

# CICS Business Transaction Service and Java OSGi: Part 2 of 2

## Sheffield University COM3310 2017-2018 Assessed Course Project

Dr Malcolm Beattie, IBM UK

23 November 2016. Updated from COM3015 version of 17 March 2016 for Sheffield and Zeusplex.

24 November 2016. Add confirmed hand-in date and confirmed MOLE hand-in instructions.

10 March 2017. Updated for Surrey.

25 April 2017. Switch import order of zeusbank\_util & zeusbank\_java\_template. Add \_v3 to zeusbank\_util.tar.gz basename. Clarify use of Java BTS utility class.

15 November 2017. Updated for Sheffield 2017-2018.

15 November 2018. Updated for Sheffield 2018-2019. Updated with deadline and instructions to download the three files from MOLE.

## Objectives

This part of the project gets you using CICS Business Transaction Services (BTS), developing an Anti-Fraud Checking application in Java with Eclipse and using CICS Explorer to deploy it to a CICS Java Open Service Gateway Initiative (OSGi) platform.

## Anti-Fraud Checking

Most real banks have Anti-Fraud and Anti-Money-Laundering (AML) checks built into their systems. For Zeusbank, we will implement a very simplistic Anti-Fraud check: when a customer has an outgoing payment transaction of more than 1000, we trigger a business process that notifies the customer and asks them to confirm whether the payment is valid or whether they want it to be investigated. If the customer does not respond within a given time the business process automatically notifies a bank employee to investigate the transaction.

## CICS Business Transaction Services (BTS)

The manual describing CICS BTS is *CICS V5R3 Business Transaction Services* (SC34-7403-00) is available in PDF format in the usual z/OS Knowledge Center on the page

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.proddoc.doc/topics/PDF.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.proddoc.doc/topics/PDF.html) - the direct link is <http://publibfi.dhe.ibm.com/epubs/pdf/dfhp9h00.pdf> . I will use this version to give

documentation references below. A version of the content written in an online web page style is available in the same Knowledge Center at

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.ts.productoverview.doc/concepts/businesstransaction.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.productoverview.doc/concepts/businesstransaction.html) .

Chapters 1 and 2 (of the PDF manual) should give a sufficient overview of the technology and terminology. Chapter 3 gives an overview of the BTS API of which only we will only be using some of the basics for this project - we will not be needing subactivities (only the implicit root activity of our processes will run) nor composite events nor explicit syncpoints.

The API for programs using BTS is included in *CICS V5R3 Application Programming Reference* (SC34-7402-00) whose PDF version is available from the same page as the PDF version of the BTS manual above. Most Java programs use the JCICS API rather than the classic API but there are no JCICS classes for BTS so for Zeusbank there are wrappers exposing the classic API to Java which are made available via class BTS in package zeusbank\_util. This package is included in an Eclipse project along with its source whose comments indicate which Java methods map to which traditional BTS APIs. You use this API by constructing a BTS object near the start of your program and invoking methods on that object, e.g.

```
BTS bts = new BTS("LARGS");
```

```
...
```

```
bts.some_method(...);
```

You will be importing this project into your own Eclipse configuration but you do not need to bundle it or deploy it yourselves because it is already deployed and active within CICS. The Java APIs for CICS are introduced in *CICS V5R3 Java Applications in CICS* (SC34-7417-00). The Javadoc detailed documentation on the JCICS APIs are online in the same KnowledgeCenter at

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.ts.jcics.javadoc/overview-tree.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.jcics.javadoc/overview-tree.html) .

## BTS Repository

The repository which holds the current states and data of these business processes (process type "AFCHECK") is the VSAM dataset SHECICS.ZEUSBANK.BTSREPO defined to CICS as file ZBREPO. However, there is no need to access this dataset externally; only via the CICS BTS API and associated transactions and programs.

