Workshop- CICS Liberty profile with remote development feature & CICS automation build with Jenkins

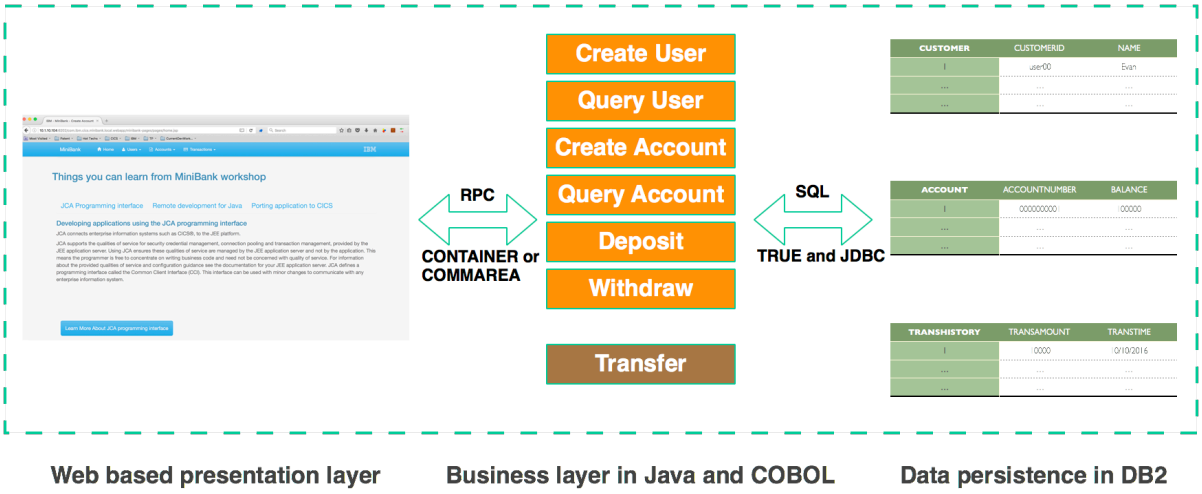
2016 CICS AP Tech Savvy

Evan (Zhou Bei Chun) ***beichunz@cn.ibm.com***
and other CICS CDL members
10/18/2016

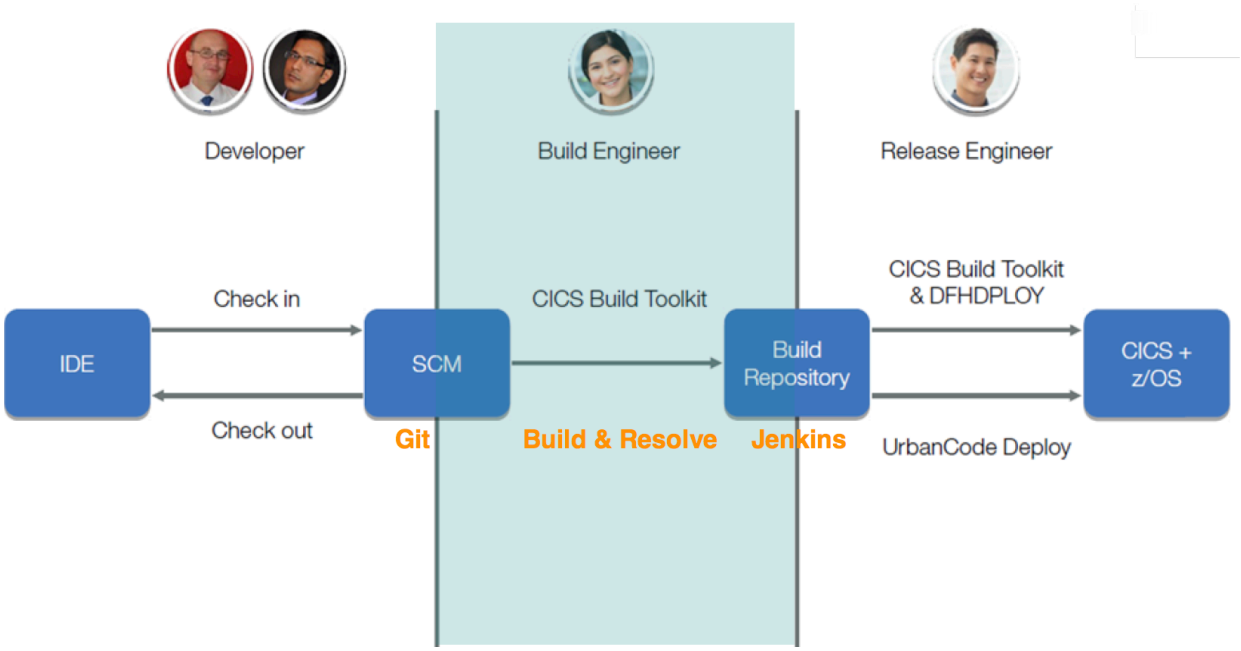
Purpose of the workshop:

In this Workshop, we have two parts.

The first part, we show how to develop a web application in CICS Liberty to access service from critical business logic written in COBOL and OSGI Java and running in CICS AORs. By using CICS remote development feature, you can develop your JEE7 based web applications remotely, aka, using a distributed Websphere Liberty Profile in your laptop, and then port the applications into CICS Liberty without any change.
We will use a demo application called Minibank which can be accessed in [cicsdev git repository].



Then the second part show how to use CICS Build Toolkit with ***Git*** as Source Control Manager and ***Jenkins*** as build server for build automation. We provide a sample script using CICS Build Toolkit to execute inside a Jenkins server to build bundles based on the source from GitHub and then to resolve the variables inside code with your own properties file.



Lab environment prepared for you (replace the ** to your team ID number):

***WOR region**:*	*CICS**W1*
***AOR region**:*	*CICS**A1*
WUI TCPIP port:	*6**0*
WUI CMCI port:	*6**1*
Liberty HTTP port:	*6**2*
***Liberty HTTPS port**:*	*6**3*

At the end of this document, we provide appendixes. We have completed the steps in Appendix A and Appendix C before the workshop, and **you need to configure FPT connection and CMCI connection based Appendix B.**

Appendix A shows how to establish development tool (Eclipse and CICS explore SDK) for CICS Java development. You can refer to appendix A to setup Eclipse for Liberty and Java development.

Appendix B shows how to set up connection between CICS Explorer and MVS, including:

- *z/OS FTP connection*
- *CMCI connection*

Appendix C shows how to set up WebSphere Application Server Liberty and CICS Remote Development Feature for Java in your laptop eclipse.

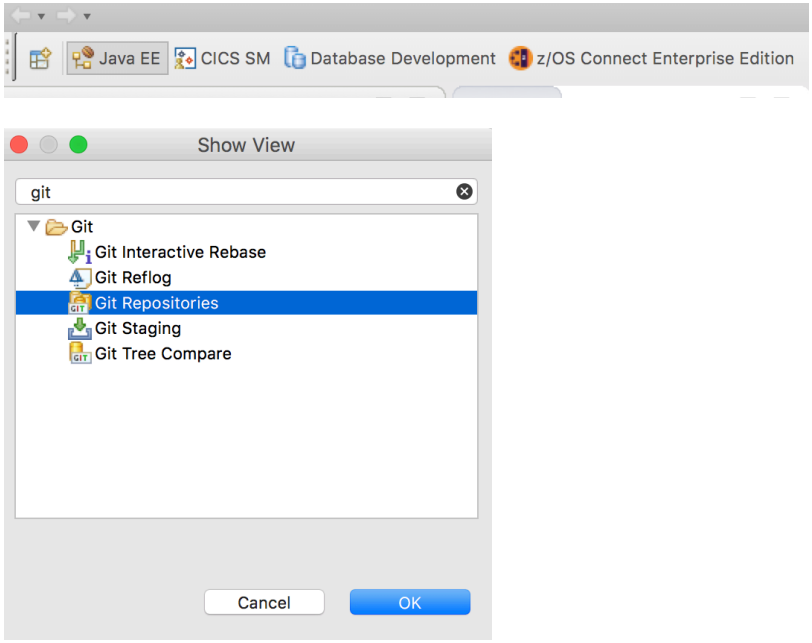
Stuffs in the workshop

Workshop A

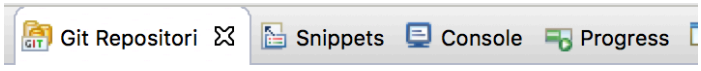
Import the CICS projects into Eclipse

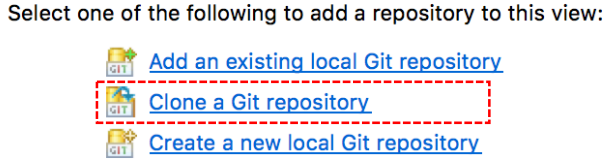
We have provided the source code to you, so you need to follow following steps to import the projects into your Eclipse work space.

1. In Eclipse, Open ***Java EE*** Perspective and then open "Git" (***Window > Show View > Other… > Git***), add "***Git Repositories***" view.

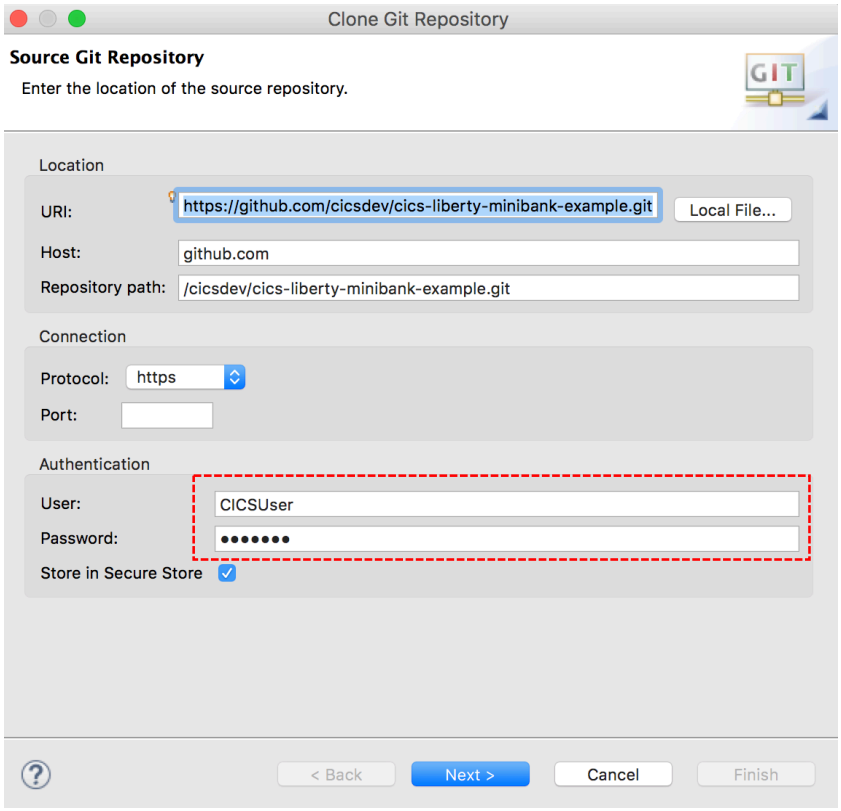


2. Then ***Clone a Git Repository*** from the view.

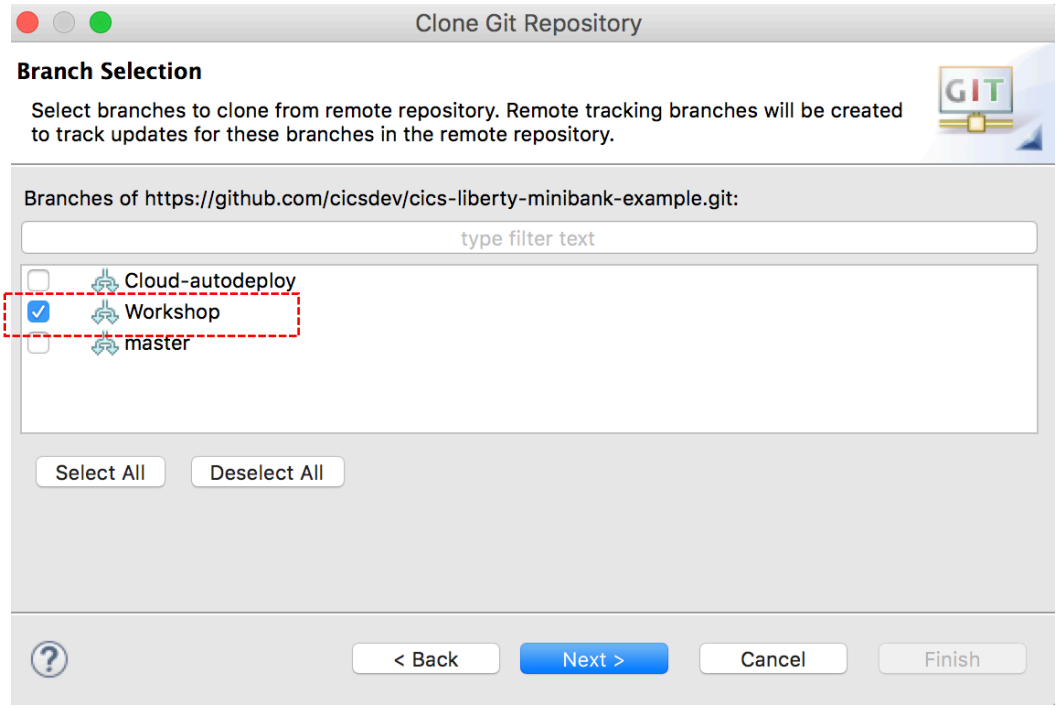




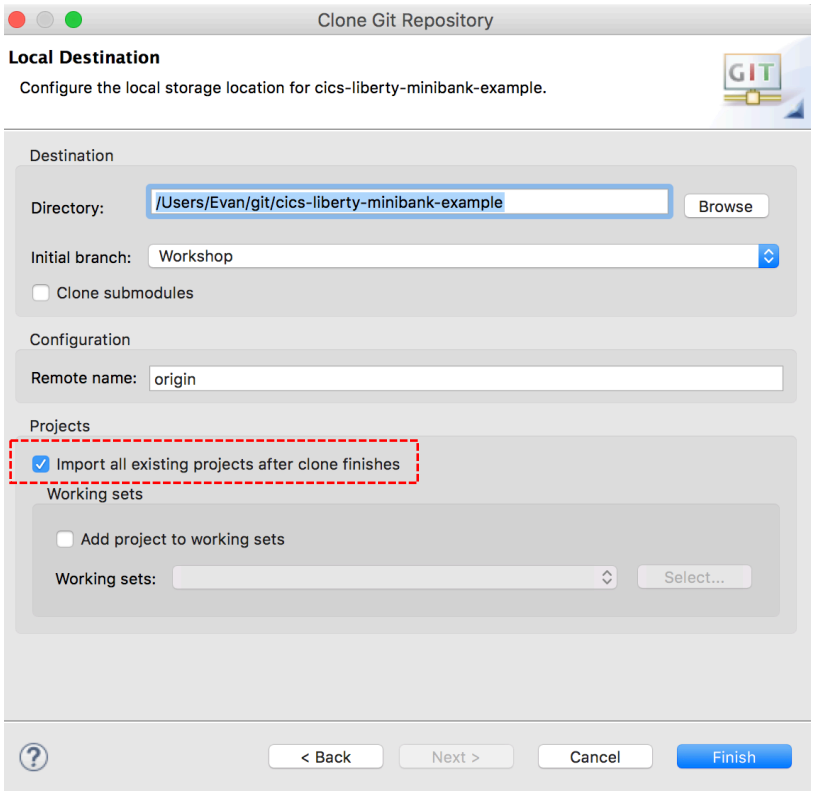
3. In the pop-up wizard, Enter the location of the source repository, click 'Next'.
   URI: https://github.com/cicsdev/cics-liberty-minibank-example.git
   User: ***CICSUser***
   Password: ***passw0rd***



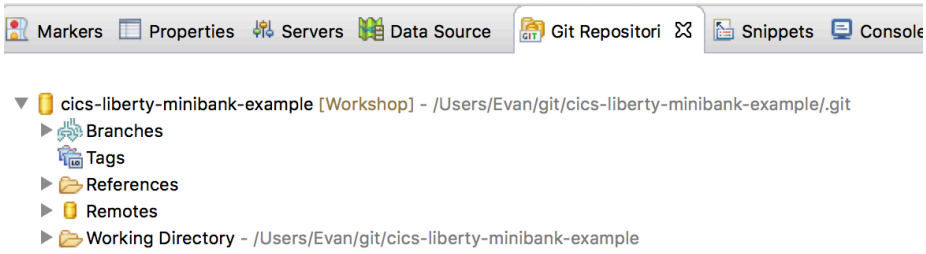
4. First ***Deselect All***, then select Branch "**Workshop**" only, click 'Next', then use the default setting in Windows system, click 'Finish'. ***Please don't select other branches.***



Then next panel, check "***Import all existing projects after clone finishes***". You can also import the project later by specifying the Destination Directory you specified below.



