IBM – CICS Integration Workshop

L34-A Simple OSGi Program with CICS Explorer

Lab Version V61.02.zVA

July 30, 2024

Please send any comments on this lab exercise to:
Steve Fowlkes, Eric Higgins, or Leigh Compton
fowlkes@us.ibm.com
eric.higgins@us.ibm.com
lcompton@us.ibm.com

Overview

This lab exercise illustrates a simple use case in CICS; using the CICS Explorer to develop a Hello World program coded in the Java programming language, deploy, and execute it in CICS TS for z/OS. This lab exercise illustrates a simple scenario—Java application program running in CICS TS using the JVMServer resource (which uses OSGi).

The OSGi support in CICS TS is required for application programs written in Java that will be running in a JVM server (i.e. non-Liberty). This is an overly simple Java program that illustrates the use of OSGi to package and manage Java applications in CICS.

We will test the OSGi Java program at a 3270 terminal. This was chosen for convenience; a Java program can be used any place that a COBOL application program can be invoked.

Lab Requirements

Please note that there are often several ways to perform functions in and for CICS. This lab exercise will present one of the ways. If you are familiar with CICS, you will notice that some of the statements are general, and not necessarily true for every situation.

This lab exercise assumes some knowledge of an Eclipse-based environment. You will also be using certain z/OS functions such as editing files, submitting jobs, and looking at job output. Additional some knowledge of CICS is preferred as we will be defining and installing CICS resources.

- **Data files:** This lab exercise assumes that you are using the VMware image that was prepared for this workshop and that the various artifacts discussed and used in this lab exercise are available in the VMware image.

- **z/OS Access:** Access to the class z/OS image is required, where some steps have been performed. It is assumed that you have been given a userid and password.

- **CICS Explorer:** This lab exercise assumes that you are using the latest CICS Explorer.

Lab Step Overview

Part 1: Start the CICS Explorer

This is a simple step the starts the CICS Explorer

Part 2: Java Development using the CICS Explorer

In this part of the lab exercise you will set the target environment (only once per development environment). You will create a Plug-in project and modify the deployment descriptor.

Part 3: Create a CICS Bundle Project

This part of the lab exercise will create a CICS Bundle project, which will contain the OSGi bundles from Part 2.

Part 4: Deploy the CICS Bundle to CICS TS

This part of the lab exercise will have you export the CICS Bundle project (and associated OSGi bundles) to z/OS and define a BUNDLE resource to CICS.

Part 5: Define CICS Resources

In this part of the lab exercise you will act as a CICS Systems Programmer and define the associated CICS resources. You will define and install a JVM Server, a BUNDLE, a PROGRAM, and a TRANSACTION.

Part 6: Test the Application

This part of the lab exercise lets you test the application on a 3270 device (again, the choice of a 3270 device was for simplicity, Java can be used as an application programming language most anywhere other application programming languages can be used).

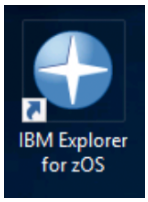
Part 7: Summary

This is a recap of the steps performed in this lab exercise.

Part 1: Start the IBM Explorers

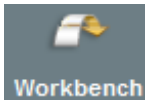
We have installed the CICS Explorer for you.

___**1.** From the **desktop**, **double-click** the **IBM Explorer** icon to start the **IBM Explorer for zOS** and **CICS Explorer.**.



