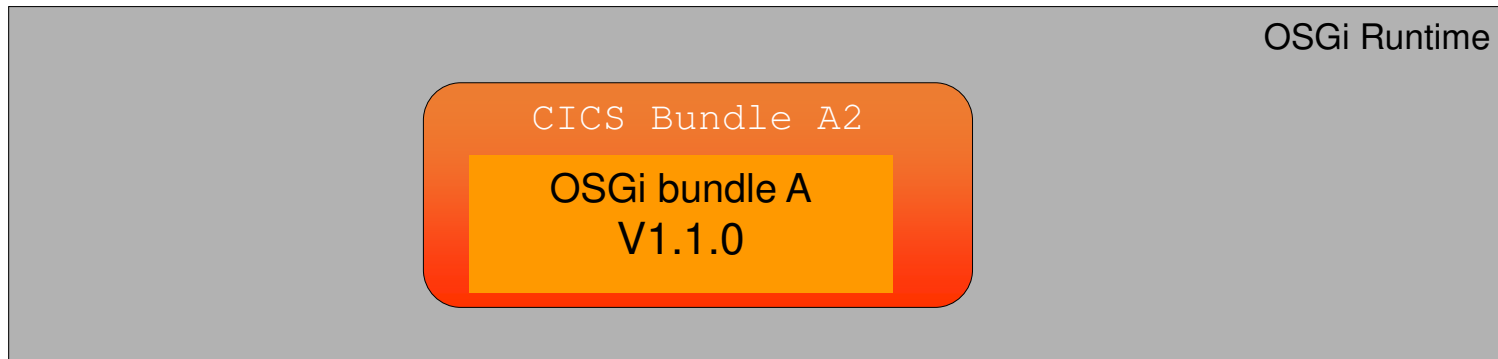
- Install new CICS Bundle A V1.0.0

## OSGi Update – Scenario 1



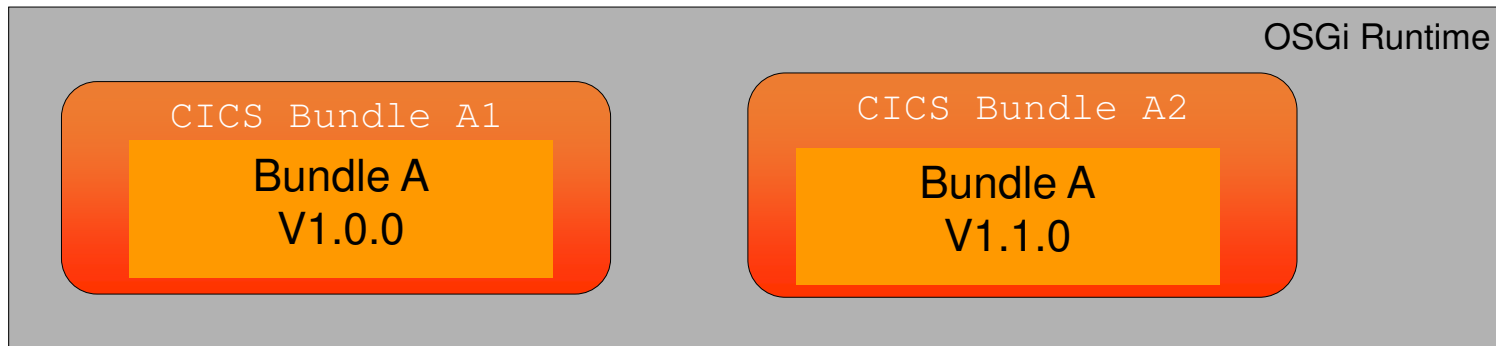
- SET BUNDLE COPY(PHASEIN) support
  - Enables registration of a new OSGi bundle replacing any currently registered version
  - New requests will use the new version
  - Existing requests will use the old version until complete