## The AFCHECK Process

When the Zeusbank ZBTXN program logs a transaction with an outgoing payment from a Zeusbank customer (Flag A) of 1000 or over, it

- defines an AFCHECK process named AFCnnnnnnnn (where nnnnnnnn is the transaction ID)
- places in it the following containers:
  - FROM\_SORTCODE holding the sort code of the payment source (6 EBCDIC characters)
  - FROM\_ACCTNUM holding the account number of the payment source (8 EBCDIC characters)
  - TO\_SORTCODE holding the sort code of the payment destination (6 EBCDIC characters)
  - TO\_ACCTNUM holding the account number of the payment destination (8 EBCDIC characters)
  - AMOUNT holding the amount of the payment (1-11 EBCDIC decimal digits)

These containers are in the *process* not the *root activity of the process*. They can be fetched using the `bts.get_process_container(container_name)` or `bts.get_acqprocess_container(container_name)` methods, as appropriate, and those methods will handle code page issues (EBCDIC v. UTF8 etc) and provide you with a Java String value in the expected way.

- starts the process with a "RUN ... ASYNCHRONOUS" which starts its associated transaction (named ZBAF).

Transaction ZBAF is defined to run the program ZBAF which simply looks at the last two digits of the transaction recipient's account number, nn, and transfers control (EXEC CICS XCTL) to program ZBAFPRnn.

### Important

When writing your programs for part 2 of the project, **DO NOT** use the Java `System.exit()` method. This terminates both the JVM server and the entire CICS region which affects all other students of the same university who are working on the project!

## Requirement 2.1: AFCHECK Process Program ZBAFPRnn

For this project, all student userids need to implement separate programs so the program started by transaction ZBAF (also called ZBAF) looks at the last two digits of the transaction recipient's account number, nn, and transfers control (EXEC CICS XCTL) to program ZBAFPRnn. Each of those program names (for nn from 01 to 40) is defined with attributes

```
JVM(YES) JVMSERVER(ZEUJVMOS) JVMCLASS(zeusbank.she00nn.AFCheck)
```

and so runs the `main()` method of the OSGi service or Java class `zeusbank.she00nn.AFCheck`. This program needs to progress the process:

- The first time it runs (event DFHINITIAL) it needs to place a random 8-digit numeric token, rtok, in a container of the process (call it TOKEN) and notify the customer of the payment transaction in question and tell them a URL where they can go to say whether they want the payment to be investigated or not. The URL should be of the form  
`https://zeuszos.edu.ihost.com:8005/afcheck/customer/she00nn?proc=AFCnnnnnnnnn&token=rtok`  
The notification would in reality send an email or SMS but for us it can simply be a logged message. The program should also set a timeout which in a real situation may perhaps be a few days or so but for us can be 30 seconds or a few minutes to simplify testing and debugging. The program should also define an input event for the customer's use of the web form to trigger.
- If the timeout triggers and the customer has not responded, do not let them respond any more, notify

the teller (i.e. yourself) with a log message of the payment which needs investigating along with a URL to go to once the investigation is complete. Define an input event for the teller's use of their web form to trigger.

- If the customer responds, write the yes/no response to a container in the process. If they have said an investigation is not needed, end the process. If they have said an investigation is needed, notify the teller immediately in the same way as you did for the customer (and remove the timeout).
- If the teller responds saying that the investigation is complete, end the process.

## Requirement 2.2: Web Form Handling for Customer

There is a URIMAP defined to CICS which maps URLs with paths of the form /afcheck/customer/\* to run program ZBAFWC. This program looks at the last two digits of the path (which should be used in the form /afcheck/customer/she00nn) and transfers control to program ZBAFWCnn. Each of those program names (for nn from 01 to 40) is defined with attributes

JVM(YES) JVMSERVER(ZEUJVMOS) JVMCLASS(zeusbank.she00nn.CustomerForm)

and so runs the main() method of the OSGi service or Java class zeusbank.she00nn.CustomerForm . This should be a web application that uses the JCICS API to handle the customer's response by finding, verifying the token for, acquiring and running the appropriate AFCHECK business process with a "customer response" event. In reality, the customer would not need to sign in to the web application with a teller userid (that is why the random token is needed to verify the person filling in the form was the one notified about the payment); however, CICS is configured on Zeus always to require a login so use your ordinary z/OS userid to do so but there is no need to make use of that userid or do other security checks in your CustomerForm program.