5. The source code would be downloaded and you should be able to see the following.



And the project files should be all in your workspace.

Below is brief introduction on these projects.

Application projects

com.ibm.cics.minibank.local.webapp:

Dynamic web project which can be developed and tested in your laptop Liberty first and would access to the service from CICS AOR via JCA local ECI adapter.

Then the application is packaged in ***com.ibm.cics.minibank.WOR.application.bundle*** *and is to be deployed in CICS WOR Liberty JVM server* ***LIBMINI****.*

com.ibm.cics.minibank.AOR.OSGI and com.ibm.cics.minibank.common.OSGI:

OSGI projects for AOR which contains back-end mini bank Java classes to create user, query user, create account, query account, deposit and withdraw.

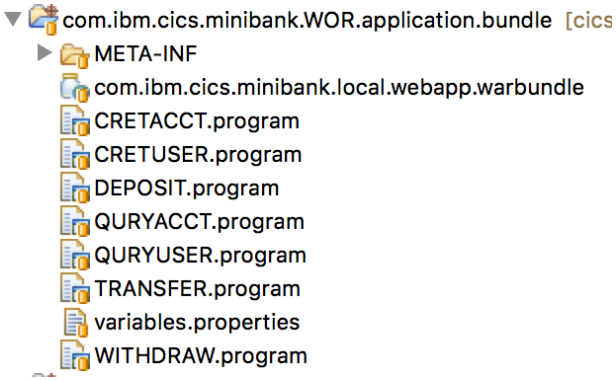
Both the OSGI projects is packaged in ***com.ibm.cics.minibank.AOR.application.****bundle and is to be deployed in CICS AOR*

*OSGI JVM server **OSGIJVM.***

System projects (Need to be exported to zFS for installation)

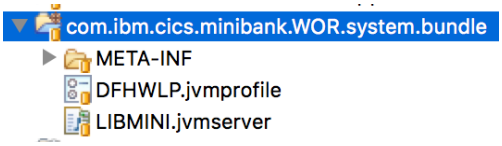
com.ibm.cics.minibank.WOR.application.bundle:

*WOR application bundle which contains a warbundle for web application **com.ibm.cics.minibank.local.webapp**, 7 remote program definition and a file called **variables.properties**. The file is used for variable resolving usage. For more information, please have a read at Variables in a CICS project*



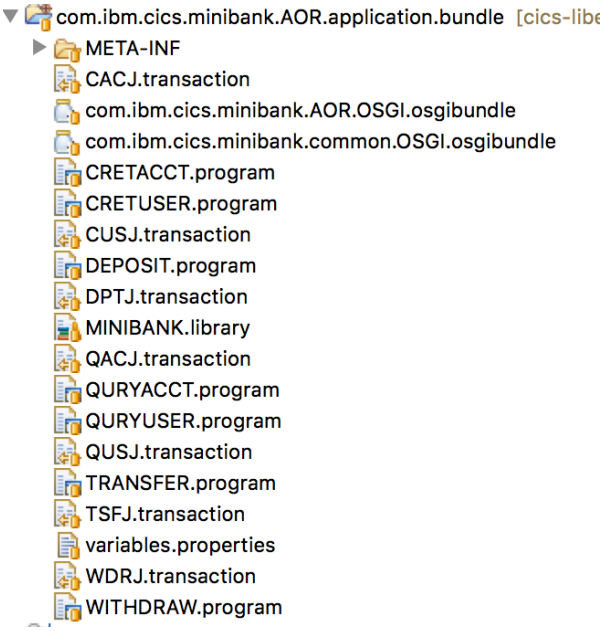
com.ibm.cics.minibank.WOR.system.bundle

*WOR system bundle which contains system wide definitions, here, mainly a JVM server to host Liberty called **LIBMINI**. WOR application runs in LIBMINI.*



com.ibm.cics.minibank.AOR.application.bundle:

*AOR application bundle which contains program definitions (6 OSGI based Java programs + 1 COBOL program), 7 mirror transactions to differentiate the workload, 2 osgibundles for **com.ibm.cics.minibank.AOR.OSGI** and **com.ibm.cics.minibank.common.OSGI**, and a variables.properties file.*



com.ibm.cics.minibank.AOR.system.bundle :

AOR system bundle which contains system wide definitions, here, mainly a JVM server to host OSGI called ***OSGIJVM****. AOR OSGI Java application runs in OSGIJVM.*

If you see any errors after importing projects, check Appendix A, make sure you have setup JRE and setup Plug-in Target Platform or Appendix D to resolve runtime problem for ***com.ibm.cics.minibank.local.webapp.***

Develop local web application in laptop

WebSphere Application Server as well as CICS remote development feature for Java have already been set up in your Eclipse. For the detailed steps, please refer to Appendix C.

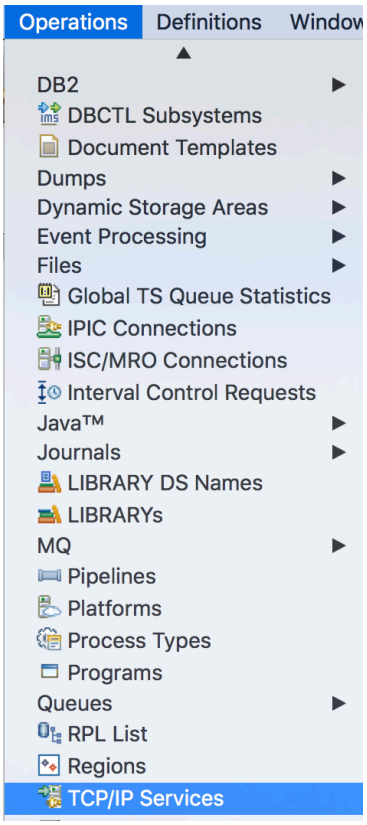
About JCA Programming interface

JCA connects enterprise information systems such as CICS, to the JEE platform.

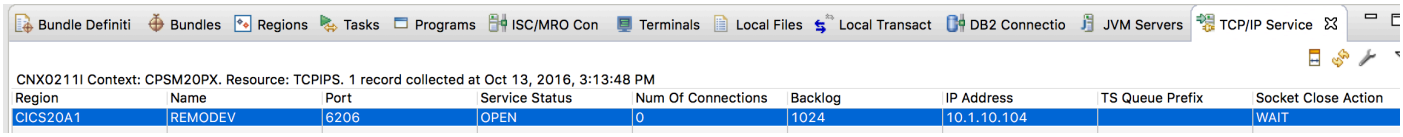
JCA supports the qualities of service for security credential management, connection pooling and transaction management, provided by the JEE application server. Using JCA ensures these qualities of service are managed by the JEE application server and not by the application. This means the programmer is free to concentrate on writing business code and need not be concerned with quality of service. For information about the provided qualities of service and configuration guidance see the documentation for your JEE application server. JCA defines a programming interface called the Common Client Interface (CCI). This interface can be used with minor changes to communicate with any enterprise information system.

Configure IPIC TCPIPSERVICE in AOR

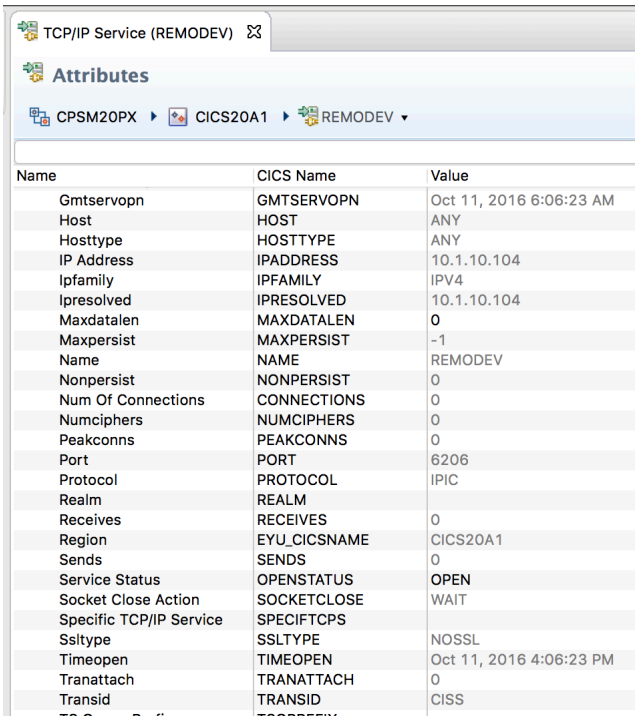
1. TCPIPSERIVCE has already been set up in AOR. You can use CICS SM perspective to check its status. Switch to CICS SM perspective, select your AOR CICS**A1, go to Operations->TCP/IP Services



2. You will see a TCPIPSERVICE named REMODEV listening to port 6**6.



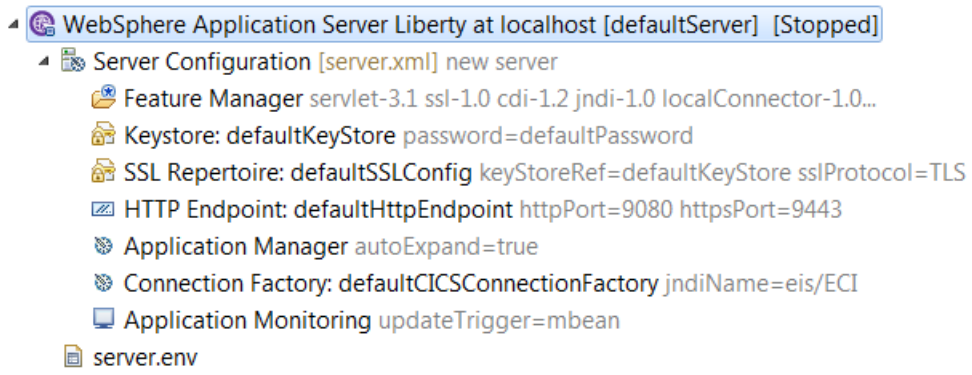
3. Double click the TCPIPSERVICE, and you can see the attributes where Protocol is IPIC and Transid is CISS.



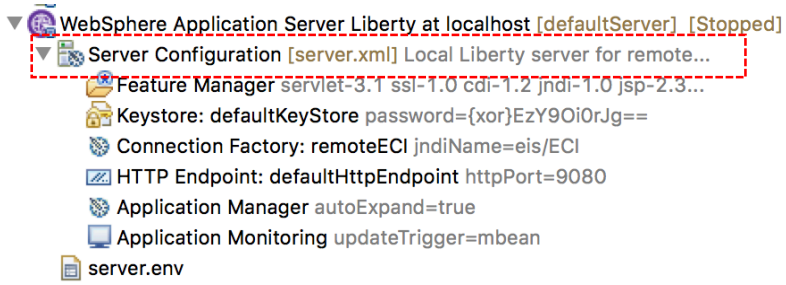
For how to define a TCPIPService for remote development purpose, check this link:
http://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.java.doc/topics/dfhpj2_jca_remote_eci_ipicconfig.html

Start your local Liberty server and configure JCA ECI feature

1. Liberty server has already set up and you can see the server in Servers view.



2. Double Click ***Server Configuration [server.xml]*** in the Servers view under your Liberty server, and you will see the content of server configuration in the server.xml.



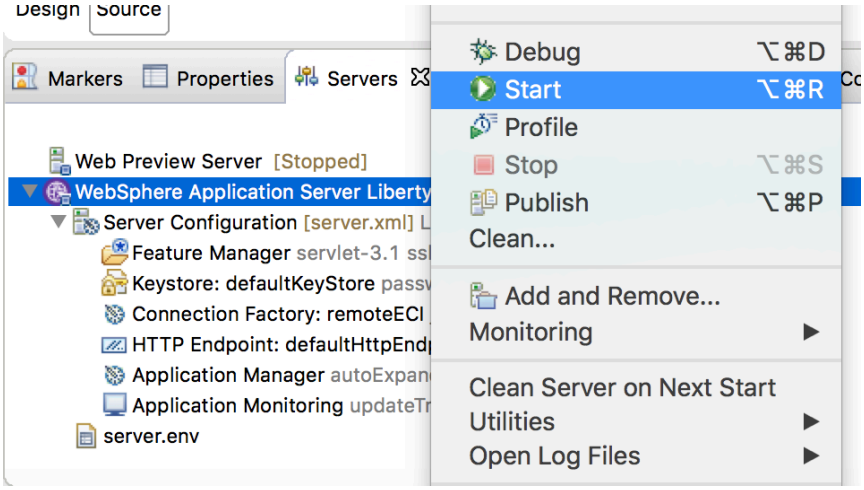
3. Verify the server.xml as below:

   Please use port 6206 in connectionFactory serverName first because a shared AOR is ready to be connected for your remote development. You can change it to 6**6 after you set up your own AOR later.

```
<server description="Local Liberty server for remote development">
    <!-- Enable features -->
    <featureManager>
        <feature>servlet-3.1</feature>
        <feature>ssl-1.0</feature>
        <feature>cdi-1.2</feature>
        <feature>jndi-1.0</feature>
        <feature>jsp-2.3</feature>
    </featureManager>

    <!-- Default key store, password is Liberty -->
    <keyStore id="defaultKeyStore" password="{xor}EzY9Oi0rJg==/>

    <!-- Connection factory used to connect to CICS JCA ECI adapter, please change the port to yours -->
    <connectionFactory id="remoteECI" jndiName="eis/ECI">
```

```
        <properties.com.ibm.cics.wlp.jca.remote.eci serverName="tcp://10.1.10.104:6206"/>
    </connectionFactory>


    <!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
    <httpEndpoint httpPort="9080" id="defaultHttpEndpoint"/>

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <applicationMonitor updateTrigger="mbean"/>

</server>
```

4. Start the server



5. You will the server is ready to service with following messages:

6. Add CICS remote development feature ***<feature>usr:jcaRemoteEci-1.0</feature>*** in the feature list after **<u>your liberty is up</u>** and save.