## OSGi Update – Scenario 2



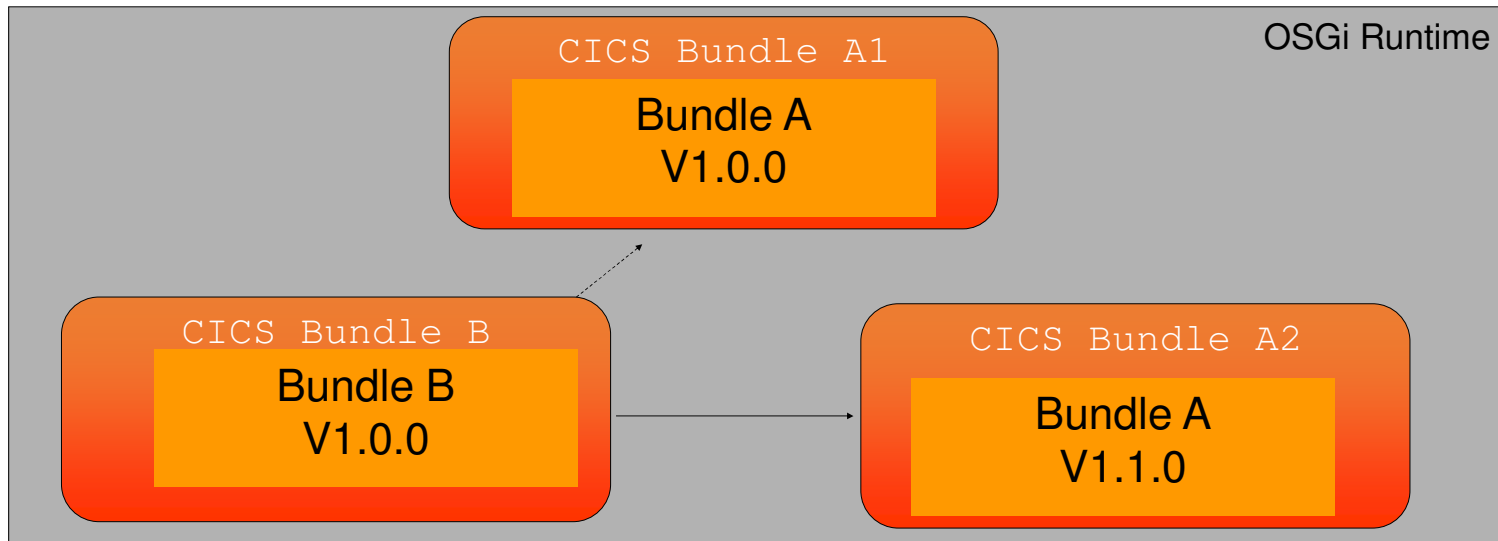
1. Disable CICS BUNDLE V1.0.0
2. Discard CICS BUNDLE V1.0.0
3. Install CICS BUNDLE V1.1.0
  1. Reads new CICS BUNDLE, cics.xml and .osgibundle files
  2. Resolves bundle dependencies
  3. Activates OSGISERVICE

## OSGi Update – Scenario 3A



1. Install and Enable new version V1.1.0
  - V1.1.0 OSGi bundle Resolved
  - V1.1.0 OSGi 'Service' *Inactive* as duplicates A1
2. Disable V1.0.0
  - V1.1.0 OSGi 'Service' goes *Active*
  - New application used for new transactions
3. Uninstall V1.0.0