## Requirement 2.3: Web Form Handling for Teller

There is a URIMAP defined to CICS which maps URLs with paths of the form /afcheck/teller/\* to run program ZBAFWC. This program looks at the last two digits of the path (which should be used in the form /afcheck/teller/she00nn) and transfers control to program ZBAFWTnn. Each of those program names (for nn from 01 to 40) is defined with attributes

JVM(YES) JVMSERVER(ZEUJVMOS) JVMCLASS(zeusbank.she00nn.TellerForm)

and so runs the main() method of the OSGi service or Java class zeusbank.she00nn.TellerForm . This should be a web application that uses the JCICS API to handle the teller's response by finding, acquiring and running the appropriate AFCHECK business process with a "teller response" event. There is no need to use a token to validate the teller - they are assumed to be logged in and trusted.

## Java APIs for web applications

Rather than introduce the Java HTTPServlet APIs which need more complex application deployment, this project makes the basic CICS Java APIs available for web applications which are described in the *Java Applications in CICS* manual.

## z/OS UNIX ssh access to Zeusplex

To get an interactive terminal session to z/OS UNIX on Zeusplex, use an ssh client (such as ssh or putty) to connect to port 2220 (for ZOS22A) or port 2221 (for ZOS22B) of host zeuszos.edu.ihost.com . Logging into either will provide the same view of the filesystems you need. Do *not* mistakenly use the default ssh port of 22 and do *not* use the zeus.edu.ihost.com gateway guest as the target hostname.

## Logged output from the ZEIJVMOS JVM used by CICSZ032

Output written from Java to System.out (which includes output written to t.out when "t" is a non-3270 task in CICS) is appended to a log file in the Zeusplex UNIX filesystem with a name of the form

/u/SHECICS/CICSZ032/CICSZ032/ZEIJVMOS/Dyyyymmdd.Thhmmss.dfhjvmout

where the (D)atestamp and (T)imestamp parts depend on when the JVM started up. When it starts up, CICS creates a symbolic link named CURRENT.STDOUT in the same directory pointing to the latest one and that may be a more convenient name for you to use when looking at your recent log messages.

You can use this mechanism freely to write progress and debugging messages as you develop and troubleshoot your programs. ***Please prefix every line that you write to this log with your z/OS userid so that everyone can tell apart which are their own messages!***

For requirements which need you to notify the customer or teller, just write an appropriate message to this log which includes all the information needed and the exact content that you want to tell them in a format such as:

she00nn: Notify customer (sortcode, accountnumber): content\_of\_message

or

she00nn: Notify teller: content\_of\_message

## Logged output from CICS

The transactions and programs involved also write telemetry and debugging information which may be useful for debugging. These appear in the sysout named CEEOUT of the CICS region (address space name CICSZ032). You can use SDSF to view this sysout and, while doing so, hitting Enter will update the view immediately with any new output that has appeared.

## Debugging Processes

You can use the CBAM transaction (an interactive 3270 transaction) to browse through the existing processes and look at their events, container names (though not container contents) and state. This is documented in the CICS Business Transaction Services manual. Note that the containers used for this project are at *process* level, not *activity* level so to see them, tab to the right-hand "Cont" column of the process and hit Enter rather than hitting Enter on the left to see the activities (just DFHROOT in our case) of the process.

## Setup on z/OS

Ensure you have logged into z/S UNIX at least once so that a SHE00nn.OMVS.ZEUSPLEX.ZFS dataset has been auto-allocated and formatted to containing your home filesystem. ssh into Zeusplex and ensure your home directory is world-readable and world-executable so that CICS can access the bundle you will be exporting there. You can do this from a command line (e.g. via ssh) with the command

```
chmod a+rx ~
```

or you can use z/OS Explorer if you prefer.

## Setup of Eclipse

Download the following three files from MOLE to your desktop - you will be importing them into Eclipse shortly:

- zeusbank\_util\_v3.tar.gz
- zeusbank\_java\_template.tar.gz
- zeusbank\_bundle\_template.tar.gz

From your Eclipse instance that has the necessary IBM Explorer for z/OS plugins installed:

(1) Set up CICS as the Target Platform for your Eclipse development as follows:

Window > Preferences...

