Practical Application Development with the CICS JVM Server

This tutorial practically shows the installation and configuration of the CICS Explorer as well as an example for the development of Java applications for the JVM Server.

The CICS Explorer is an interface for the local Eclipse IDE that is used for the management of CICS JVM Servers. It enables to communicate with the USS file system and with the CICS region.

In order to create a Java application for the JVM Server the following tasks have to be carried out (excluding the configurational tasks). Note that the basic procedure for this tutorial has been adapted from Transaction Server for z/OS Version 4 Release 2 - Java Applications in CICS.

- 1. Create a **Plug-in Project** in Eclipse that represents the OSGi bundle
- 2. Create a CICS bundle
- 3. Include the OSGi bundle into the CICS bundle and assign the JVM Server name needed for its execution
- 4. Export the CICS bundle into the USS file system
- 5. Create the following definitions in the CICS SM Perspective
 - (a) JVM Server
 - (b) Bundle
 - (c) Program
 - (d) Transaction
- 6. Install all definitions into the CICS region

A step by step procedure for all tasks is provided within this tutorial.



The interaction of OSGi bundles, the JVM Server, CICS programs and transactions is illustrated in the following

Install the CICS Explorer SDK into Eclipse

Requirements

- Eclipse IDE v. 3.7 (and higher).
- CICS Explorer SDK (download from http://www-01.ibm.com/software/ htp/cics/explorer/).
- **1.** Start your Eclipse IDE.
- 2. Go to Window \rightarrow Install New Software....

e SDK							
Refactor	Navigate	Search	Project	Run	Window	Help	
3 ↓ ☆	- () -	Q	# G	•	👝 🔗	. 🚳 Welcome	
at X						Image: Contents Image: Contents	
						Install New Software	
						Hood Lappo Dok	

3. Click on Add... in the Install panel.

🗢 Install	
Available Software	
Select a site or enter the location of a site.	
Work with: ⁰ type or select a site	✓ <u>A</u> dd
Find more software by working with the "Available Soft	ware Sites" preferences.

4. Name the repository, click on Archive... and browse to the CICS Explorer SDK zip file.

👄 Add Repository 🛛 🔀				
<u>N</u> ame:	CICS Explorer SDK	Local		
Location:	jar:file: <path sdk="" to=""></path>	Archive		
?		OK Cancel		

5. Select all options in the Install panel and click Next.

🥏 Install		
Available : Check the it	Software ems that you wish to install.	8 -
Work with:	CICS Explorer SDK - jar:file: <path 5<br="" to="">Find more</path>	DK> Add software by working with the <u>"Available Software Sites"</u> preferences.
type filter te	xt	
Name		Version
😑 🔽 💷 I	BM CICS Explorer SDK	
	🖗 IBM CICS Explorer SDK	1.1.0.0_201106161553
	IBM CICS SDK for Java	1.1.0.0_201106161553
Select All	Deselect All 2 items selected	1

6. Accept the license agreement and finish the installation.

After restarting Eclipse several new Perspectives will be available. The ones important for this tutorial are the CICS SM and the z/OS.

Configure required Connections

In order to run applications the connections to the CICS Management Interface and the z/OS interface need to be configured.

1. Go to Window \rightarrow Preferences.



2. In the **Preferences** panel expand the **CICS Explorer**menu, click on the **Credentials** node and click on **New**.

3. Name the credentials and enter your username and password and click on **Apply**. Keep the **Preferences** panel open.



4. Click on the **Connections** node and create a new **CICS Management Interface** connection using the previously defined credentials. The default port for the CICS Management Interface is **1490**. Click on **Apply** and click on **Connect**. Keep the **Preferences** panel open.

	_ 🗆 🖂
	New Connect Delete
Restore <u>D</u> efaults	Apply
	Restore Defaults

5. Choose **System z - FTP** as connection type and click on **New** to create a new FTP connection using the previously defined credentials. The default port for the **System z - FTP** is **21**. Apply the changes, click on **Connect** and click on **OK** to close the **Preferences** panel.