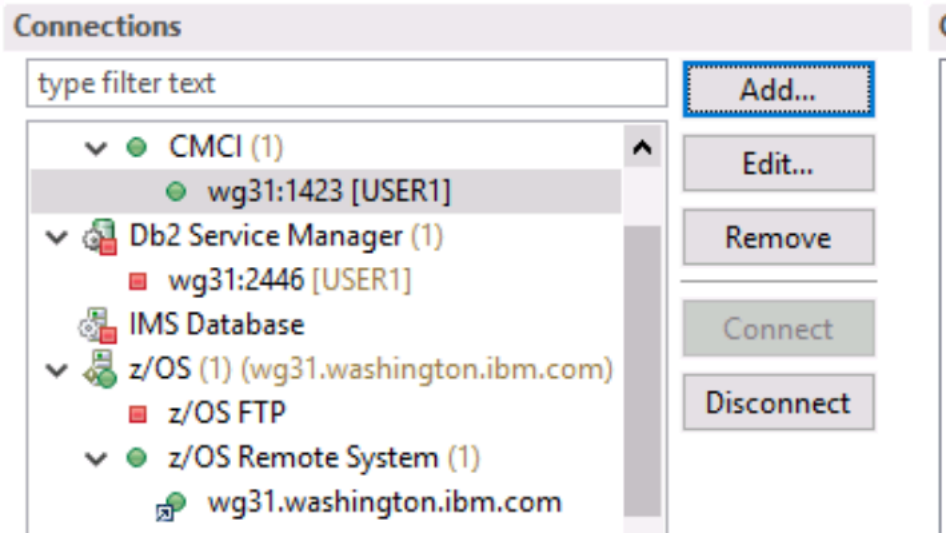
___**2.** **If** prompted, from the **Select a workspace** dialog, click the **OK** button.

___**3.** **If** the CICS Explorer shows you a **Welcome page** click the **Workbench icon** in the upper-right corner to go to the CICS Explorer. Then **maximize** the CICS Explorer window.



Verify that you have connections to the z/OS host system.

___**4.** Connections to the z/OS host system have been defined for you in the Host Connections view of the Remote System Explorer perspective. If either the CMCI and z/OS Remote System options are not green, highlight the option and click the **Connect** button. Both the **Remote System Explorer** and **CMCI** connections should be started and active**.**



Part 2: Develop the Java Program

Set the Target Environment (once per development environment)

These steps add a 'CICS TS V6.1' OSGi target environment to your workspace. When creating a project with a target environment of CICS TS V6.1, the CICS Explorer (Eclipse) environment will add the necessary support to your project to develop Java programs for CICS TS V6.1. The 'CICS TS V6.1' environment needs to be added only once per workspace.

___**1.** From the CICS Explorer **menu bar**, select **Window > Preferences**. On the left, expand **Plug-in Development** and click **Target Platform** (takes a moment to load on first entry).

___**2.** From the **Preferences** dialog, the **Target Platform** pane (on the right), click the **Add** button.

___**3.** From the **New Target Definition** dialog, press the **Template** radio button, and from the **Pull-down menu** next to Template, select **CICS TS V6.1 with Enterprise Java**. Then click the **Next** button.

___**4.** From the **Target Content** editor dialog (after it is done checking), click the **Finish** button.

___**5.** **Back** on the **Preferences** dialog, the **Target Platform** pane, **check** the box next to **CICS TS V6.1 with Enterprise Java** to set it as the default target environment. Then click the **Apply and Close** button (click OK on the Target Version dialog).

___**6.** You may receive a warning stating "You have selected a target with a newer version that your current Eclipse installation. This can cause unexpected behaviour in PDE. Please use a newer version of Eclipse." You can ignore this message by clicking the **OK** button.

Create a Plug-in Project

In this part of the lab exercise you will add a Java class to a Plug-in project. The Plug-in project corresponds to an OSGi bundle. This OSGi bundle will contain a Hello World program for the CICS environment.

___**7.** Switch to the **Plug-in Development** perspective.

___**8.** From the **CICS Explorer menu bar**, select **File > New > Plug-in Project**.

___**9.** From the **Plug-in Project dialog**, specify a **Project name** of **com.ibm.cics.simple.sample**, and in the **Target Platform** section, select an **OSGi framework** of **standard**, then click the **Next** button.

___**10.** On the **Content page** of the New Plug-in Project dialog, change the **Version** to **1.0.0** (remove the '.qualifier' we will not use it).