## OSGi Update - Scenario 3B

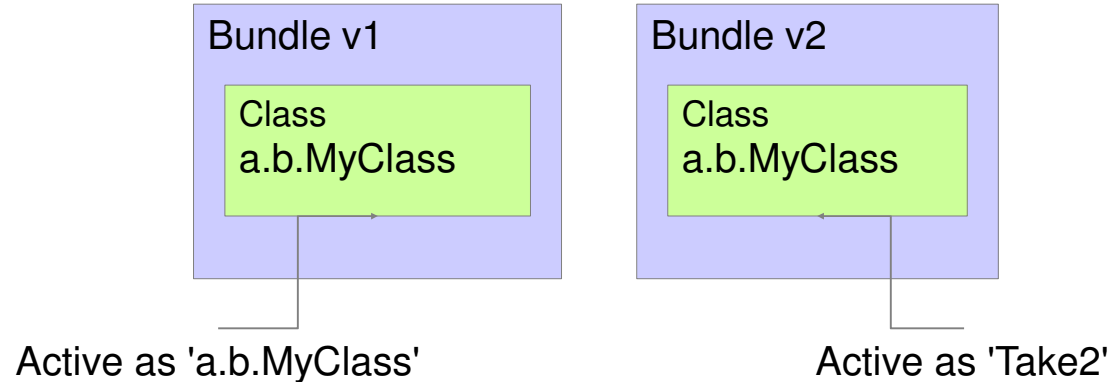


1. Install new CICS Bundle A2
2. Enable new CICS Bundle A2
3. Disable old CICS Bundle A1
4. Uninstall old CICS Bundle A1
5. If bundle A is imported into bundle B then recycle (disable/install) bundle B to refresh the classloader for bundle B

## Program to OSGi bundle

```
CICS-MainClass: a.b.MyClass;alias="Take2"
```

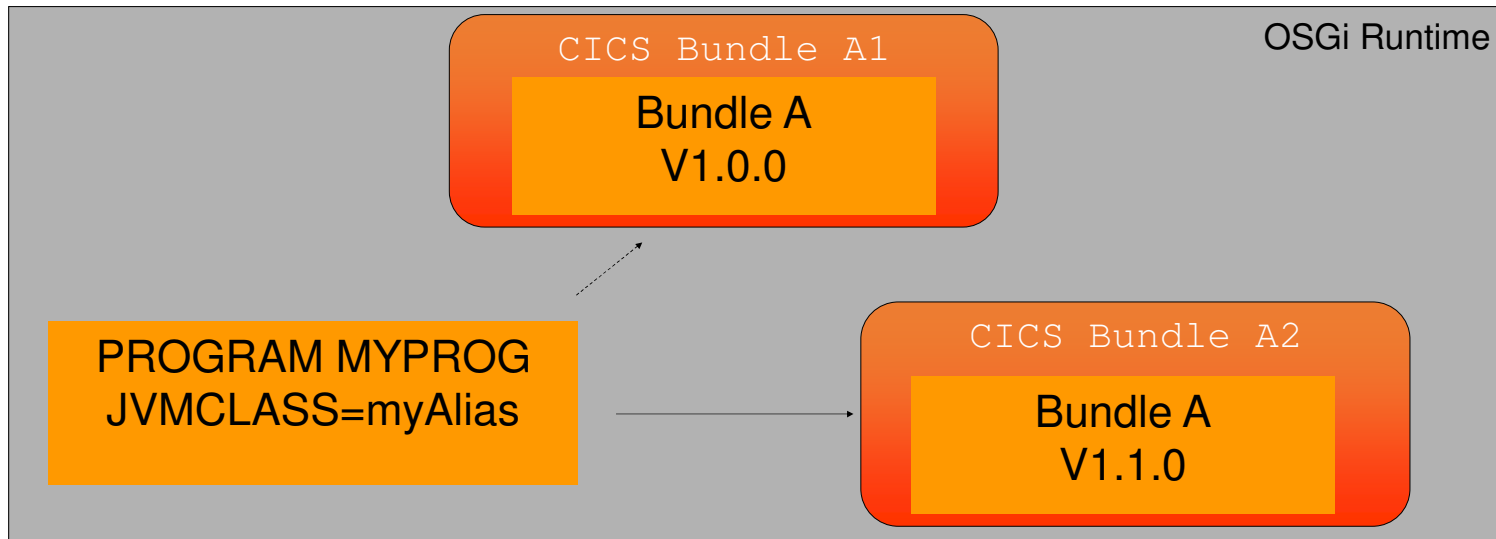
- 'alias' attribute can be added to the CICS-MainClass header.
- Service is registered as the alias, instead of the service name (if it already exists)
- Service can now be run with "class name" set to "Take2".
- Both services can be run at once (using two OSGi bundles)