Preferences		_ 🗆 🛛
	Connections Connection Type: System z - FTP Name: My zFS Location Host name: Port number: 21	Lever Connect
	Authentication Credentials W Credentials FTP Additional Details Transfer mode O Passive O Active	
	Attempt secure connection Enable MBCS support Restore Default	ts Apply
?	ОК	Cancel

After successfully connecting to the System Z - FTP, you will be able to see all your z/OS and USS files within the \mathbf{z}/\mathbf{OS} Perspective.



n Data Sets 🧣 z/OS UNIX Files 🛛	÷ ¬ ¬ ¬ ¬
Path: /	(23) 🖄
⊕- 🕞 \$SYSNAME [\$SYSNAME/]	
sversion [\$version/]	
😟 🗀 ADCD	
😟 🕞 bin [\$VERSION/bin]	
CICS1.DFH\$JV2.dfhjvmtrc	
📄 CICS1.USJVMSRV.dfhjvmtrc	
🕀 🗀 CICS1	
employee com.ibm.cics.server.examples	
⊕∽ 🕞 dev [\$SYSNAME/dev]	
dfhjvmerr	
📄 🧾 dfhjvmin	
dfhjvmout	
tere [\$SYSNAME/etc]	
■ Ib [\$VERSION/Ib]	
Umung opt [\$VERSION/opt]	
Umunities samples [\$VERSION/samples]	
Umunities the states of the st	
ter [\$vERSION/usr]	
ternung var [\$SYSNAME/var]	
210	

Configure the Target Platform

In order to develop applications for the CICS JVM Server, the target platform needs to be configured for the CICS TS 4.2 environment.

1. Go to Window \rightarrow Preferences.

2. In the **Preferences** panel expand the **Plug-in Development** menu, click on the **Target Platform** node and click on **Add**.



3. Choose CICS TS 4.2 Runtime as the template in the new window and click Next.



- 4. Review if the locations
 - $\bullet\ com.ibm.cics.server.runtime42.feature$ and
 - org.eclipse.rcp

are included in the ${\bf Locations}$ tab of the ${\bf Target}\ {\bf Content}$ panel and click on ${\bf Finish}.$

New Target Definition	_ 🗆 🗙
Target Content Edit the name, description, and plug-ins contained in a target.	
Name: CICS TS V4.2 Runtime	
Locations Content Environment Arguments Implicit Dependencies	
The following list of locations will be used to collect plug-ins for this target de	finition.
& com.ibm.cics.server.runtime42.feature - \${aclipse_home} 2 plug- org.eclipse.rcp - \${eclipse_home} 31 plug-ins available	<u>A</u> dd
	<u>R</u> emove
	Update
Show location content	
(?)	Cancel

5. Choose CICS TS V4.2 Runtime as the standard target platform and klick OK.



Create a HelloWorld Application

Since Eclipse is based on the OSGi Equinox Framework, OSGi bundles in Eclipse are designated as **Plug-ins**.

- 1. Open the **Plug-in Development** Perspective.
- 2. Create new Plug-in Project.



3. Enter the name of the project, choose **standard** as the target platform and click **Next**.

∋ New Plug-in Project	_ 🗆 🛛			
Plug-in Project Create a new plug-in project				
Project name: HelloWorld				
Use default location				
Location: C:\Dokumente und Einstellungen\robert\workspace\HelloWorld	Browse			
Project Settings				
Source folder: src				
Output folder: bin				
Target Platform This plug-in is targeted to run with: O Eclipse version: 3.7				
⊙ an OSGi framework: standard ▼				

4. Remove the *qualifier*, uncheck the option for the generation of an activator and click **Next**.

🛢 New Plug-in Project 📃 🗆 🔀				
Content Enter the data required to generate the plug-in.				
Properties				
ID:	HelloWorld			
Version:	1.0.0,qualifier			
N <u>a</u> me:	HelloWorld			
Provi <u>d</u> er:				
Execution Environme	nt: JavaSE-1.6	Environments		
- Options				
Generate an activator, a Java class that controls the plug-in's life cycle				
Ac <u>t</u> ivator: hello	world.Activator			

5. Create a package helloworldpackage and a HelloWorld class in the previously created project.



6. Add the following code into the HelloWorld class (adapted from the Hello CICS World CICS Explorer Example).

```
package helloworldpackage;
import com.ibm.cics.server.*;
public class HelloWorld {
    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(" transaction started");
                t.out.println("Hello from a Java CICS application");
     }
}
```

7. Open the MANIFEST.MF and add

Import-Package: com.ibm.cics.server

to enable the package import carried out in the HelloWorld class and

CICS-MainClass: helloworldpackage.HelloWorld; alias=helloworld

to identify the bundle and its package holding the main class. Press the key combination $\langle \mathbf{CTRL} \rangle + \mathbf{S}$ on your keyboard to save your changes. Note that the alias=helloworld statement assigns an optional alias for the main class. Also note that the last statement of the header file should include a line break (in this case a line break should appear in the CICS-MainClass line).