___**11.** **Still** on the **Content** page, specify an **Execution Environment** of **JavaSE-1.8**.

___**12.** **Still** on the **Content** page of the New Plug-in Project dialog, **uncheck** the box next to **Generate an activator**, then click the **Next** button.

> **Note** that we asked you to uncheck the box requesting an activator class be generated. We aren't going to use this activator class, however activator classes can be used if some action needs to take place when the OSGi environment activates your bundle. There are some constraint around the use of Activator classes in a CICS environment (for example they cannot contain JCICS API), so be sure to view the CICS TS V5.6 documentation on Activator classes.

___**13.** From the **Templates** page of the New Plug-in Project, **uncheck Create a plug-in using one of the templates** and press the **Finish** button (a manifest editor will open).

___**14.** From the **com.ibm.cics.simple.sample** manifest editor, select the **Dependencies** tab (across the bottom of the editor) and to the right of **Imported Packages**, click the **Add** button.

___**15.** Select **com.ibm.cics.server** and click **Add** (this allows your program to access the JCICS classes).

___**16.** **Still** in the manifest editor, **click** on the **MANIFEST.MF** tab, and in the manifest, **add the following line** into the editor (**after Bundle-Version**).

```
CICS-MainClass: com.ibm.cics.simple.sample.HelloCICSWorld
```

___**17.** **Save** and close the **manifest editor.**

___**18.** In your com.ibm.cics.simple.sample **project**, **right-click** on **src > New > Package** and add a **Package** with a name of **com.ibm.cics.simple.sample**. Click the **Finish** button.

___**19.** From the **Project Explorer** view, **right-click** on your **package** and add a **New > Class** named **HelloCICSWorld**. Click the **Finish** button.

___**20.** **Replace** the contents with the following:

```
package com.ibm.cics.simple.sample;
import com.ibm.cics.server.CommAreaHolder;
import com.ibm.cics.server.Task;
public class HelloCICSWorld {
    public static void main(CommAreaHolder CAH) {
        Task t = Task.getTask();
        if ( t == null )
            System.err.println
                 ("HelloCICSWorld example: Can't get Task");
        else
            t.out.println("Hello from a Java CICS application");
    }
}
```

___**21. Note** that there should be no error messages. If there are any errors, contact a lab instructor.

___**22. Save** and **close** your HelloCICSWorld.java editor.

Note: we are relying on the default behavior of Eclipse to compile our program when we save the file. If you turn off the default behavior, you have to specifically ask for your program to be compiled. Most IDEs have the default set to compile your program when you save (so you are never told to compile your program, only to save your program). The program is compiled into a .class file. Although the .class file is not visible (by default) in the Project Explorer view, if you displayed the files in your project from Windows, you can see the .class file. Also, you can change the Eclipse default and have it display your .class file (but very very few people do that).

___**23. Collapse** your com.ibm.cics.simple.sample project.

Part 3: Create a CICS Bundle Project

In this part of the lab exercise, you will create a CICS Bundle project on your workstation, you will add the OSGi bundle (plug-in project) to the CICS Bundle. In the next part of the lab, you will then export the CICS Bundle from your workstation to Unix System Services (z/OS), then define and install a CICS BUNDLE definition in CICS that points to the CICS Bundle on Unix System Services.

A CICS Bundle project contains a cics.xml manifest file and one or more XML resource files. We will use wizards to add information about your OSGi bundles to the CICS BUNDLE project.

___**1.** From the CICS Explorer, the **CICS Explorer** menu bar, select **File > New > Project.**

___**2.** From the **New project** dialog, **select CICS Resources > CICS Bundle Project** and click the **Next** button.

___**3.** From the **CICS Bundle Project** dialog**,** provide a **Project name** of **com.ibm.cics.simple.sampleBUNDLE**, and click the **Finish** button.