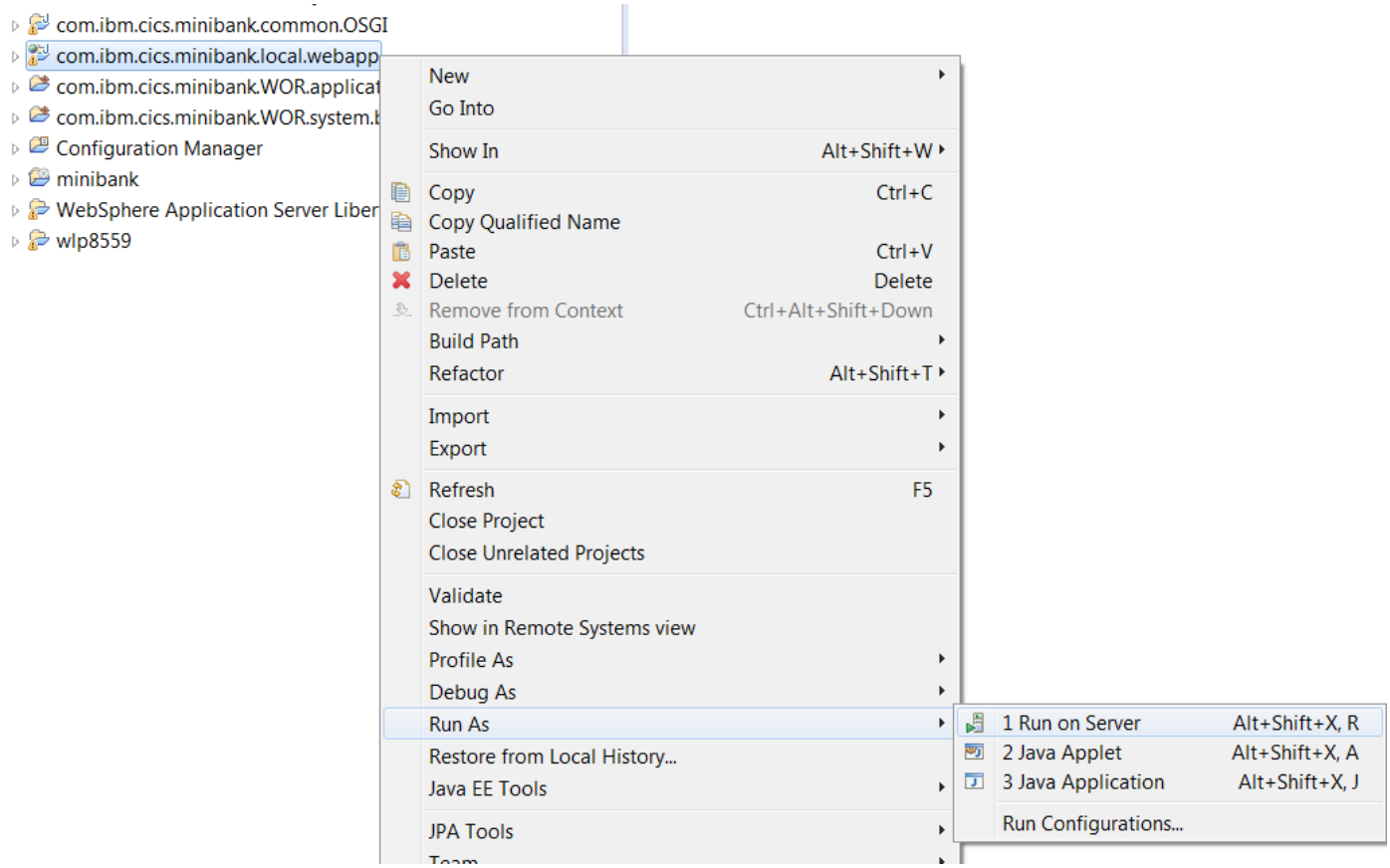
```
<!-- Enable features -->
<featureManager>
    <feature>servlet-3.1</feature>
    <feature>ssl-1.0</feature>
    <feature>cdi-1.2</feature>
    <feature>jndi-1.0</feature>
    <feature>jsp-2.3</feature>
    <feature>usr:jcaRemoteEci-1.0</feature>
</featureManager>
```

7. Liberty would pick up the change automatically. From console, you can see CICS JCA ECI resource adapter is installed.

```
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.375 seconds.
[AUDIT   ] CWWKF0012I: The server installed the following features: [jca-1.7, usr:jcaRemoteEci-1.0].
[AUDIT   ] CWWKF0008I: Feature update completed in 0.376 seconds.
[AUDIT   ] J2CA7001I: Resource adapter com.ibm.cics.wlp.jca.remote.eci installed in 0.465 seconds.
[AUDIT   ] J2CA7009I: The resource adapter com.ibm.cics.wlp.jca.remote.eci has uninstalled successfully.
[AUDIT   ] J2CA7001I: Resource adapter com.ibm.cics.wlp.jca.remote.eci installed in 0.034 seconds.
```

Run local web application in the local Liberty

1. Right click ***com.ibm.cics.minibank.local.webapp*** and select ***Run As->Run on Server***, and select your Liberty server



2. Liberty console shows:

```
[AUDIT   ] CWWKG0016I: Starting server configuration update.
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.006 seconds.
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/com.ibm.cics.minibank.local.webapp/
[AUDIT   ] CWWKZ0001I: Application com.ibm.cics.minibank.local.webapp started in 0.171 seconds.
```

3. Open the link in the console to access your local web application. Local Liberty would handle web request as web layer and call AOR's business logic via COMMAREA or CHANNEL/CONTAINER.

4. You can try application, e.g. create your own user and accounts, and deposit a huge amount money there to make yourself a millionaire, and transfer some to others ☺

Update JVM profiles

JVM profiles are text files that contain Java launcher options and system properties, which determine the characteristics of JVMs. There is user specific information need to be updated.

WORK_DIR** is working directory on z/OS UNIX that the CICS region uses for activities that are related to Java. The CICS JVM interface uses this directory when it creates the STDOUT, STDERR, and JVMOUT files.* ***So you need to update to a USS directory under your name.

***Dcom.ibm.cics.jvmserver.wlp.server.http.port**. It is Liberty server http listener port.*
***Dcom.ibm.cics.jvmserver.wlp.server.https.port**. It is Liberty server https listen port. So you need to change them to ports assigned to you to avoid conflicts.*

Details as below. Please update them one by one.

1. In project **com.ibm.cics.minibank.AOR.system.bundle**, update **DFHOSGI.jvmprofile**
   - In line *WORK_DIR=/cicssvy/cics**/workdir* updated cics** to your teamid.

2. In project **com.ibm.cics.minibank.WOR.application.bundle**, update **variables.properties**
   - In line ***WOR.DPL.Remote_System=00A1*** updated 00A1 to your **A1. ** is your teamid.

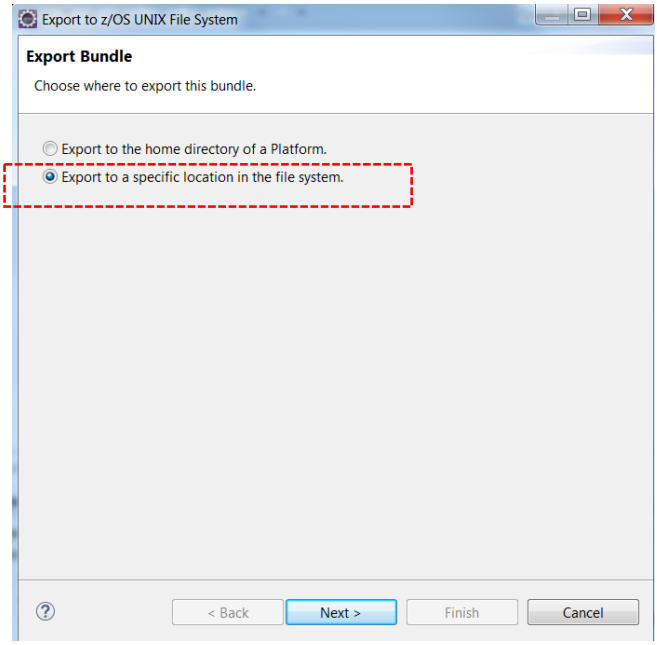
3. In project **com.ibm.cics.minibank.WOR.system.bundle,** update DFHWLP.jvmprofile
   - In line WORK_DIR=/cicssvy/cics**/workdir updated cics** to your teamid.
   - In line -Dcom.ibm.cics.jvmserver.wlp.server.http.port=6**2. Use port number assgined to your team.
   - In line -Dcom.ibm.cics.jvmserver.wlp.server.https.port=6**3. Use port number assigned to your team.

Export the projects to USS

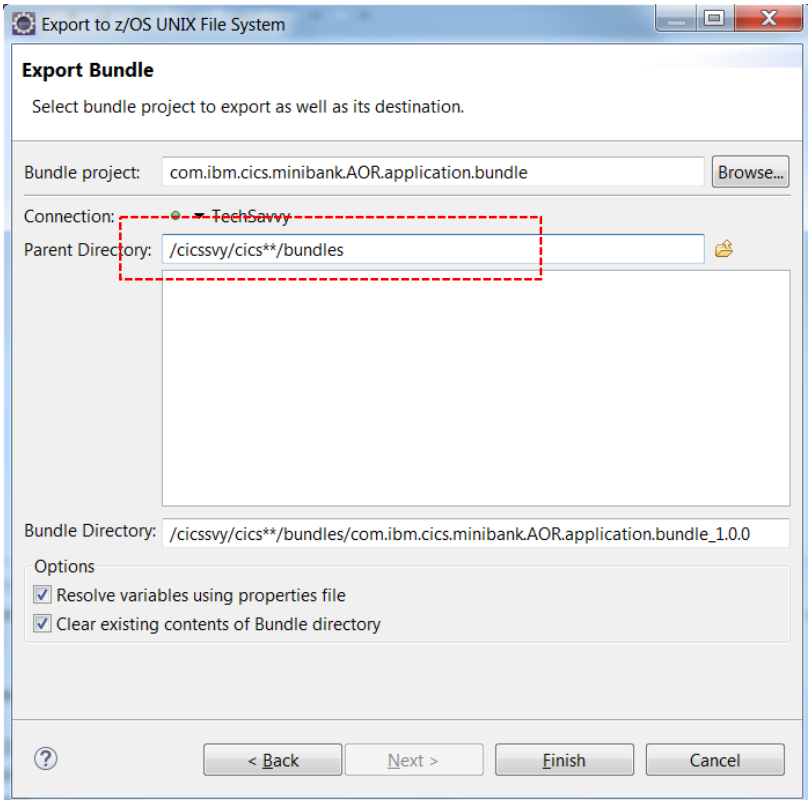
Before this step, make sure you have configured FTP connection to MVS and configured CMCI connection to MVS. If not, reference Appendix B.

A CICS bundle is a collection of resources, artifacts, references, and a manifest file that you can deploy into a CICS region to represent all or part of an application. In this step, you need to upload the below four CICS bundle projects to USS.
com.ibm.cics.minibank.AOR.application.bundle;
com.ibm.cics.minibank.WOR.application.bundle;

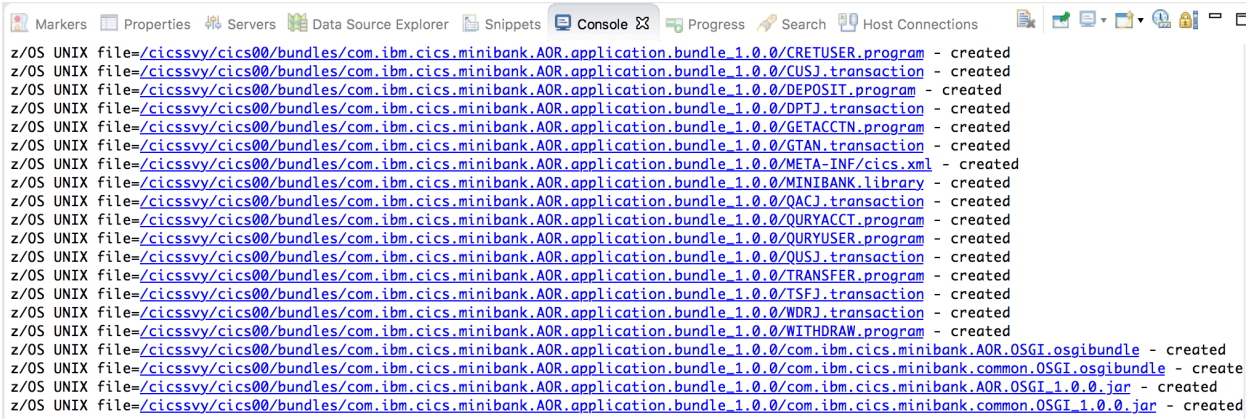
1. Right click the **com.ibm.cics.minibank.AOR.application.bundle,** choose the menu "***Export Bundle Project to z/OS UNIX File System…***"
2. Select the "***Export to specific location in the file system***"



Enter your USS folder **(/cicssvy/cics**/bundles**) as Parent Directory to export the bundle. It does not matter the directory is not there because the tool will create one for you. if you already have old content in the folder, for example the old version of the bundle, you can enable the check box "Clear existing contents of Bundle directory" to clean the folder before exporting. Click **Finish** to export the **com.ibm.cics.minibank.AOR.application.bundle** bundle.



3. After you press finish, you will see the similar output in the console:(*note: please record down the bundle directory before you press Finish, it will be used later when you define the bundle resource*)



4. Repeat step 1-3 to upload below left 3 bundle projects.

com.ibm.cics.minibank.AOR.system.bundle ;

com.ibm.cics.minibank.WOR.application.bundle;

com.ibm.cics.minibank.WOR.system.bundle;

Define resources for WOR and AOR

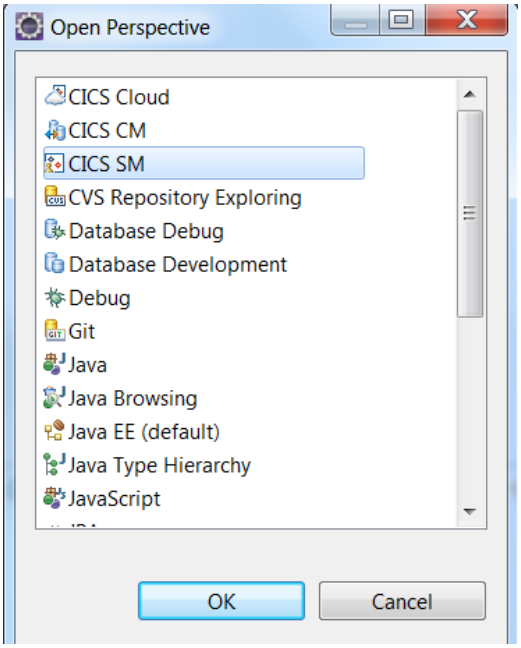
Before define resources, you need to set up CMCI connection to your CPSM environment. Please refer to ***Appendix B*** for how.

All bundle artifacts are at USS now. In this step, you need to define and install CICS bundle resource by CICS explorer.

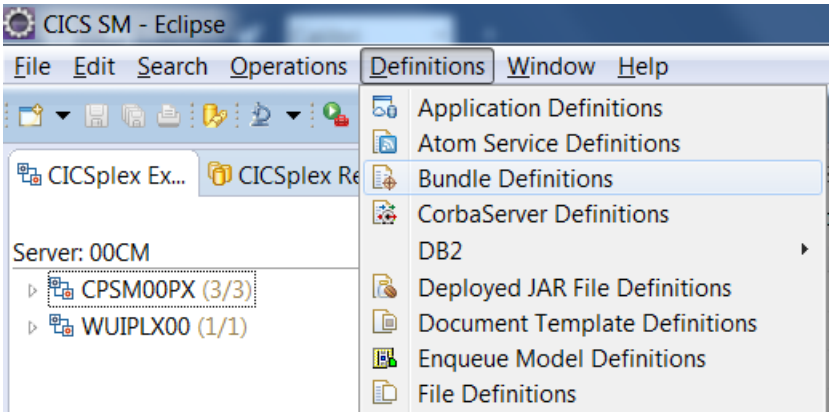
- For WOR:
    - BUNDLE: WAPPBUND – for WOR application bundle
    - BUNDLE: WSYSBUND – for WOR system bundle
- For AOR:
    - BUNDLE: AAPPBUND – for AOR application bundle
    - BUNDLE: ASYSBUND – for AOR system bundle

.