l Package Explorer 🛛 🍣 Plug-ins 🛛 🤹 🏱 🗖 🗖	Helloworld 🔀
😑 😂 HelloWorld	Manifest-Version: 1.0
IRE System Library [JavaSE-1.6]	Bundle-ManifestVersion: 2
🖲 🔿 Plug-in Dependencies	Bundle-Name: HelloWorld
🖨 进 src	Bundle-SymbolicName: HelloWorld
😑 🌐 helloworldpackage	Bundle-Version: 1.0.0
🖲 🚺 HelloWorld. java	Bundle-RequiredExecutionEnvironment: JavaSE-1.6
META-INF	Import-Package: com.ibm.cics.server
MANIFEST.MF	CICS-MainClass: helloworldpackage.HelloWorld; alias=helloworld
and build.properties	
	Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF build.properties

8. Go to File \rightarrow New \rightarrow Other.. \rightarrow CICS Resources, choose CICS Bundle project and click Next to create a CICS Bundle.



9. Name the CICS bundle and click **Finish**.

😂 Bundle Proje	ect	_ 🗆 🔀		
CICS Bundle Pr Create a new proj	oject ect containing the files for deployment in a CICS Bundle			
Project name: h	elloworldcics			
Use gehault location Location: C:\Dokumente und Einstellungen\robert\workspace\helloworldcics Browse				
?	<back next=""> Finish</back>	Cancel		

Package Explorer	🛛 💲 Plug-ins 🖉 🍇		
⊕ 🗁 HelloWorld ⊡ 📂 📂 helloworldcics			
🗄 🗁 META-INF	New		Plug-in Project
🛶 🔂 🖓 🖓	Go Into		🎼 Feature Project
	Open in New Window		📬 Project
	Show In	Alt+Shift+W	Component Definition
	📄 Сору	Ctrl+C	Product Configuration
	🗎 Copy Qualified Name		📸 Target Definition
	💼 Paste	Ctrl+V	🖶 Package
	💢 Delete	Delete	🞯 Class
	Build Path		🕨 🗊 Interface
	Refactor	Alt+Shift+T	Source Folder
			🗌 😂 Folder
	Import		Example
	Logurust a		
	. III 2/OS UNIX File System		Ctrl+N Ctrl+N
	🔆 Defead		

10. Perform a right-click on the previously created CICS bundle and go to $\mathbf{New} \to \mathbf{Other...}$

11. Choose CICS Resources \rightarrow Include OSGi Project in Bundle and click Next.

la New	
Select a wizard	
Wizards: type filter text	
CICS Bundle project CICS Event Binding CICS Event Processing Adapter CICS Event Processing Adapter	

12. Choose the **HelloWorld** Plug-in project (OSGi bundle), enter the name of the JVM Server your application will use and click **Finish**. Note that a definition of a JVM Sever will be described step 18.

🔿 Include OSGi Project in Bundle	_ 🗆 🔀
Include OSGi Project in Bundle Choose Project containing Java code to be included in the CICS bundle	⊡ ∌
Select the project to be included in the bundle	
[™] HelloWorld	
OSGi Bundle directive:	
Symbolic Name: HelloWorld	
Version: 1.0.0	
JVM Server: MYJVMSRV (required)	

13. Click right on the previously created CICS bundle and go to z/OS UNIX File System to export the CICS Bundle into the USS file system.

	en pararproporcios	
÷۰	🖄 helloworldcics	
	New	+
	Go Into	
	Open in New Window	
	Show In	Alt+Shift+W 🕨
	📄 Сору	Ctrl+C
	音 Copy Qualified Name	
	💼 Paste	Ctrl+V
	💢 Delete	Delete
	Build Path	•
	Refactor	Alt+Shift+T 🕨
	🚵 Import	
	🛃 Export	
	🚛 🗓 z/OS UNIX File System	

14. Choose the export directory and click Finish.

Export to z/G	OS UNIX File System	_ 🗆 🗙
Export Bundle		
Select bundle proj	ject to export as well as its destination.	
Bundle project:	helloworldcics	Browse
Connection:	● ▼ My System z FTP	
Parent Directory:	/u/ <username></username>	ß
Bundle Directory:	/u/ <username>/helloworldcics</username>	
	Options	
?	Einish	Cancel

15. Open the CICS SM Perspective.

16. Click on the **CICSPlex Repositories** view, expand your CICS repository and select your CICS region



17. Since CICS requires a definition for each resource, it is mandatory to define a JVM Server, an OSGi bundle a program and a transaction. Go to **Definitions** \rightarrow **Bundle Definitions** / **JVM Server Definitions** /

Program Definitions / **Transaction Definitions** to open all views needed for the definitions.