NOTE that your CICS BUNDLE manifest is opened in a manifest editor.

Add the OSGi bundle to your CICS BUNDLE Project

___**4.** From the **CICS manifest editor**, in the **Defined Resources** section, **click** the **New** button.

___**5.** **From** the **drop-down** menu, select **OSGi Bundle Project Include.**

___**6.** In the **OSGi Bundle Project Include** dialog, **select** the **com.ibm.cics.simple.sample** project, enter a **JVM Server** name of **JVMSRV01**, and click the **Finish** button.

Add a Transaction Definition to your CICS BUNDLE Project

___**7.** From the **CICS manifest editor**, in the **Defined Resources** section, **click** the **New** button.

___**8.** **From** the **drop-down** menu, select **Transaction Definition** and supply the following values:

Attribute	Value
Name	HELO
Description	Hello World Java Transaction
Program Name	HELOWRL1

___**9.** Remove the check from **Open** Editor, then click **Finish t**o complete the steps to add the Transaction Definition.

Add a Program Definition to your CICS BUNDLE Project

___**10.** From the **CICS manifest editor**, in the **Defined Resources** section, **click** the **New** button.

___**11. From** the **drop-down** menu, select **Program Definition** and supply the following values:

Attribute	Value
Name	HELOWRL1
Description	Hello World Java Program
Program Type	check the **Java** radio button
Service Name	com.ibm.cics.simple.sample.HelloCICSWorld
JVMServer	JVMSRV01

___**12.** Complete the steps to add the Program Definition.

___**13. Close** the **Manifest Editor.**

Part 4: Deploy the Bundle to z/OS

In this part of the lab exercise you will export your CICS BUNDLE to the z/OS UNIX System Service File System. When you export your CICS BUNDLE, all of the OSGi bundles referenced by your CICS BUNDLE will also be exported.

You will then create a BUNDLE resource definition in CICS that will reference the bundle directory on zFS. When the BUNDLE resource is installed, the OSGi bundles and the code they contain will be installed into the indicated JVM server.

To copy the CICS Bundle information and the associated OSGi bundles to z/OS, we will use the 'Export Bundle Project to z/OS UNIX File System'. We will then create a BUNDLE resource in CICS using the CICS Explorer.

Export the CICS Bundle Project

___**14.** From the **CICS Explorer**, the **Plug-in Development** perspective, **Project Explorer** view, **right-click** on the **com.ibm.cics.simple.sampleBUNDLE** CICS bundle project and from the context menu, select **Export Bundle Project to z/OS UNIX File System**.

___**15.** From the **Export Bundle** pop-up, **click** the **Export to a specific location in the file system** and **click** the **Next** button.

___**16.** If you have a connection, but not signed on, select the **drop-down** to the right of **Connection** and choose your z/OS Connection to sign on.

___**17. From** the **Export Bundle** dialog enter the information below, then press the **Finish** button.

Note: If you are redeploying your Bundle, check the box for **Clear existing contents of Bundle directory**.

Field	Value
Bundle project:	com.ibm.cics.simple.sampleBUNDLE
Parent Directory:	/u/user1/cicslab/bundles
Bundle Directory:	/u/user1/cicslab/bundles/com.ibm.cics.simple.sampleBUNDLE_1.0.0
Clear existing contents of Bundle directory	Checked

Caution: be sure the parent directory and bundle directory start with your home directory (/u/user1).

Part 5: Define Resources to CICS

This section defines resources to CICS.

Define and Install a JVMServer

A JVMServer is needed to run your Java programs. You may already have a JVM server with a name of JVMSRV01. **If** you don't already have a JVM server named **JVMSRV01**, you will need to define and install one as follows.

___**1.** Switch to the **Remote Systems Explorer** perspective, the **Remote Systems** view. **Right-click** on **Local > Local Files > Drives > C:\CICSLAB\Java\ JVMProfiles\DDWOSGI** and select **copy** from the popup menu.