1. Open perspective and choose **CICS SM**.

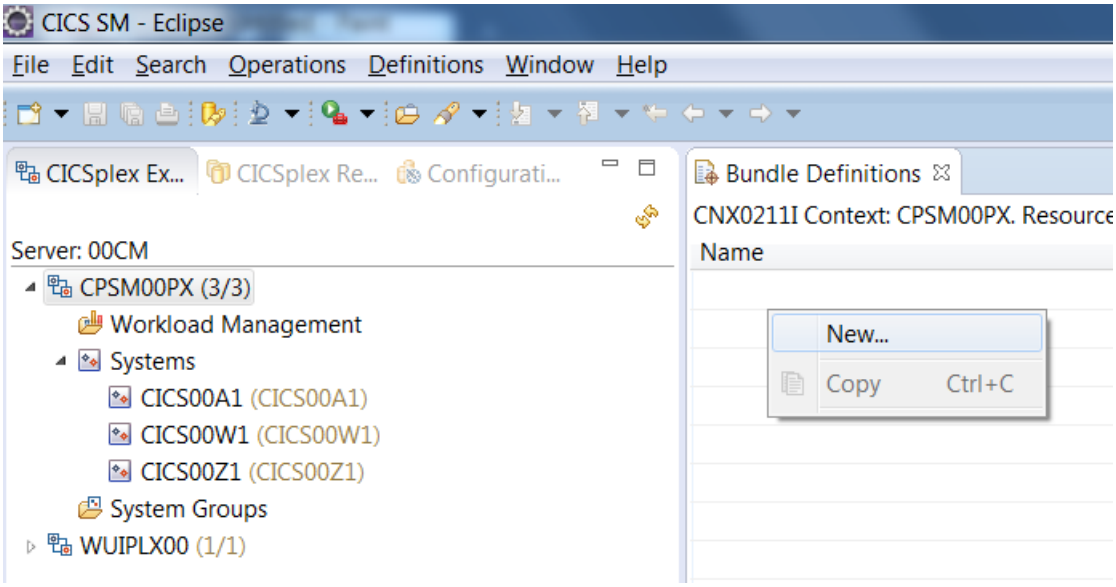


2. Go to Definitions -> Bundle Definitions

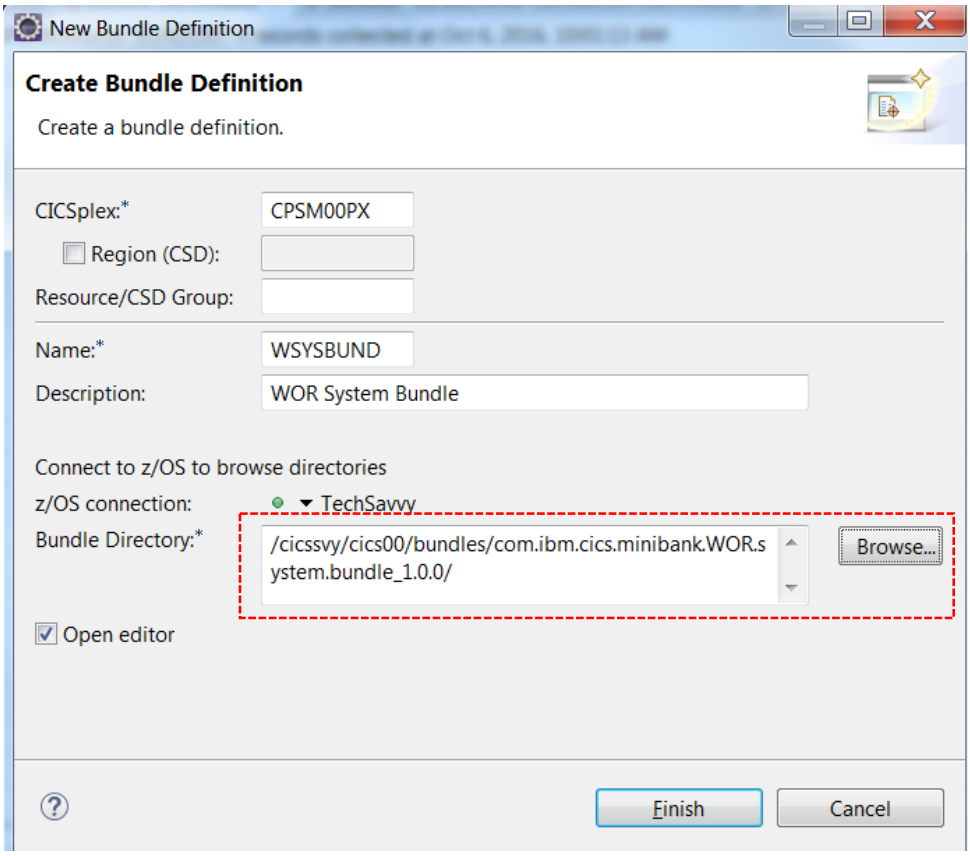


3. Click ***CPSMPX**** in CICSplex Explorer, it will be the CONTEXT for the resource definitions.

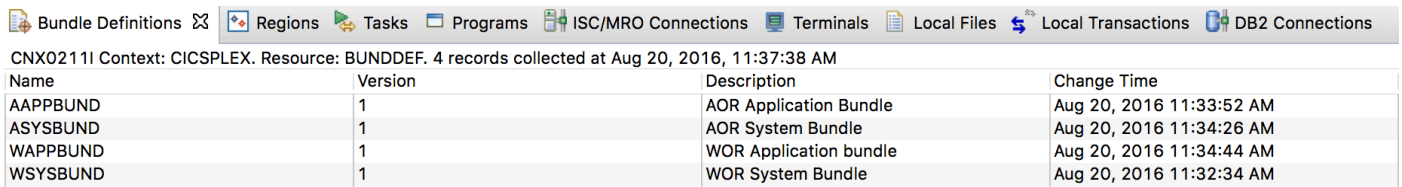
4. Right-click the empty area, and press New.



5. Input ***Name*** and ***Bundle Directory*** for the definition. (*note: Bundle Directory is the directory where you uploaded your Bundle project in USS, you can either copy the path or browse the USS directory*)

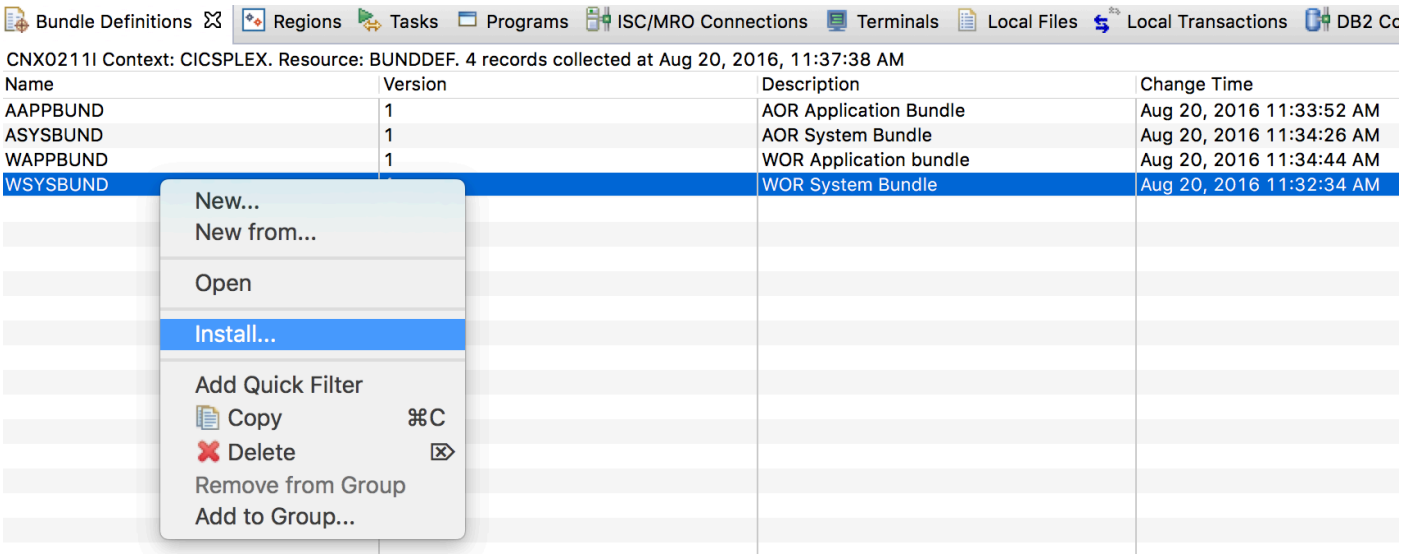


6. Repeat step 4-5 to recreate bundle definitions for other three bundles, and finally you will have four bundle definitions.

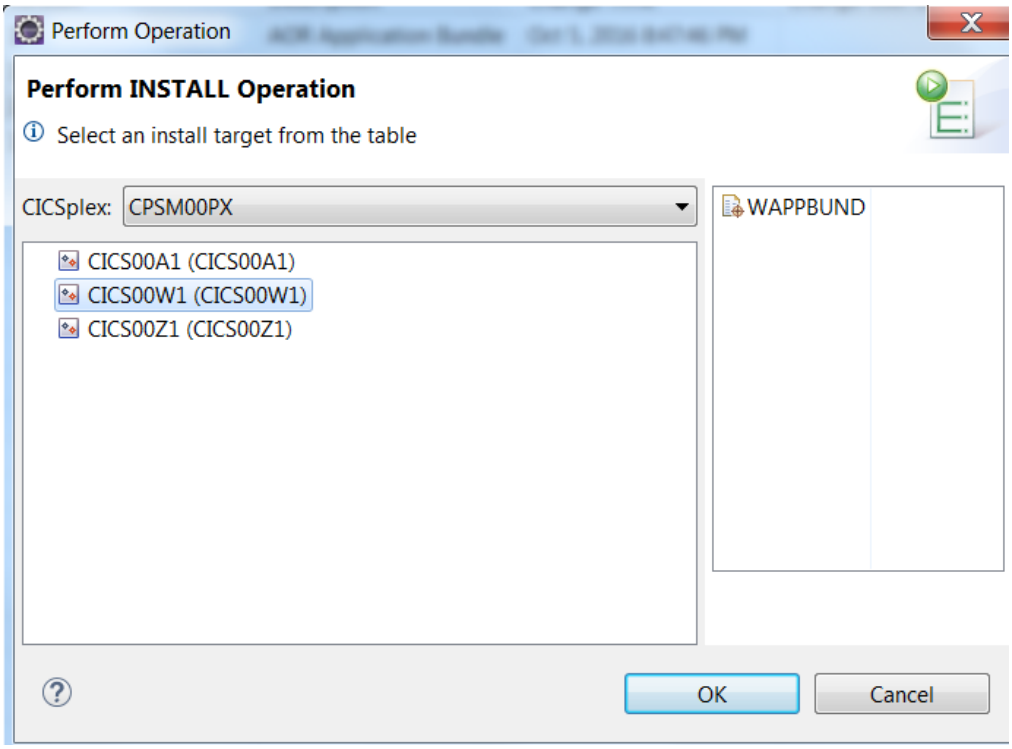


Name	Version	Description	Change Time
AAPPBUND	1	AOR Application Bundle	Aug 20, 2016 11:33:52 AM
ASYSBUND	1	AOR System Bundle	Aug 20, 2016 11:34:26 AM
WAPPBUND	1	WOR Application bundle	Aug 20, 2016 11:34:44 AM
WSYSBUND	1	WOR System Bundle	Aug 20, 2016 11:32:34 AM

7. Now you are ready to install the BUNDLEs. Remember to install system bundles before application bundles because of dependency. Let use WSYSBUND as example to show installation process. Right-click on WSYSBUND, and press Install.



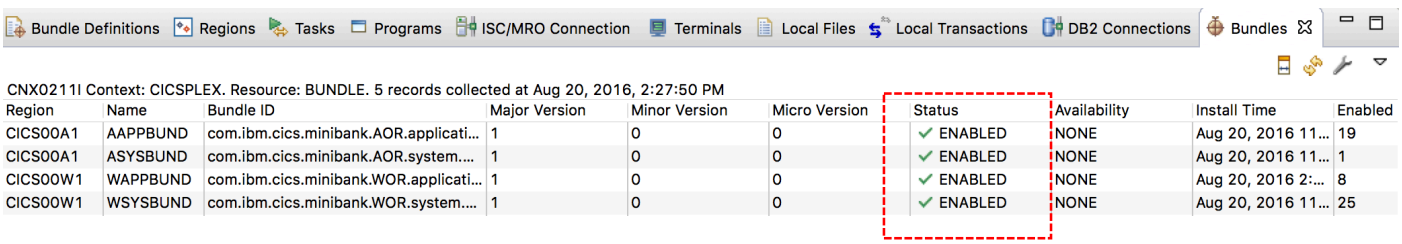
8. Select the target region, CICS**W1



9. Repeat step 7-8 to install below bundles **in sequence**:
   - WAPPBUND to CICS**W1.
   - ASYSBUND to CICS**A1.
   - AAPPBUND to CICS**A1.

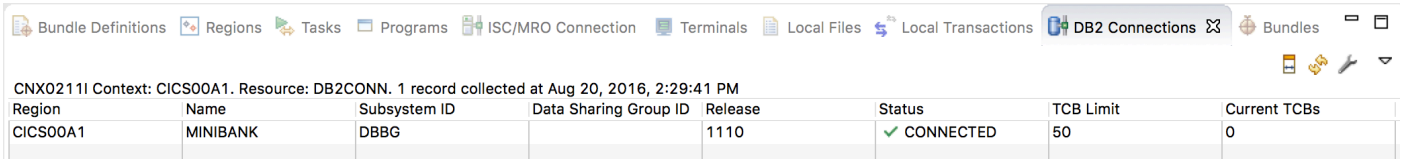
   Finally, you have four bundle definitions installed, two for CICS**W1, and two for CICS**A1.

10. Click Operations -> Bundle to review all the bundles installed on the system. You will be able to see:

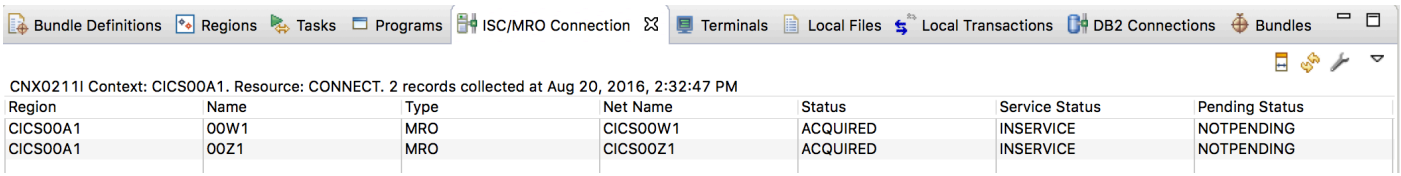


Following resources were already setup, please check if they are installed correctly.

11. Go to Operations -> DB2 -> DB2 Connections, and check the DB2 definition status. Right-click on it and press ***Connect*** in case the status is ***Notconnected***.

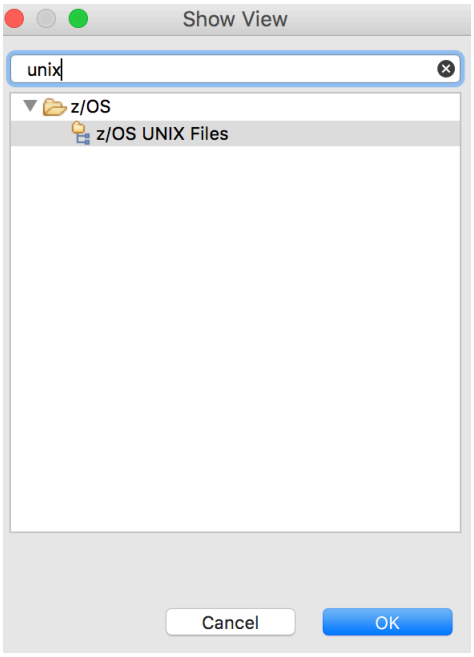


12. Go to Operations->ISC/MRO connections, check your WOR to AOR connection is in acquired status.

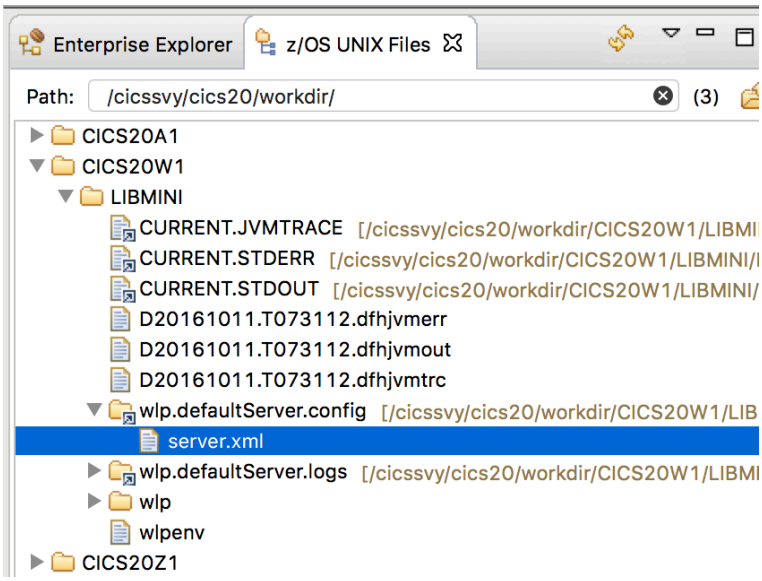


Tailor CICS Liberty to make the web application work

1. Now you have already installed a CICS bundle named ***com.ibm.cics.minibank.WOR.application.bundle*** which contains the local web application.
2. Now open ***z/OS Unix Files*** view



3. Enter your USS workdir ***/cicssvy/cics**/workdir***, and expand subdirectory ***CICS**W1/LIBMINI/wlp.defaultServer.config***



4. Open the ***server.xml***, add necessary features in the feature list as below:

```
<!-- Enable features -->
<featureManager>
    <feature>jsp-2.2</feature>
    <feature>ssl-1.0</feature>
    <feature>cicsts:core-1.0</feature>
    <feature>cicsts:jcalocalEci-1.0</feature>
    <feature>jndi-1.0</feature>
    <feature>cdi-1.0</feature>
</featureManager>
```