This requires a new PROGRAM definition



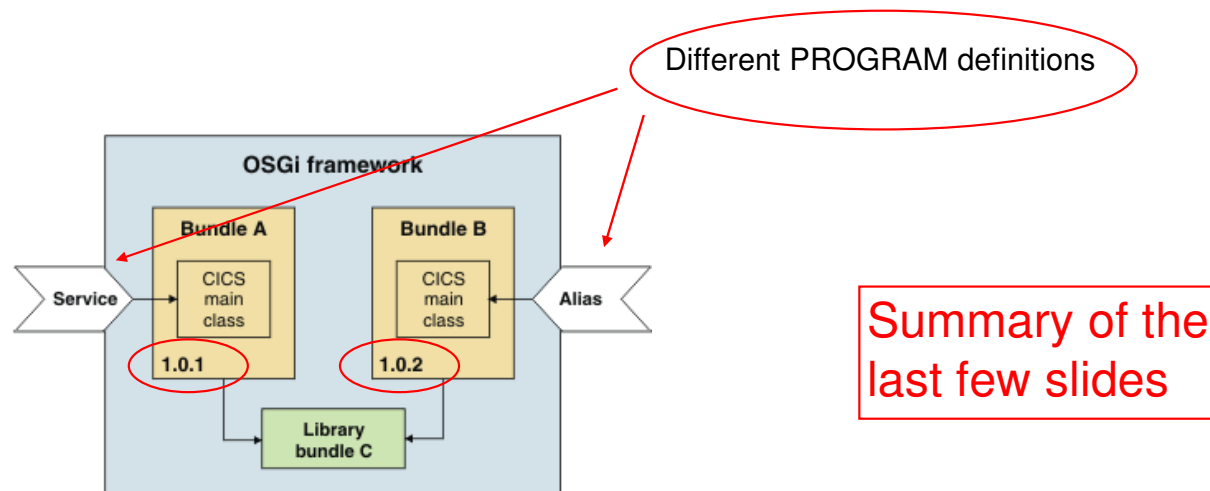
## OSGi Update – Scenario 4



1. Install new version of CICS Bundle A2 (with Alias name for Service)
  - CICS-MainClass: package.myClass;alias=myAlias
2. Enable new CICS Bundle A2
  - Bundle activates
3. Create a new PROGRAM definition with reference to myAlias as main class
  - **JVMCLASS=myAlias**

# Running multiple versions of the same app

- The OSGi framework supports running multiple versions of an OSGi bundle in a framework, so you can phase in updates to the application without interrupting availability.
- You cannot have multiple versions of the same OSGI 'service'
- If different versions of the OSGi bundle have the same CICS main class, you can use an alias to override the duplicate service
- Specify the alias on another PROGRAM resource to make the application available.



## OSGi versioning

- BUNDLE lifecycle controls OSGi bundle and OSGi service activation
  - Bundle not refreshed until it is installed
  - JVM server restart or CICS region warm start will recover BUNDLES, OSGi framework, and reload JARs
- Duplicate OSGi bundles can be deployed in CICS if at different version levels
- OSGi Service entries can't be duplicated
  - The duplicate OSGi service will be inactive
  - Aliases can be used to have different PROGRAM entry points

Recap

# Summary

What is OSGi?