___**2.** From the **CICS Explorer**, the **Remote System Explorer** perspective, the **Remote Systems** view, **right-click** on **MyzOS > z/OS UNIX Files > My Home > cicslab > JVMProfiles** and from the context menu select **Paste**.

NOTE: The JVMProfile needs to be DDWOSGI.jvmprofile, not DDWOSGI. The .jvmprofile extension is a required.

___**3.** **Right-click** on **DDWOSGI** (the file you just pasted in) and select **Rename** from the popup dialog. Enter a **New name** of **DDWOSGI.jvmprofile** and click **OK**.

___**4.** Double-click on **DDWOSGI.jvmprofile** to open it in the editor. Ensure that it looks okay (otherwise contact a lab instructor). Close DDWOSGI.jvmprofile.

___**5.** Switch to the **CICS SM** perspective and define a **JVM server** resource definition to run the Java programs. **Define** a JVMserver with the following attributes.

Attribute	Value
Group	WORKSHOP
JVMServer	JVMSRV01
Jvmprofile	DDWOSGI

___**6.** From the **JVMSRV01 (JVM Server Definition)** view, use the drop-down menu to **Install** your JVMserver

___**7.** From the **Perform INSTALL Operation** dialog, click the **OK** button. If the installation of your JVM server resource was not successful, contact a lab instructor.

Define and Install the CICS BUNDLE resource definition

___**8.** From the **CICS SM** perspective, select **Definitions > Bundle Definitions**.

___**9.** From the **Bundle Definitions** view, **right-click** in an **open area** and from the context menu, select **New**.

___**10.** From the **Create Bundle Definition** dialog, **specify the following**, then click the **Finish** button.

Attribute	Value
Resource/CSD Group	WORKSHOP
Name	OSGIBUN1
Description	CICS Bundle for Simple Java Program
BUndledir	/u/user1/cicslab/bundles/com.ibm.cics.simple.sampleBUNDLE_1.0.0/

	Note: Use the Browse button to select your Bundle

___**11.** From the **OSGIBUN1 (Bundle Definition)**, use the drop-down menu to **Install** your **OSGIBUN1** bundle definition.

___**12.** From the **Perform INSTALL Operation** dialog, click the **OK** button. If the installation of your BUNDLE resource was not successful, contact a lab instructor.

___**13.** Close your **OSGIBUN1 (Bundle Definition)** view.

View the OSGi Bundles in your CICS region

___**14.** From the **CICS SM** perspective, from the **menu bar**, select **Operations > Java > OSGi Bundles**. Verify that **your** OSGi bundle exist, has been installed properly, and is in the ACTIVE state. If you don't see your OSGi bundle or if it is not all in the ACTIVE state, contact a lab instructor.

View the OSGi Service in your CICS region

___**15.** From the **CICS SM** perspective, from the **menu bar**, select **Operations > Java > OSGi Services**. Verify that you have an OSGi services named com.ibm.cics.simple.sample.HelloCICSWorld, and that it is in the ACTIVE state. If you don't see your OSGi service or it is not ACTIVE, contact a lab instructor.

___**16.** You refer to these services when you define PROGRAM definitions. You added the program definition to the CICS BUNDLE, so it was dynamically defined for you. You can select **Operations > Programs** to see your **HELOWRL1** program. Set a **Quick Filter** if you like (right-click a program name and select **Add Quick Filter**).