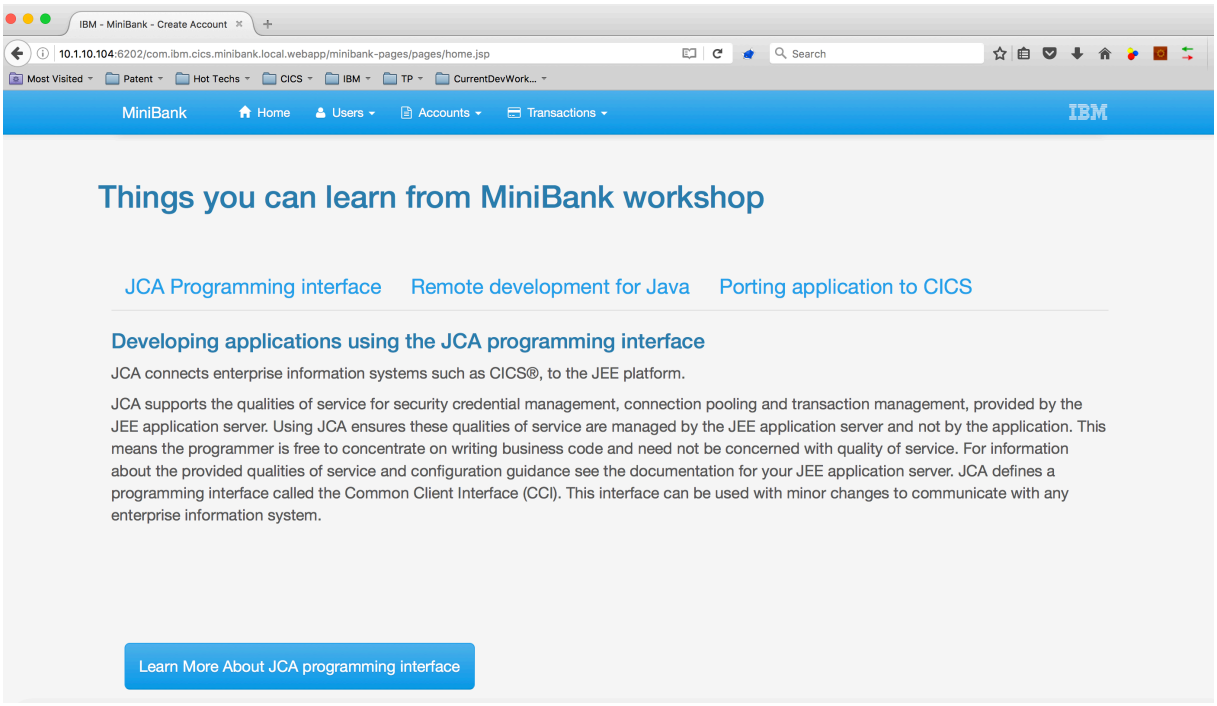
5. Add connectionFactory in the server.xml and save all the change

```
<connectionFactory id="localEci" jndiName="eis/ECI">
    <properties.com.ibm.cics.wlp.jca.local.eci/>
</connectionFactory>
```

6. Then open ***CURRENT.STDOUT*** from LIBMINI directory and you should be able to see liberty pick the update automatically as below:

```
[AUDIT   ] CWWKG0017I: The server configuration was successfully updated in 0.452 seconds.
[INFO    ] CWWKZ0018I: Starting application com.ibm.cics.minibank.local.webapp.
[INFO    ] SRVE0169I: Loading Web Module: com.ibm.cics.minibank.local.webapp.
[INFO    ] SRVE0250I: Web Module com.ibm.cics.minibank.local.webapp has been bound to default_host.
[AUDIT   ] CWWKT0016I: Web application available (default_host): http://10.1.10.106:6202/com.ibm.cics.minibank.local.webapp/
[AUDIT   ] CWWKZ0001I: Application com.ibm.cics.minibank.local.webapp started in 1.131 seconds.
```

7. Access to the link showed in the STDOUT for your web application inside CICS liberty and you will see the same page as your local liberty server.



8. You should already notice the JCA feature is named as “local” and you don’t need to specify TCPIP address any more. That is because CICS has made this JCA communication running via CICS intercommunication method DPL. So it means you can deploy the application without any code change. ***Cool?***

Workshop B

Jenkins and CICS Build Toolkit

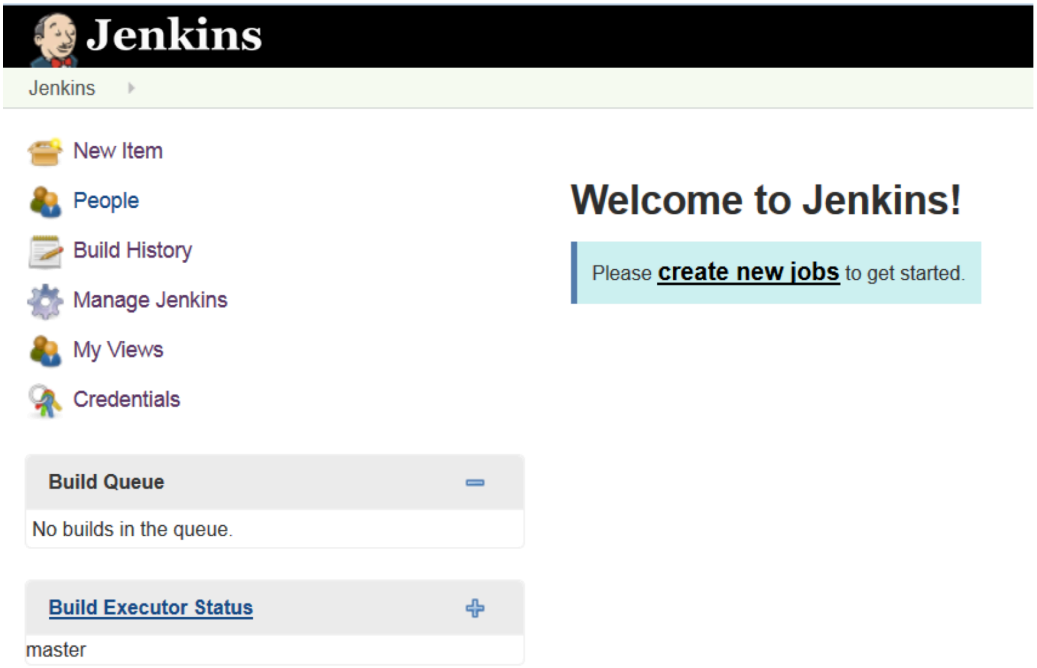
1. Log in Jenkins, launch your web browser

   URL: http://<jenkins_server_ipaddress>:8080/

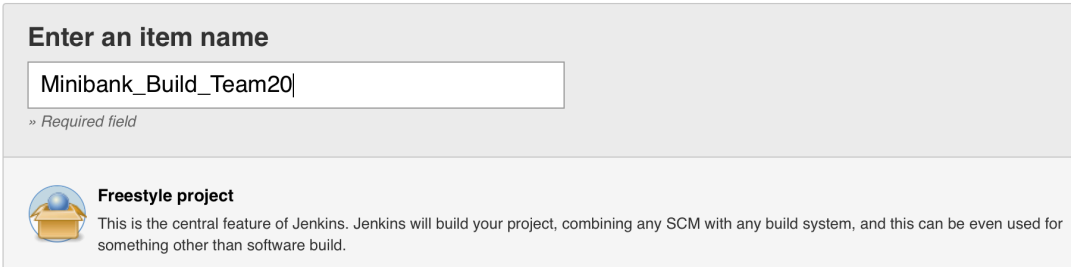
   Jenkins_server_ipaddress: ***9.185.189.181 (Could be changed, if not work, ask for help)***

   User: ***team*****

   Password: ***team*****



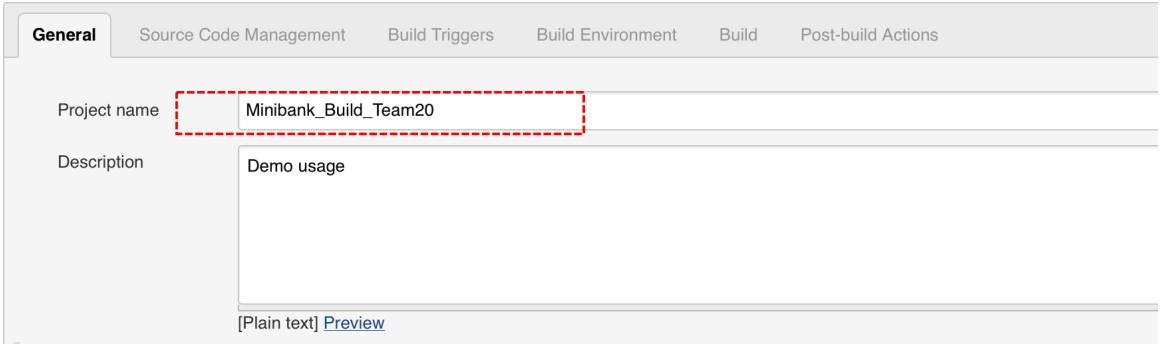
2. Click "**New Item**" and select "**Freestyle Project**", input a name e.g. **Minibank_Build_Team****, click 'OK'

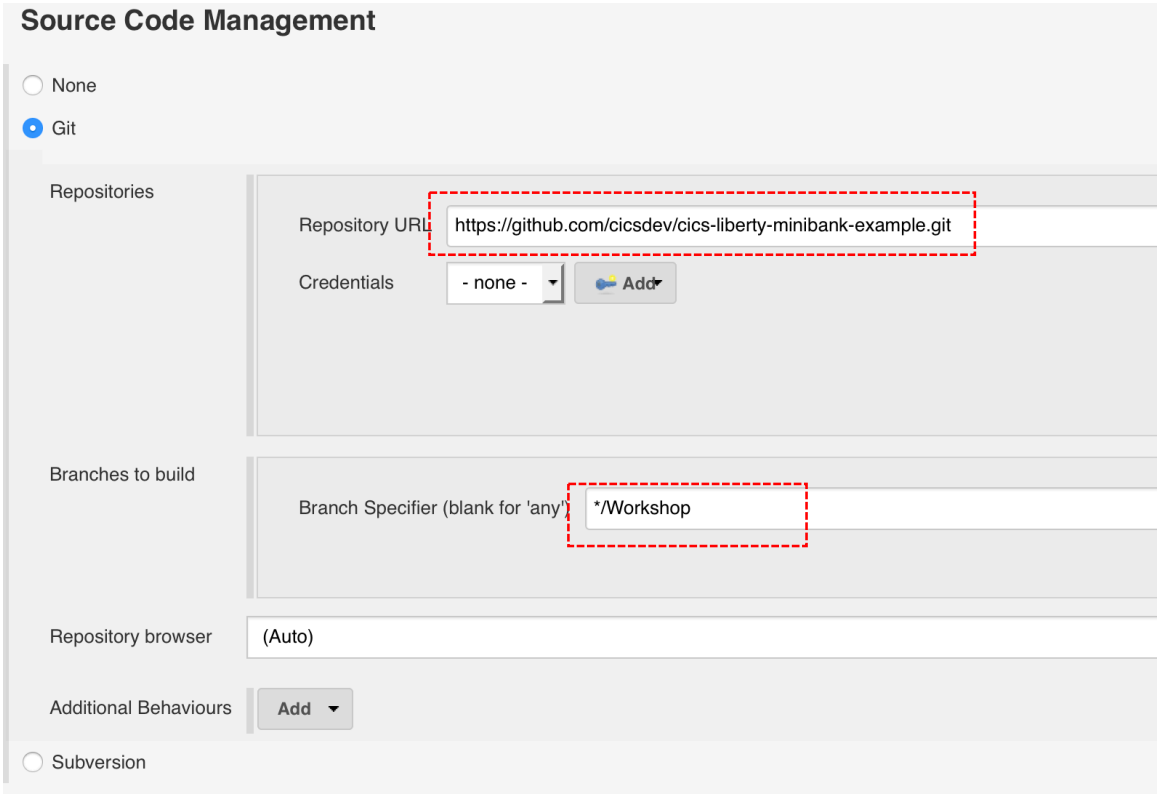


3. Minibank_Build configure parameter

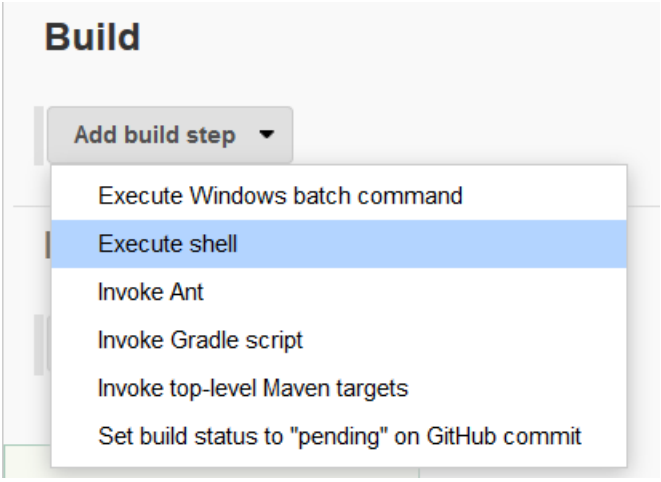
   Repository URL: https://github.com/cicsdev/cics-liberty-minibank-example.git

   Branch Specifier: */Workshop





4. For **Build** environment setting, add build step "**Execute shell**"



5. Add scripts to run CICS build toolkit under Execute shell. You can find a script file called ***Jenkins_cicsbt.sh*** on your

desktop and you can also find the script under the demo Jenkins project ***Minibank_Build_Team20***. Now copy the script into your own Jenkins project and change “team20” in the script into your own team id, team**.

Jenkins ▸ Minibank_Build_Team20 ▸

Build

General | Source Code Management | Build Triggers | Build Environment | **Build** | Post-build Actions

Execute shell

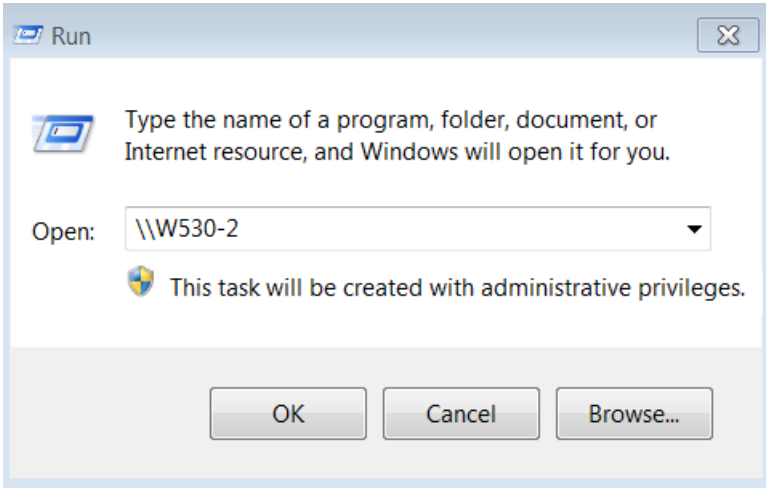
Command

```
#Export system variables for CICS build toolkit
export CICSBT_HOME=C:/cicsbt-5.3.8/cicsbt/
export CICSBT_OUTPUT=$CICSBT_HOME/output
export CICSBT_PROP=$CICSBT_HOME/props

#Hide command output and prevent from immediate EXIT in case of none zero exit code
set +xe

#Run cicsbt to build two CICS bundles from Jenkins workspace where Git source codes are fetched. As the sou
#Please change team20 to your own team name, e.g. team01
$CICSBT_HOME/cicsbt --input $WORKSPACE/com.ibm.cics.minibank.* --build com.ibm.cics.minibank.WOR.applicatio
rc=$?
if [ $rc -eq 0 ]; then
    echo "cicsbt build has completed successfully with no error or warning."
elif [ $rc -eq 1 ]; then
    echo "cicsbt build has completed successfully with warning"
    echo "Return code from cicsbt is " $rc
    echo "Description: Warning. Some source projects were not imported."
elif [ $rc -eq 2 ]; then
    echo "cicsbt build completed successfully with warning"
    echo "Return code from cicsbt is " $rc
    echo "Description: Warning. Artifacts were built, with warnings."
else
    echo "cicsbt build failed with return code " $rc
    echo "Please check log."
    exit
fi

#Run cicsbt to resolve the varibales in the unresolved bundles.
#Please change team20 to your own team name, e.g. team01
$CICSBT_HOME/cicsbt --resolve $CICSBT_OUTPUT/team20/unresolved --output $CICSBT_OUTPUT/team20/resolved --pr
rc2=$?
if [ $rc2 -eq 0 ]; then
    echo "cicsbt resolve completed successfully with no error or warning."
else
    echo "cicsbt resolve failed with return code " $rc2
    echo "Please check log."
fi
```

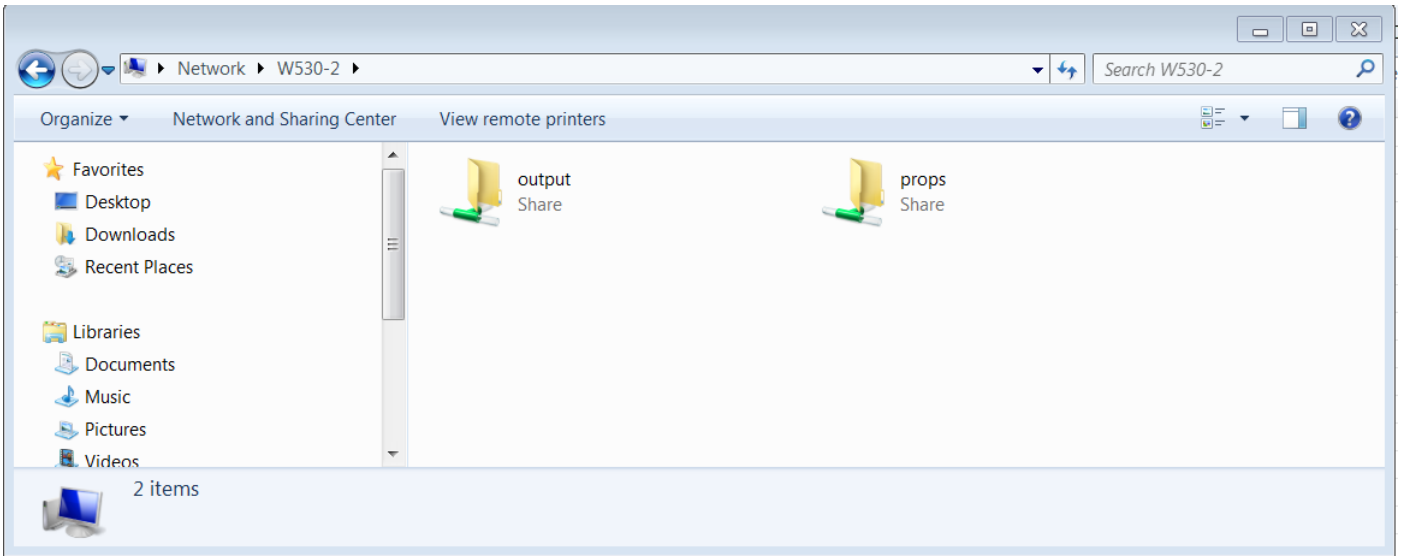
The first cicsbt invocation is to build application bundles from SCM (Github here), and the second one is to replace a variable called ***WOR.DPL.Remote_System*** (defined in the source for REMOTESYSTEM of the remote programs in WOR) with a value defined in a local properties file (specified by --properties)

6. The path you see above is all in the Jenkins machine, and there are two shared folders, one for output, and the other for you to specify your own properties.