Definitions	Run	Window	Help	
👫 Atom Service Definitions				
🚯 Bundle Definitions				
A CALLER AND DECEMBER				

18. Activate the **JVM Server Definitions** view and perform a right-click \rightarrow **New...** to create a new JVM Server definition. Skip this step if you want to use an existing JVM Server definition.

Bundle Definitions	🚜 JVM Server Definitions 🛛 📑 Program (
CNX0211I Context: IB	MMISC. Resource: JVMSVDEF. 0 records collected		
Name	Version		
	New		

19. Assign the group (MYGROUP), define a unique name (MYJVMSRV) and assign the JVM profile (DFHJVMAX) to the JVM Server and click **Finish**.

🗢 New JVM Server Definition 📃 🗖 🔀			
Create JVM Server Defir	nition		
Data Repository: ✓ Region (CSD) Resource Group:	CICS1 CICS1 MYGROUP		
Name:	MYJVMSRV		
Description:	My own JVM Server		
Enabled Status:	Enabled 💌		
LE Runtime Options Program:	DFHAXRO		
JVM Profile:	DFHJVMAX		
Open editor			
?	<u>Einish</u> Cancel		

Description of options:

- **Resource Group:** Resources in CICS are grouped. Therefore, one needs to assign a group to the JVM Server. If the entered group is not defined, it will be created automatically.
- Name: The name is a unique identifier for the JVM Server. An OSGi bundle requires to be assigned to the JVM Server (refer step 12).
- LE Runtime Option Program: Assigns the program that defines the options for Language Environment. By default set to DHFAXRO. Do not alter this field.
- JVM Profile: The profile defines necessary parameters such as the working directory or optional startup parameters for the JVM. By default profiles are located in the USS file system directory: /usr/lpp/cicsts/cicsts42/JVMProfiles

To get a user-friendlier look on your definitions, find your resource group and click in it. With it being activated, only your personal definitions will be displayed.



20. Perform a right-click \rightarrow **Install...** on the new created JVM Server definition.

Name	Version	Description	Change Time
MYJVMSRV	0	My own JVM Server	24.03.2012 00:15:24
		New Open Install.,	
		X Delete	Delete Ctrl+C

21. Choose the CICS region for the installation and click OK.



22. Go to $\mathbf{Operations} \to \mathbf{Java} \to \mathbf{JVM}$ Servers to open the \mathbf{JVM} Server view.



23. Activate the **JVM Servers** view, refresh the view and verify if the JVM Server installation was successful (Enable Status = ENABLED).

Bundle Definitions	🖓 JVM Server Definitions 🔳	Program Definitions 📕 JVM Servers 💈	×
CNX0211I Context: CI	CS1. Resource: JVMSERV. 1 reco	ords collected at 24.0 <mark>3.2012 21:04:13</mark>	
Region	Name	Enable Status	
CICS1	MYJVMSRV	ENABLED	

24. Activate the **Bundle Definitions** view and perform a right-click \rightarrow **New...** to create a new bundle definition.

25. Assign the group (MYGROUP), define a unique name (hello) and the directory of the bundle chosen in step 14 and click on **Finish**.

😂 New Bundle Defi	nition 📃 🗖 🔀
Create Bundle Defi	nition
Data Repository:	CICS1 CICS1 MYGROUP
Name: Description:	hello Hello World Bundle
Bundle Directory:	/u/ <username>/helloworldcics</username>
Open editor	
?	<u>Finish</u> Cancel

26. Perform a right-click \rightarrow **Install...** on the new created OSGi bundle definition. Choose the CICS region for the installation and click **OK**.

27. Go to **Operations** \rightarrow **Java** \rightarrow **OSGi Bundles** to open the **OSGi Bundles** view. Activate the **OSGi Bundles** view, refresh the view and verify if the OSGi bundle installation was successful (State = Active).

am Definitions 🕻	📕 JVM Servers	🔄 Transaction Defini	tions 🔂 OSGi Bundles 🛛
ollected at 24.0	3.2012 20:47:50		୍ଦ୍ୱର 👯 + ଅ
Version		State	Bundle
1.0.0		 ACTIVE 	hello

28. Activate the **Program Definitions** view and perform a right-click \rightarrow **New...** to create a new program definition.