CICS and OSGi

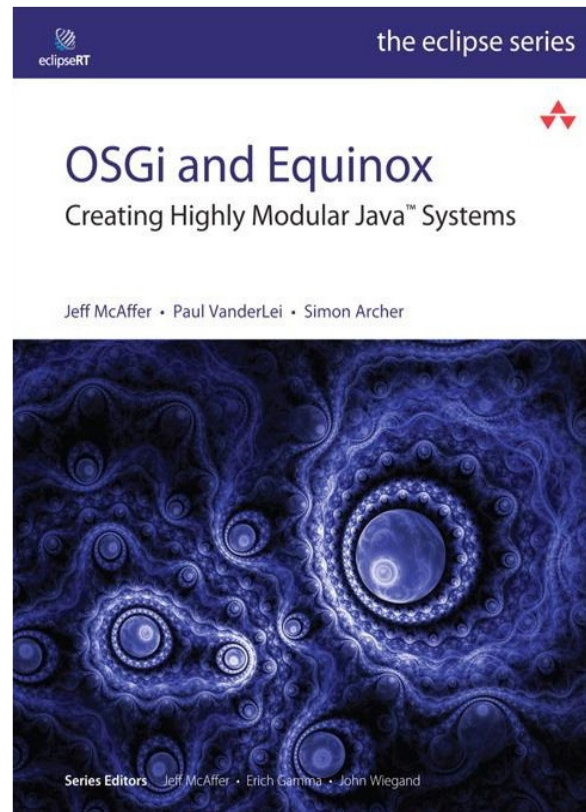
- Developing your program
- Defining your program as an OSGi program
- Put your OSGi bundle into a CICS BUNDLE
- Transfer your program to z/OS
- Define your program to CICS

Updating OSGi bundles

# Lab Exercises

- L10 – CICS Explorer (not really anything to do with Java)
- L34 – Simple OSGi program
- L27 – An OSGi bundled Java program used as the target of a CICS web service
- L32 – Converting an existing CICS Java program to use OSGi (goes over the three conversion techniques)

## Further Reading



## Resources (1 of 2)



- IBM CICS Transaction Server for z/OS
  - <http://www.ibm.com/cics/>
  - CICS Documentation
  - Javadoc
- IBM CICS Transaction Gateway – [ibm.com/software/http/cics/ctg/](http://www.ibm.com/software/http/cics/ctg/)
- WebSphere Application Server - [ibm.com/software/webervers/appserv/was/](http://www.ibm.com/software/webervers/appserv/was/)
- WebSphere MQ - [ibm.com/software/integration/mqfamily/index.html](http://www.ibm.com/software/integration/mqfamily/index.html)
- IBM Developer for z/OS and IBM Rational Application Developer
  - <https://www.ibm.com/products/developer-for-zos/details>
  - <https://www.ibm.com/products/rad-for-websphere-software>
- Redbook – Java for CICS SG24-5275 and SG24-8038
- Diagnostic Guide: <http://www.ibm.com/developerworks/java/jdk/diagnosis/>
- Performance: [www.ibm.com/software/http/cics/library/whitepapers/java\\_benchmark\\_whitepaper.pdf](http://www.ibm.com/software/http/cics/library/whitepapers/java_benchmark_whitepaper.pdf)

## Resources (2 of 2)

- **IBM Java InfoCenter:** <http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp>
- **Class Sharing:** <http://www.ibm.com/developerworks/java/library/j-ibmjava4/>
- **IBM Developer Kits:**  
<http://www.ibm.com/developerworks/java/jdk/docs.html>
- **IBM 64-bit SDK for z/OS, Java Technology Edition, V6:**  
<http://www-03.ibm.com/systems/z/os/zos/tools/java/products/j6pcont64.html>
- **IBM JDK Diagnosis Documentation:**  
<http://www.ibm.com/developerworks/java/jdk/diagnosis/60.html>
- **Java Troubleshooting:**  
[http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic=/com.ibm.java.doc.igaa/\\_1vg00011e17d8ea-1163a087e6c-7ffe\\_1001.html](http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic=/com.ibm.java.doc.igaa/_1vg00011e17d8ea-1163a087e6c-7ffe_1001.html)
- **IBM Developer Kits:**  
<http://www.ibm.com/developerworks/java/jdk/docs.html>
- **Visual Performance Analyzer:** <http://www.alphaworks.ibm.com/tech/vpa>
- **Garbage collection:**  
<http://www.ibm.com/developerworks/forums/thread.jspa?messageID=13940438>  
<http://www.ibm.com/developerworks/java/library/j-ibmtools2/index.html>