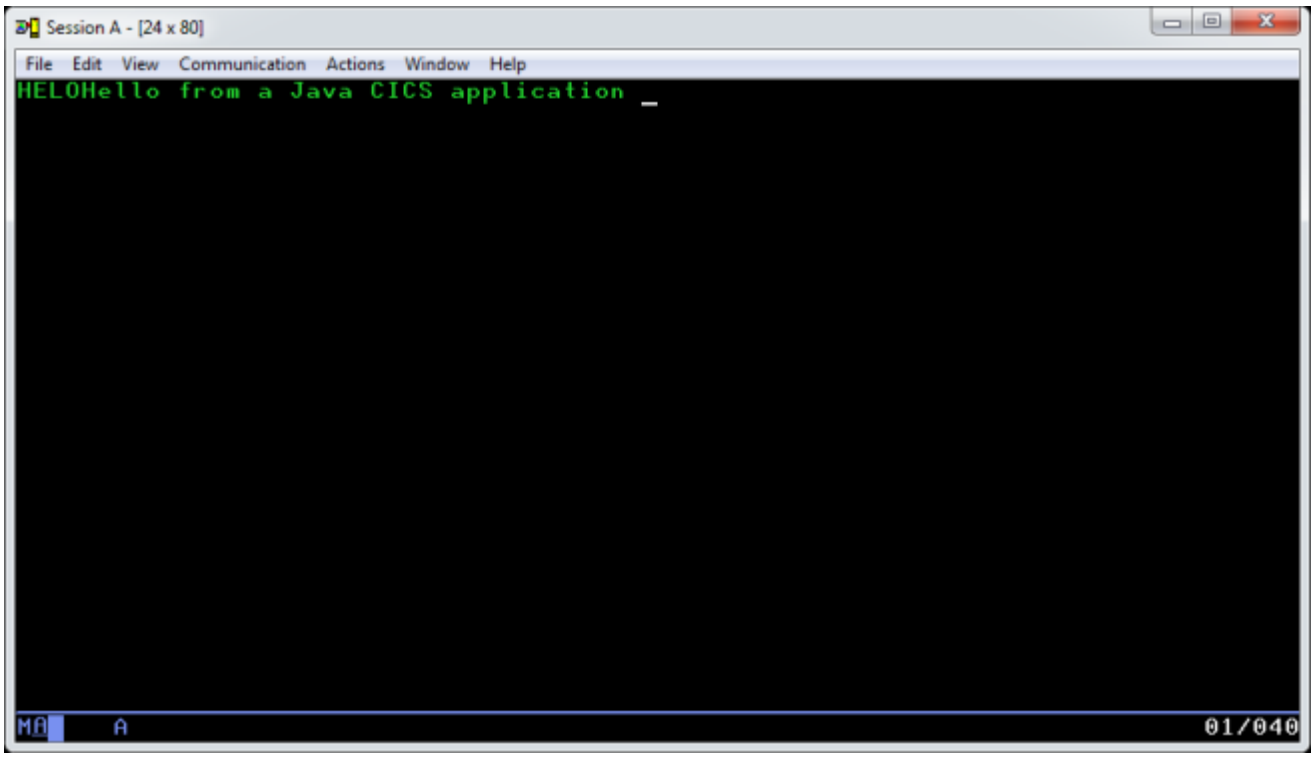
Part 6: Test the Application Program

We are done installing the application so the next step is to verify that all of our application artifacts are in the right place and that all of our CICS definitions are correct. We will be using a 3270 device to test your work.

___**1.** **Start** a 3270 session with your CICS region (from the VTAM MSG 10, enter **CICS** and press **enter**. Then clear your screen).

___**2.** Type in a tranid of **HELO**.

___**3.** Your terminal should look as follows…



___**4.** If you have any question, contact a lab instructor.

Part 7: Summary

Congratulations, you have just built a CICS Java application using the OSGi support in CICS.

In this lab you:

- Started the CICS Explorer.
- You wrote a simple HelloWorld program that uses some of the CICS API (you copied it)
- You packaged Java code as an OSGi bundle in a CICS BUNDLE, along with CICS resource definitions (a Program and a Transaction)
- You deployed the CICS BUNDLE into CICS
- You defined a JVMServer definition (a JVM) for the application
- You tested your application

If you have any questions about anything you did in this lab exercise, please contact one of the lab instructors.