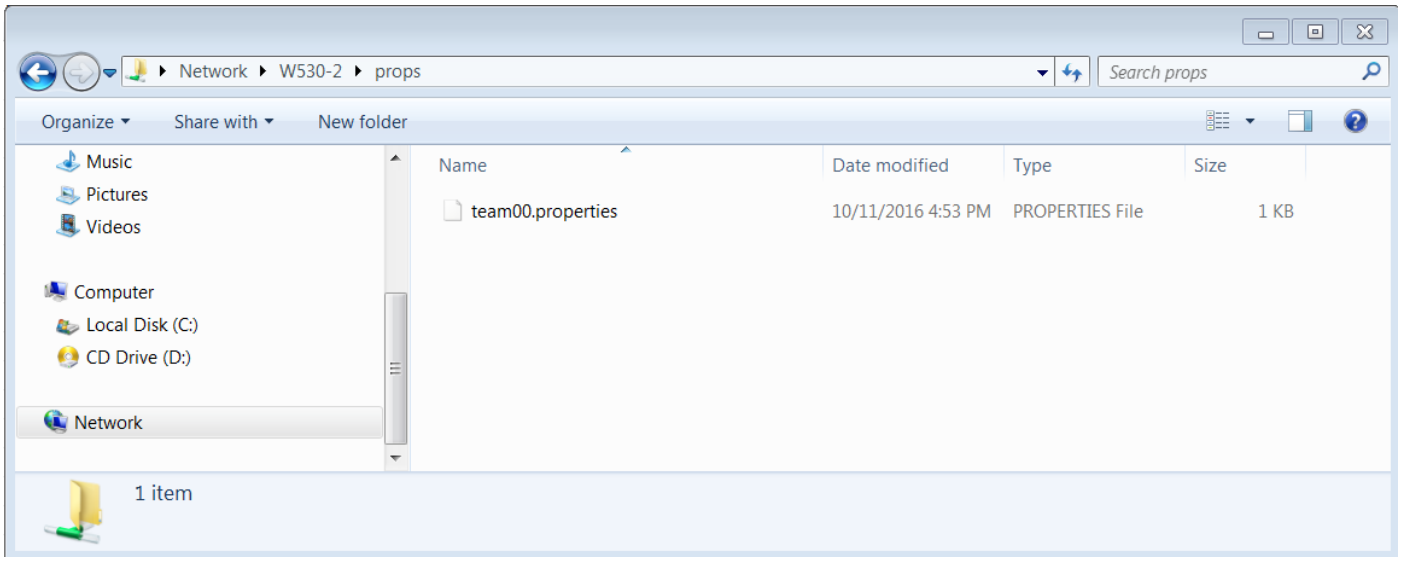
You can open ***Run*** from Windows start panel, and input Jenkis server network name, e.g. \\W530-2. Enter user/password as ***cicsuser/passw0rd.***



Then you should be able to see the shared directories as below:



7. Open ***props*** directory to create your own properties file.

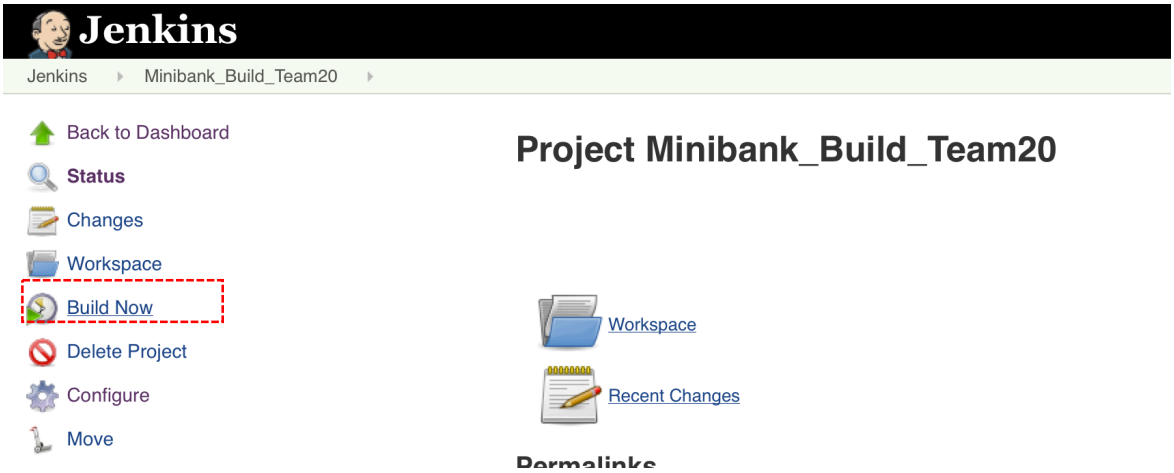


team00.properties is an example for you. You can should create a file called ***team**.properties*** as specified in the Jenkins script in step 5 with the content as below:

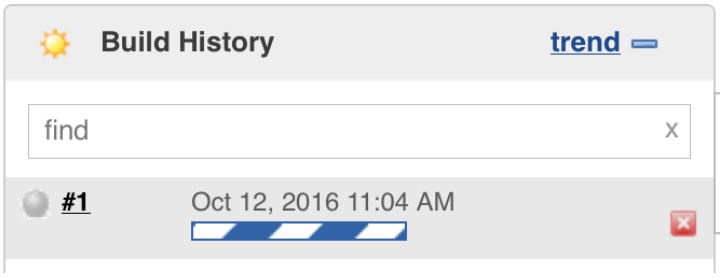
```
AOR.OSGI.JVM_Server=OSGIJVM
AOR.Library.Data_Set_Name=CICSSVY.CICS.LOAD.PDSE
WOR.DPL.Remote_System=**A1
WOR.Liberty.JVM_Server=LIBMINI
```

Please change ** to your teamid.

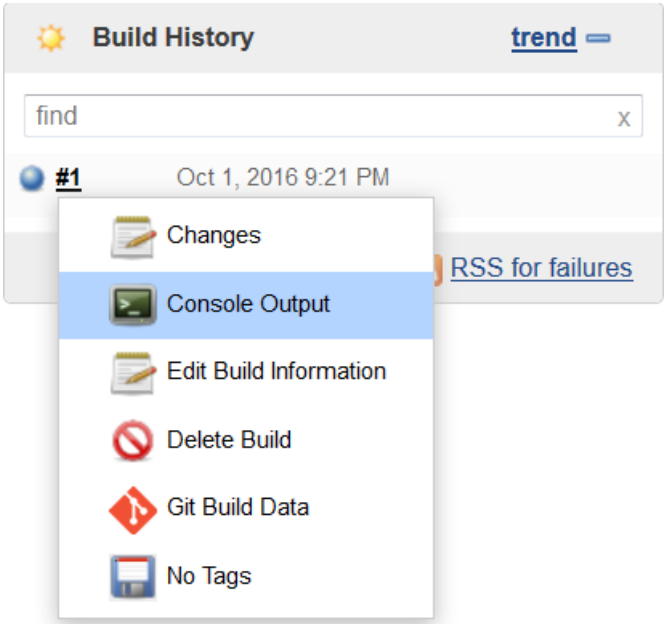
8. Now save the Jenkins item, and you are ready to run your first build, click “Build Now”



9. You will the build status from Build History and you can check the Console Output once the build completes.



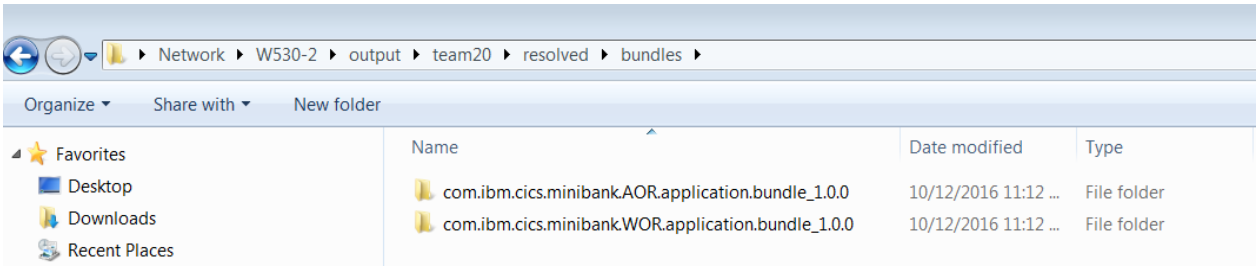
Check **Console Output** from Build History



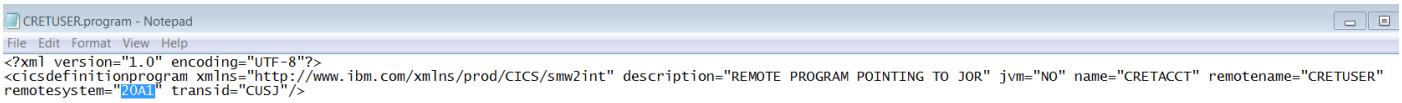
10. The Console Output showed as following, please verify the build result

11. Add parameter "**--verbose**" in shell script to get more detailed message, build again, you will see more detailed information from cicsbt.

12. Now go to shared ***output*** directory and in the subdirectory ***team**/resolved/bundles,*** you should be able to see two bundles, WOR application bundle and AOR application bundle.



13. Check the WOR bundles, and open a program definition e.g. CREUSER.program. You will see that the remotesystem value is resolved by the value in your properties file.



What next?

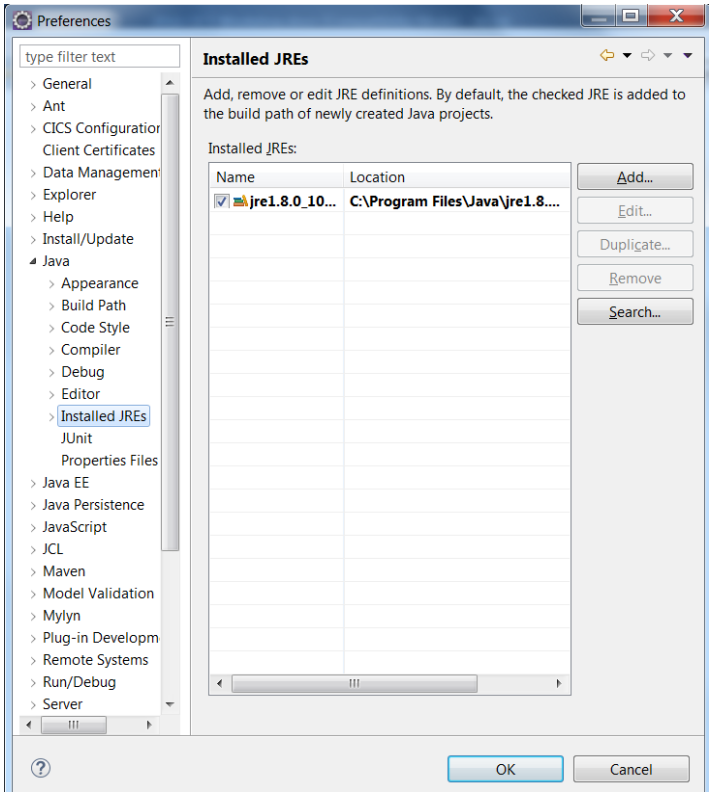
The bundles built by ***cicsbt*** is exactly the same from the bundles compiled by Eclipse IDE. You can easily write some scripts to ftp the bundles to CICS, and install these from CICS. CICS provides other utility DFHPLOY and also UCD plug-in to automate deployment part. Due to time limitation, we will not cover this in the workshop.

Congratulations! All the CICS workshop has been done now.

Appendix A: Set up Eclipse with CICS Explore SDK for CICS Java development

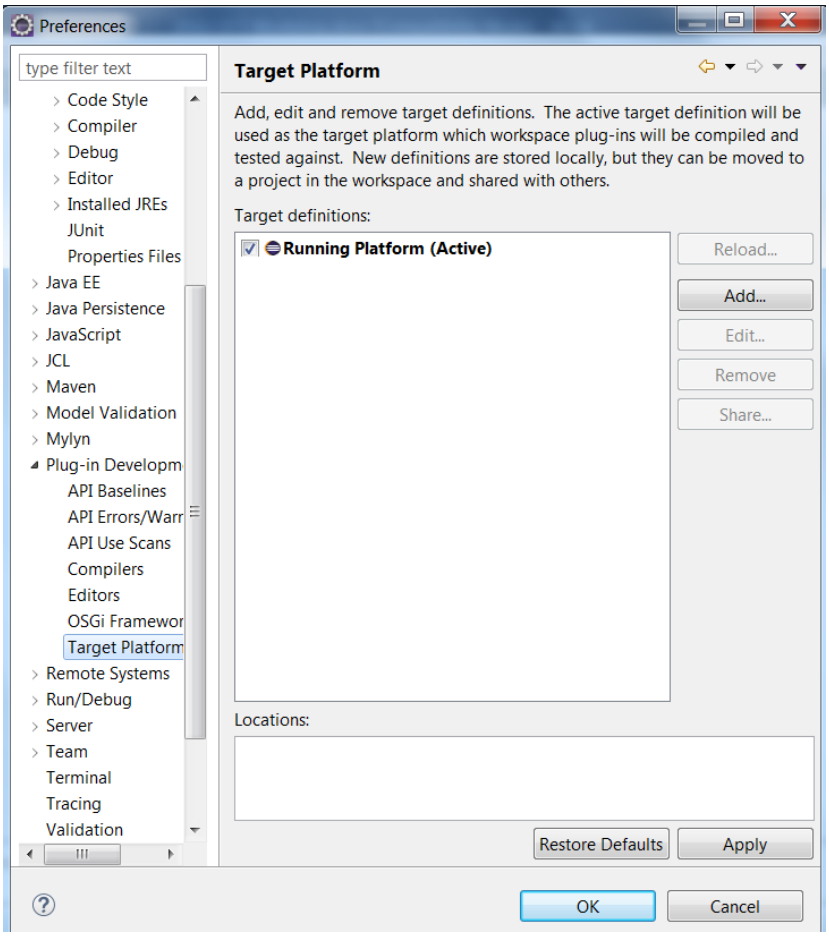
Setup JER

1. Go to “Window -> Preferences -> Java -> Installed JREs”, to add a JRE definition pointing to your local JRE folder. Please note this workshop need JRE 1.7 or above. Like below:

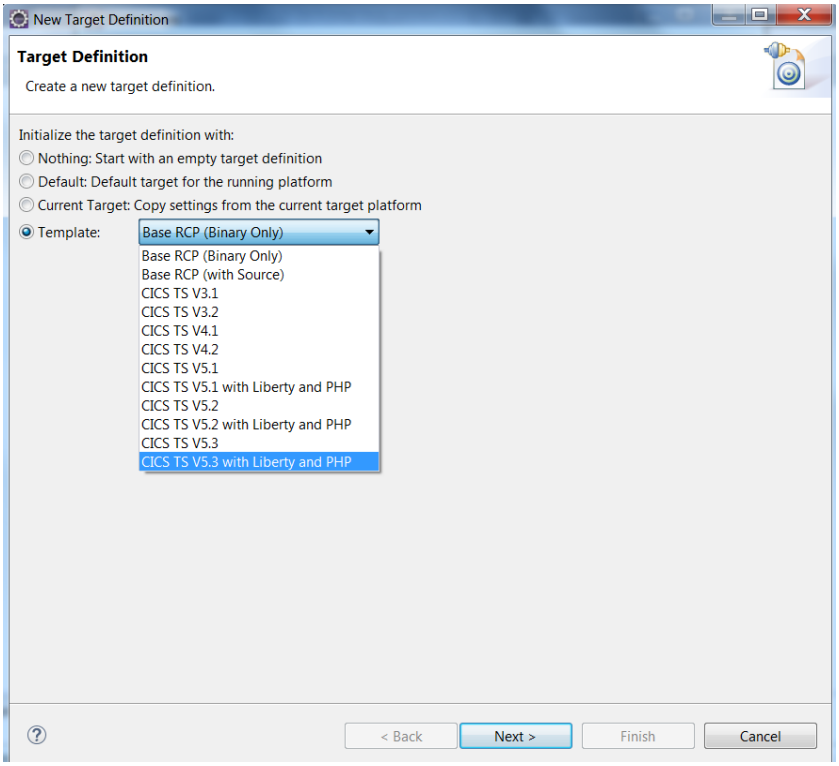


Setup Plug-in Target Platform

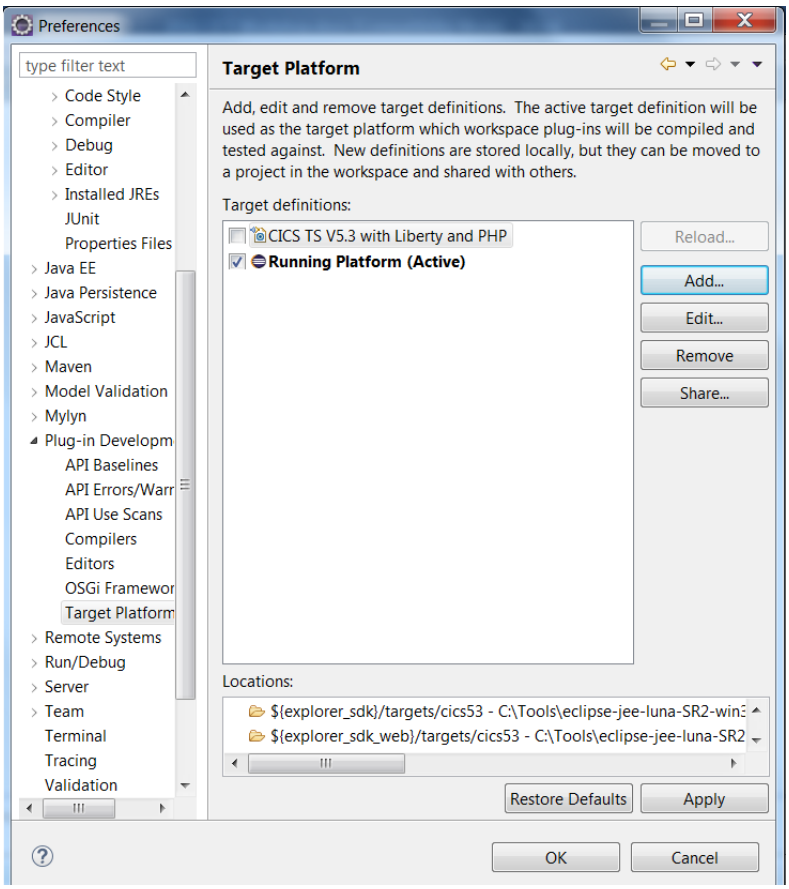
1. Go to “Window -> Preferences”, in the left list, jump to “Plug-in Development -> Target Platform”, click “Add”:



2. Select “Template”, in the drop-down list, and select “CICS TS V5.3 with Liberty and PHP”, click Next and then Finish to close the Target Definition window.



You will see:



3. **Click the "CICS TS V5.3 with Liberty and PHP" to active it**, and then click OK to exit the Preference window.

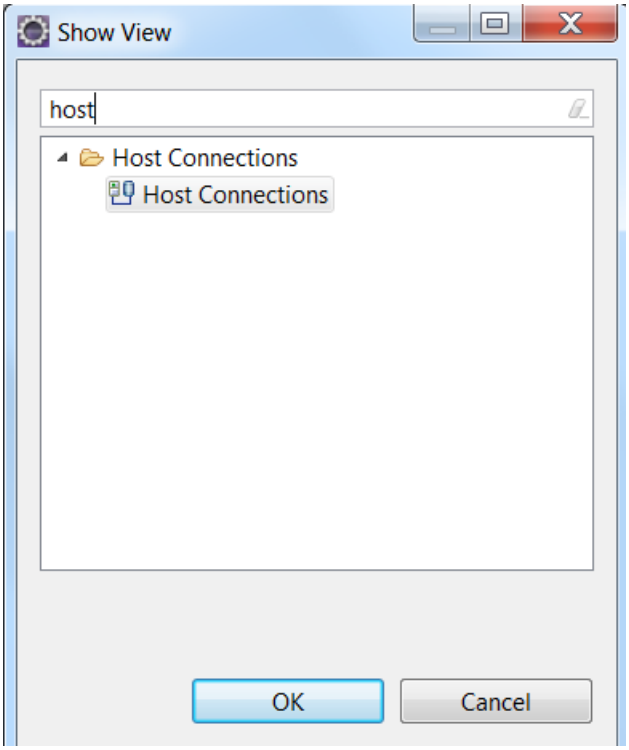
Appendix B: Set up connection between CICS Explorer and mainframe for the workshop

To support this workshop, you need to set up two types of connection for CICS Explorer:

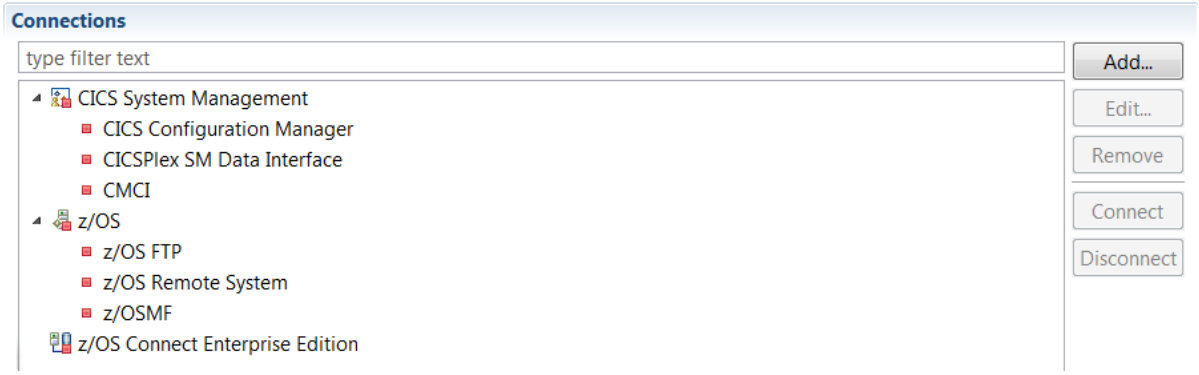
1. z/OS FTP connection
2. CMCI connection

Configure FTP connection to MVS

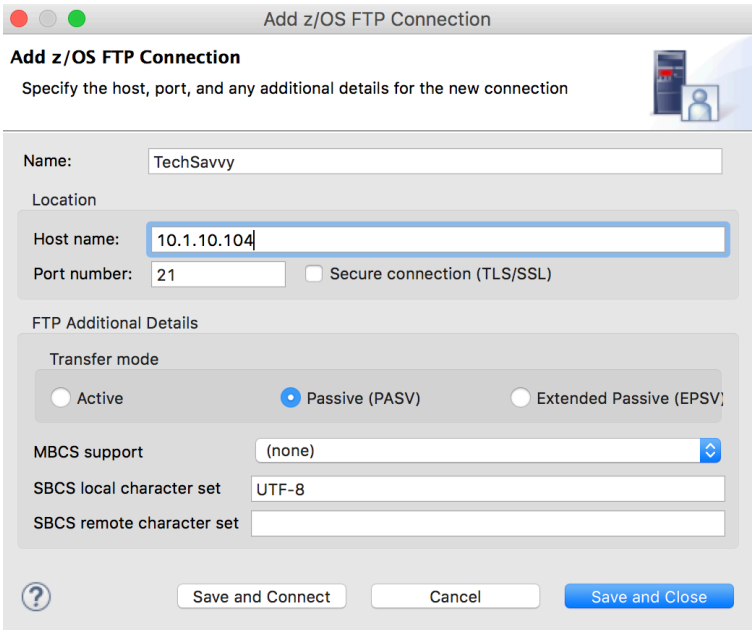
1. Here we need to establish a FTP connection to MVS so that we can easily deploy CICS bundles into MVS USS. In Eclipse, go to Window -> Show View -> Other -> Host Connections



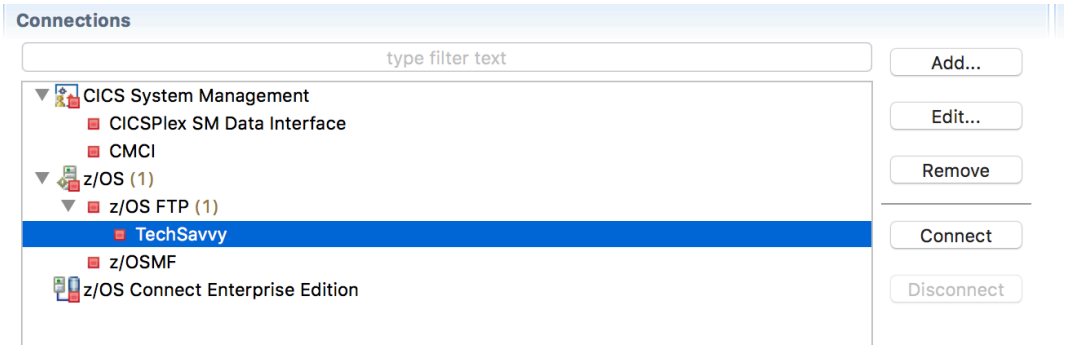
You will see the Host Connection view as below:



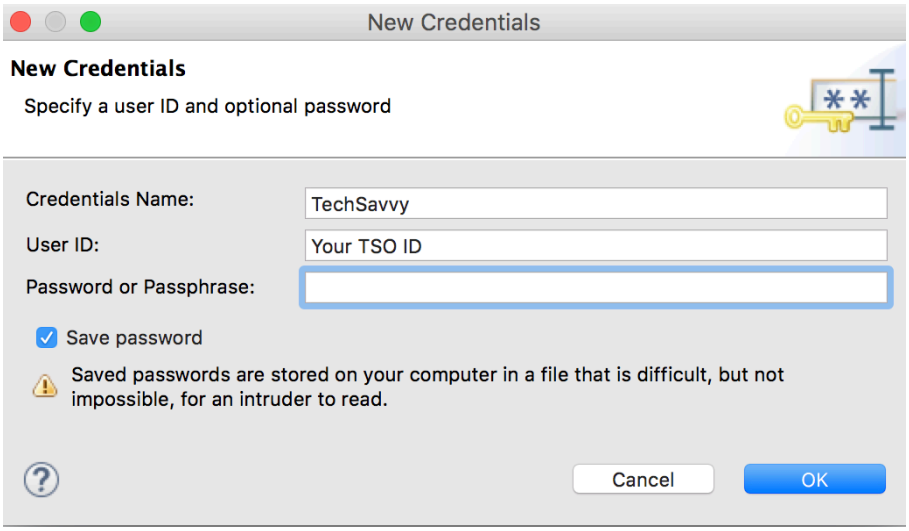
2. Select the z/OS FTP, and click the Add button to configure the z/OS FTP connection to *172.16.27.63*:



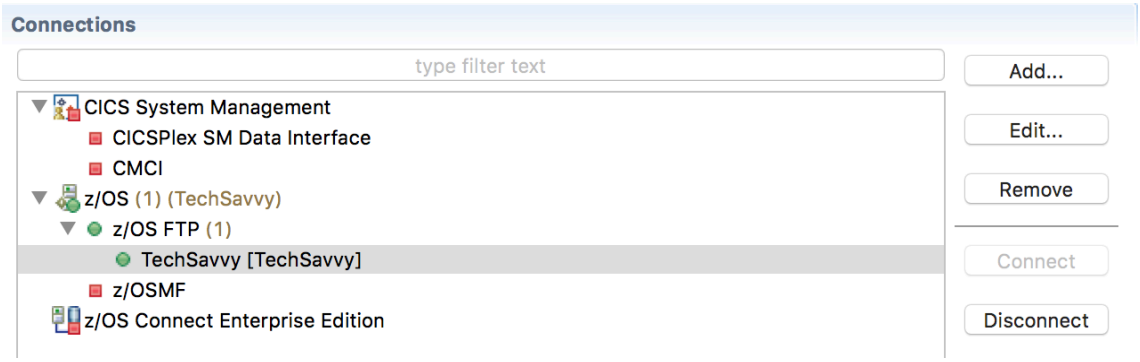
3. Click “Save and Close”, in the list you can see a z/OS FTP connection configuration is created:



4. In the Credentials panel, click Add button to add a credential to connect MVS. Input the assigned userid and password and then click **OK** to create the credential.

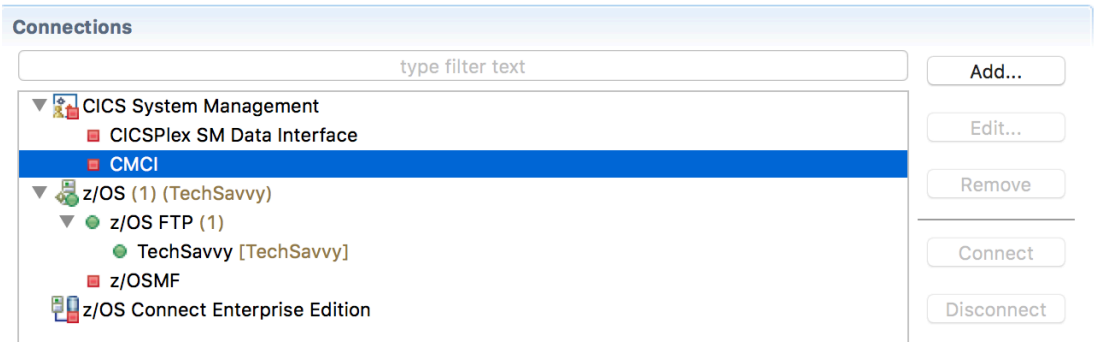


5. Connect to TechSavvy server using your credential. If successfully connected, you will see the TechSavvy connection turns to green and Connect button turns Gray:

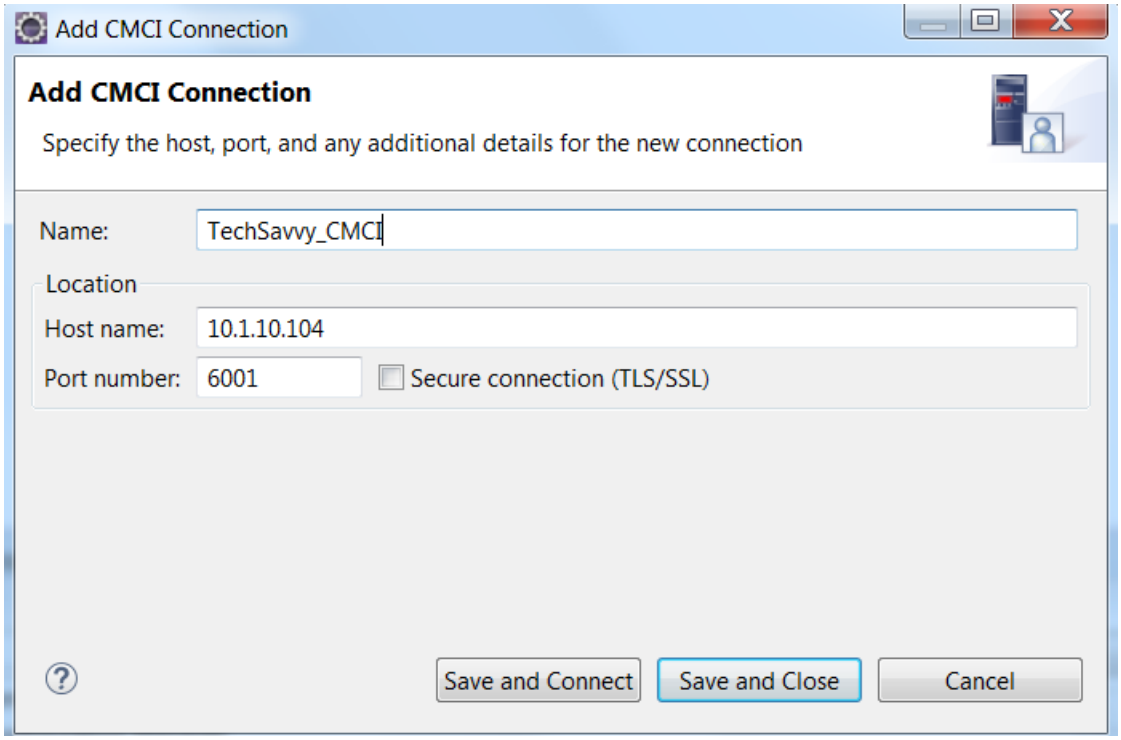


Configure CMCI connection to MVS

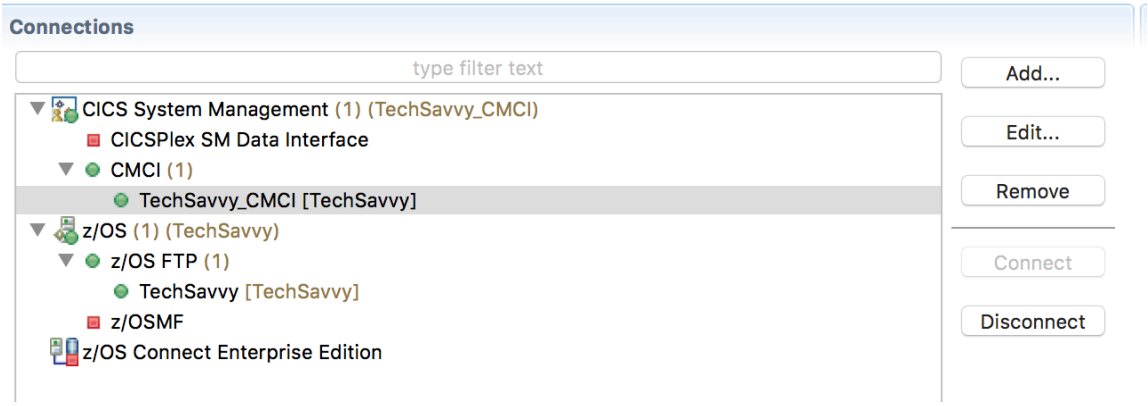
1. Select CMCI as connection type, and press Add.



2. Input Name, Host name and port number. This port number is CMCIPORT (6**1) assigned for your WUI region.



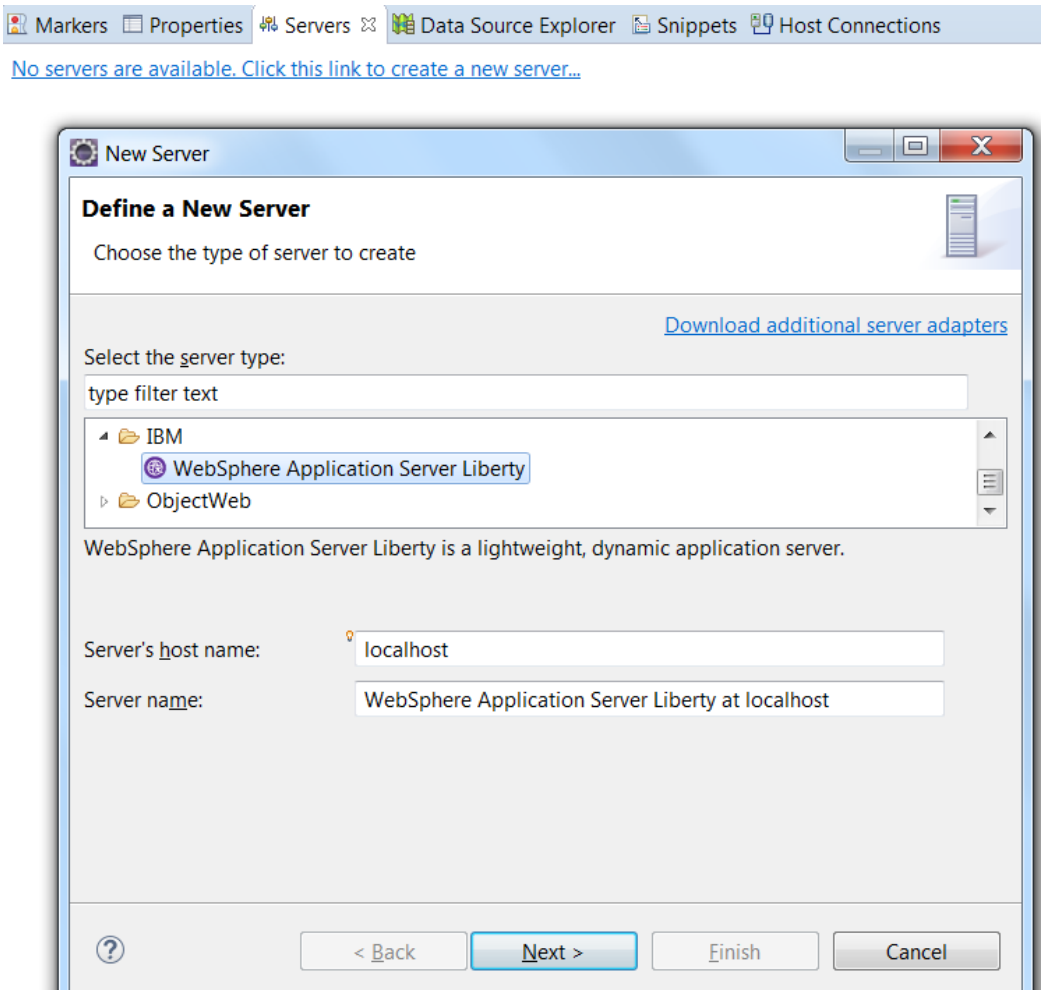
3. Save and connect using your own credential. You will be able to see the CMCI connection is configured successfully.



Appendix C: *Set up WebSphere Application Server Liberty and CICS Remote Development Feature for Java*

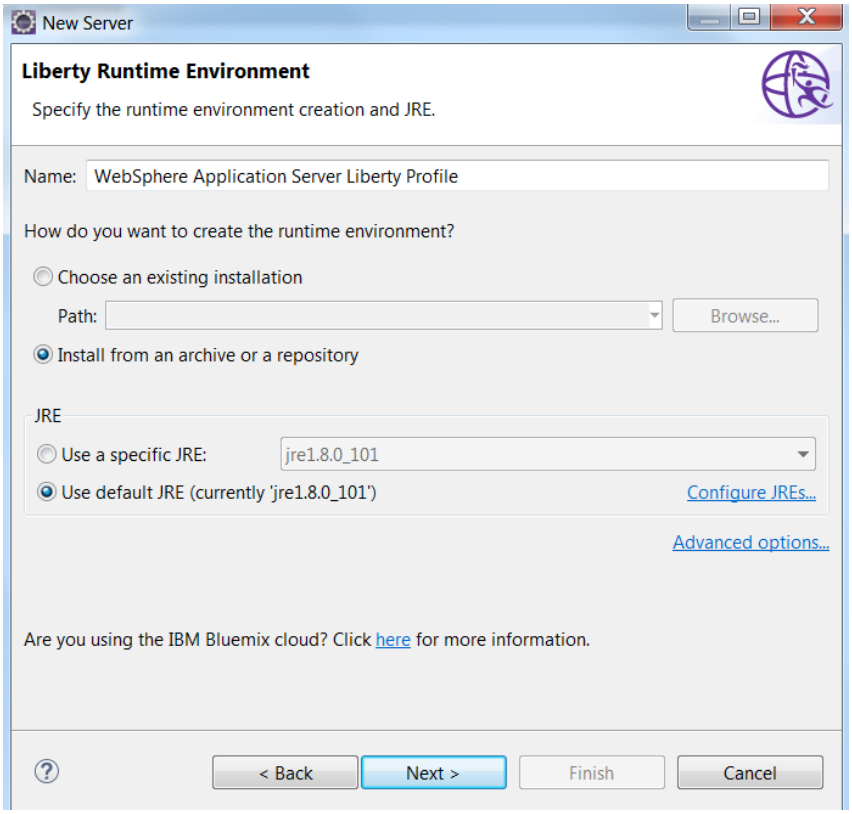
Setting up a local Liberty server

1. Open Window->Show View>Other… and search for Servers. Once you have opened the view, right click anywhere in the view and select New->Server, or click on the hyperlink as shown below.

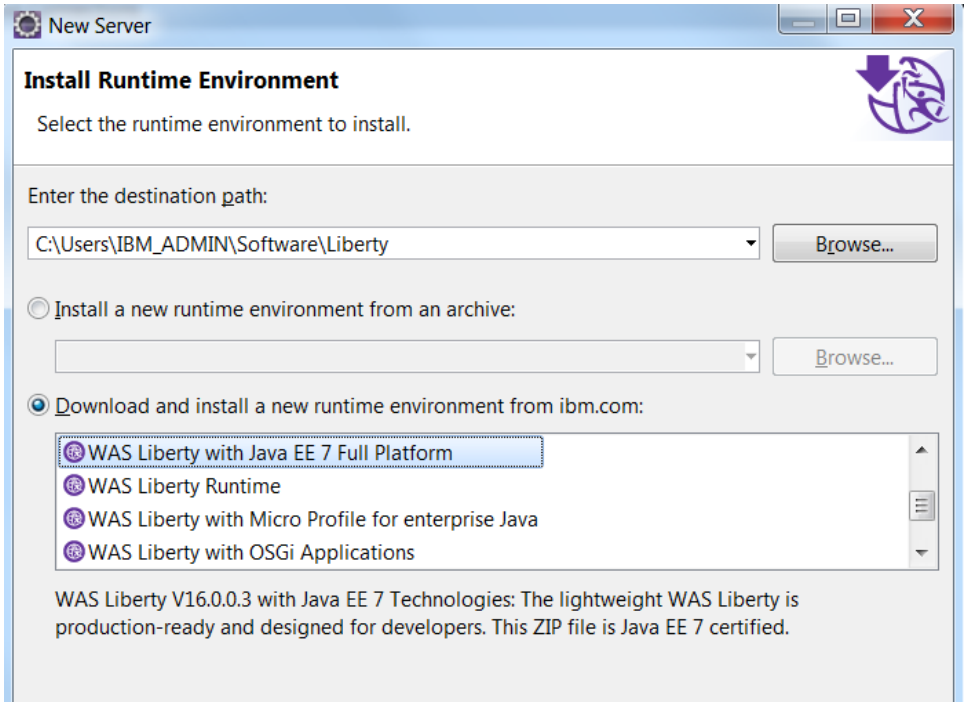


2. Select WebSphere Application Server Liberty as the server type and leave the other options as the defaults. When ready click Next >

3. You can keep most of the options on the following page the same, just select Install from an archive or a repository to download the runtime environment.



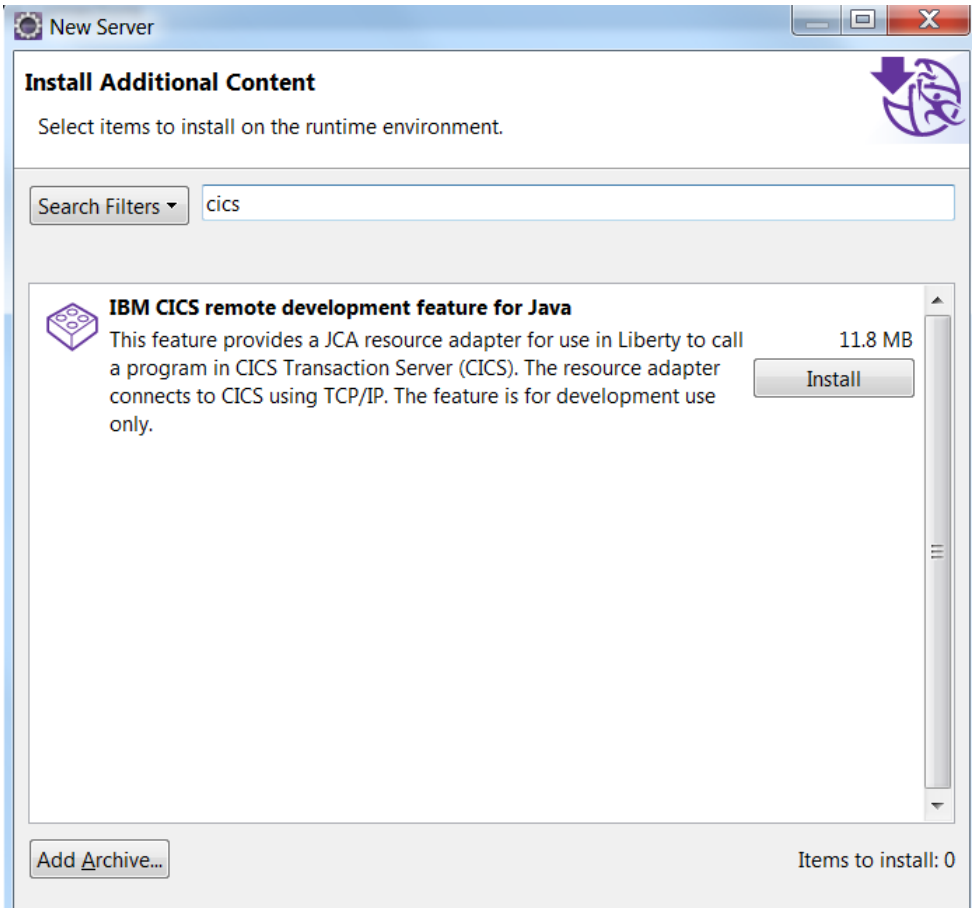
4. On the next page you enter a destination path for your new server on your workstation



5. Select the option to Download and install a new runtime environment from ibm.com and select whichever option suites your requirements, then select Next >

6. On the next page of the wizard you can select any additional features that you want to install. To use the IBM CICS remote development feature for Java, type CICS into the filter text box and select Install next to the feature, as shown

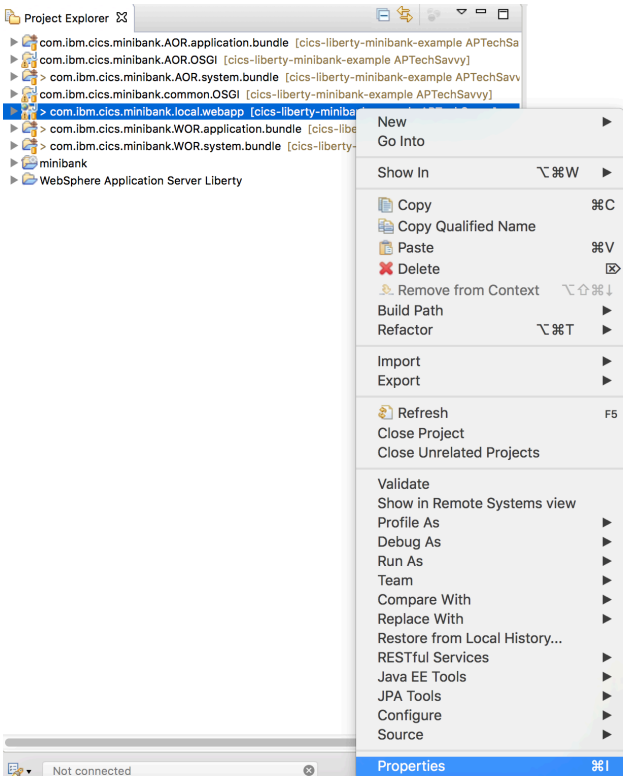
below:



7. All other additional features are optional and depend on what you want in your Liberty server. If you do not require any additional features, click Next through the wizard and finally click Finish.

Appendix D: *Resolve local web application error after downloading from Git*

You may see some errors in **com.ibm.cics.minibank.local.webapp** and it is probably because the project cannot find the runtime environment. To resolve the error, right click the project and click ***Properties***



Then select ***Project Facets***, then check ***Runtimes*** is correct. You need to make sure you specify your local liberty sever as the runtime, and it should be able to solve the issue. Sometimes, even the runtime is right, error may still remain. In this case, uncheck and recheck, because Eclipse is not very clever ☺