Choose Plug-in Development | Target Platform

If "CICS TS V5.1 Runtime" does not appear in the resulting list

- click Add...
- In the resulting dialog box, choose radio button "Template" and from the resulting drop-down menu choose "CICS TS V5.3" and click Next>.
- Click Finish to return to the Target Platform dialog box, this time with a checkbox for CICS TS 5.3 listed in addition to any other existing targets

Tick the checkbox for "CICS TS V5.3" and click OK.

If you need to use your Eclipse instance to develop programs for other targets you may need to revisit this dialog box to change to other targets - just remember to switch back to CICS when working on this project. If you forget to set this Target Platform then any attempted build of this project will result in errors complaining that "com.ibm.cics.server" classes cannot be found.

(2) Create the zeusbank\_util Java project, imported as follows:

File > Import...

Choose General | Existing Projects into Workspace and click Next>.

Select radio button "Select archive file".

Click Browse... and find the zeusbank\_util\_v3.tar.gz file you downloaded from MOLE.

Click Finish to import the project named zeusbank\_util which will appear in Package Explorer.

You do not need to edit or modify this project since it simply provides some wrapper code for using the BTS API from Java. You may well find it useful through to browse its src/zeusbank.util/BTS.java file where the comments preceding each public method of the BTS class indicates how the methods map to the documents "EXEC CICS ..." BTS API functions.

(3) Create a new Java project, imported as follows:

File > Import...

Choose General | Existing Projects into Workspace and click Next>.

Select radio button "Select archive file".

Click Browse... and find the zeusbank\_java\_template.tar.gz file you downloaded from MOLE.

Click Finish to import the project named zeusbank\_java\_template which will appear in Package Explorer.

In Package Explorer, right-click on zeusbank\_java\_template and choose Refactor > Rename... .

Enter name zeusbank\_java\_she00nn being sure to *replace nn with the last two digits of your z/OS userid*.

This renames the project to zeusbank\_java\_she00nn.

In Package Explorer, open the twisties of zeusbank\_java\_she00nn to find directory src/zeusbank.abc9999 .

Right-click on zeusbank.abc9999, choose Refactor > Rename...

Tick the checkboxes for "Update references" and "Update textual occurrences in comments and strings (forces preview)" and fill in the "New name" field with zeusbank.she00nn .

Click Preview>.

Ignore the two "Type zeusbank.abc9999 ... contains a main method ..." warnings - there should be no others.

Click Continue.

In the resulting "Rename package" dialog box, click OK.

The package directory now appears as zeusbank.she00nn in Package Explorer.

Open the twisty for zeusbank.she00nn (the subdirectory of src) and for each of the three Java source files AFCheck.java, CustomerForm.java and TellerForm.java, double-click to open them in the editor, click in the resulting file contents tab to focus on it and then use Edit > Find/Replace... to **Replace All** instances of abc9999 with she00nn .