29. Assign the group (MYGROUP), define a unique name (HELLOW) to and click on **Finish**. A new view opens that enables to specify parameters for the program.

🔿 New Program Definition 📃 🗖 🔀			
Create Program De	efinition		
Data Repository: ✓ Region (CSD) Resource Group:	CICS1 CICS1 MYGROUP		
Name:	HELLOW		
Description:	My Hello World Program		
♥ Open editor			
?		Einish Cancel	

- **30.** Activate the options for
 - This program is threadsafe
 → Since the JVM Server is a multi application environment, all application
 are required to be threadsafe.
 - This program is CPU intensive...
 → Enables higher priority scheduling.
 - This program only runs on versions of CICS later that 4.1... \rightarrow Since the JVM Server is introduced for CICS version 4.2, this option is mandatory.
 - Program can write to CICS-key storage \rightarrow Since the JVM Server uses a T8 TCB and since the storage protection key 8 is assigned to CICS, the program needs to be able to write to the CICS-key storage.

🎇 Bundle Definition (h 🛛 🖻 *Program Definition 🛛 💙			
Program Definition (HELLOW) Hello World CICS			
💠 CICS1 🕨 🍘 CICS1 🕨 📷 HELLOW			
🖬 Overview			
Basic			
Name: Description: Hello World CICS			
CSD Group: Created:			
Enabled Changed:			
Details			
Language: N/A 🔽			
Display Execution Diagnostic Facility (EDF) screens			
Open Transaction Environment			
Programs that are coded to threadsafe standards can exploit t environment (OTE). OTE provides the opportunity for perform- toleration of APIs that are not normally permitted in a CICS ap			
This Program is threadsafe			
This Program is CPU intensive, which might include threat ✓ issues few non-threadsafe CICS commands, or uses AP: permitted by CICS			
This Program * is only run on versions of CICS later than v4.1 * issues many external resource manager calls * only uses APIs that are always permitted by CICS			
Storage			
Can handle 31 bit addresses (above the 16MB line)			
Use Program from the Link Pack Area (LPA)			
Program can write to CICS-key storage			
Program reuse			
💿 Reuse if possible			
O Force reuse			
Always load a new copy (requires explicit FREEMAIN)			
Load a new copy whenever use count drops to zero			
Overview Remote Java™ Attributes			

31. Activate the **Java** tab, check the option **Operate program under**..., insert the alias defined in step 7 (or the string helloworldpackage.HelloWorld), assign the JVM Server to the program and press the key combination **CTRL**+**S** on your keyboard to save your changes.

🏭 Bundle Definition (h 🛛 💼 *Program Definition 🙁 🏾 >>>
Program Definition (HELLOW) Hello World CICS
🍄 CICS1 → 🍓 CICS1 → 📷 HELLOW
⊡ Java™
Java Virtual Machine (JVM)
Operate program under control of a JVM
Fully qualified main Java class name to be run
helloworld
Environment
The Java Program can run in a JVM Server, or in a JVM Pool with options sp Profile.
OUse a JVM Server: MYJVMSRV
O Use the default JVM Profile (DFHJVMPR)
O Use a named JVM Profile:
Overview Remote Java™ Attributes

32. Activate the **Program Definitions** view and perform a right-click \rightarrow **Install...** on the new created program definition. Choose the CICS region for the installation and click **OK**.

33. Activate the **Transaction Definitions** view and perform a right-click \rightarrow **New...** to create a new transaction definition.

34. Assign the group (MYGROUP), define a unique name (HWC), assign the previously defined program (HELLOW) and click **Finish**.

🗢 New Transaction Definition 📃 🗖 🔀				
Create Transaction	Definition			5
Data Repository: ✓ Region (CSD) Resource Group:	CICS1 CICS1 MYGROUP			
Name: Description:	HWC Hello World CICS	 		
Program Name: Remote System Name:	HELLOW			
Open editor				
?			Einish	Cancel

35. Activate the **Transaction Definitions** view and perform a right-click \rightarrow **Install...** on the new created program definition. Choose the CICS region for the installation and click **OK**.

36. Connect to CICS using a *3270* emulator and execute the previously installed transaction. The output should show the text

HWC transaction started Hello World from CICS.

File	Options		
HWC tr	ansaction started		
Hello	World from CICS 📕		