In Package Explorer, find and open META-INF/MANIFEST.MF . In the editing panel, click on the MANIFEST.MF tab (in the list of tabs across the *bottom* of the editing panel). The contents are:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ABC9999 Zeusbank Bundle
Bundle-SymbolicName: zeus.>>>put_userid_here<<<.zeusbank
Bundle-Version: 1.0.0
Bundle-RequiredExecutionEnvironment: JavaSE-1.7
Import-Package: com.ibm.cics.server;version="[1.0.0,2.0.0)",zeusbank.util
CICS-MainClass: zeusbank.abc9999.AFCheck, zeusbank.abc9999.CustomerForm, zeusbank.abc9999.TellerForm
```

Change the one instance of ABC9999 in the Bundle-Name value to SHE00nn (upper case).

Change the one instance of >>>put\_userid\_here<<< in the Bundle-SymbolicName value to she00nn (lower case). Ensure you remove all the ">" and "<" characters too so that you leave the resulting final value as "zeus.she00nn.zeusbank" (without the quotes).

Change the three instances of abc9999 in the CICS-MainClass value to she00nn (lower case).

If you have not done so already, save all the files that you have edited. At this point the "Problems" panel should only show some warnings about unused packages and variables (since the source code in the template is minimal) and should show no errors.

(4) Create a new CICS bundle project, imported as follows:

File > Import...

Choose General | Existing Projects into Workspace and click Next>.

Select radio button "Select archive file".

Click Browse... and find the zeusbank\_bundle\_template.tar.gz file you downloaded from MOLE.

Click Finish to import the project named zeusbank\_bundle\_template which will appear in Package Explorer.

In Package Explorer, right-click on zeusbank\_bundle\_template and choose Refactor > Rename... .

Enter name zeusbank\_bundle\_she00nn being sure to *replace nn with the last two digits of your z/OS userid*

and click OK. This renames the project to `zeusbank_bundle_she00nn` but sometimes (not always) Eclipse or one of its plugins then gets upset and pops up a dialog box saying that the rename failed part way through. If this happens (the Details>> button will show a Java Null dereference error) then use the following workaround:

There will be buttons to Undo or Abort. Click Abort. The Package Explorer will show the renamed `zeusbank_bundle_she00nn` correctly but something in Eclipse is still caching the old information and attempts to build or compile may fail spuriously. There may be other ways to force Eclipse to drop the incorrectly cached information (Project → Clean All... perhaps) but the following is known to work:

In Package Explorer, right-click on the "`zeusbank_bundle_she00nn`" project name and choose "Export..." (not "Export Bundle Project to z/OS UNIX File System...").

Choose General | Archive File and click Next>

Ensure only the `zeusbank_bundle_she00nn` project name is checked on the left (along with its contents on the right).

In the "To archive file: ..." field, enter (or use the Browse... button) to choose a non-existent filename on your local filesystem (e.g. `tempbundle.tar.gz`) in which you are going to create (temporarily) an exported tarball of this package.

Click Finish to create the exported package file.

Now delete the package and all its files from Eclipse: right-click on the "`zeusbank_bundle_she00nn`" project name and choose Delete.

Tick the checkbox for "Delete project contents on disk (cannot be undone)" and click OK.

Now import from that exported package file you've just created in the same way you imported the original `zeusbank_bundle_template.tar.gz` file (i.e. File -> Import, General | Existing Projects into Workspace etc).

Now we continue with the main setup instructions, regardless of whether you needed the above export/import workaround for the bundle project rename.

In Package Explorer, open up the twisty for the now-called `zeusbank_bundle_she00nn` project and double-click on file `zeusbank.osgibundle` to edit it. It contains:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<osgibundle symbolicname="zeus.abc9999.zeusbank" version="1.0.0" jvmserver="ZEUJVM05" />
```

In the symbolicname value, change `abc9999` to `she00nn`.

In the META-INF subdirectory of `zeusbank_bundle_she00nn`, double-click on the `cics.xml` file to open it. In the General Information section, change the Version field from 0.0.0 to 1.0.0.

From the menu bar, do File -> Save All (or use the corresponding shortcut key, Shift-Ctrl-S).

At this point the "Problems" panel should show no errors and no additional warnings beyond the ones present from code in the Java project.

## Setup in CICS

***Note that for this project, you have been given RACF authority to define, alter, delete and install ("deploy") any resource in the CICSZ032 region so be careful when using CEDA or CICS Explorer to do so since you can affect your fellow users of the region.***

Use CEDA (or the equivalent CICS Explorer functionality if you prefer) to define the following resources to CICS, **replacing each `nn` with the last two digits of your userid**. Note that, by default, the arguments you type directly to each "CEDA ..." command will be uppercased by CICS before the program sees them (which will cause problems with the case-sensitive Java classnames) so you can either

- leave those off the command you type (e.g. just type "CEDA" or "CEDA" and the first few arguments) and hit Enter. You can then overwrite the case-sensitive attributes into the on-screen fields where case sensitivity is respected (in fields marked "Mixed case"); or else
- Before invoking CEDA, enter the command `CEOT TR` which will set your terminal (for the duration of your CICS logon session) so that CICS will only upper-case the first word (the transaction name) of your initial input and not the rest.

```
CEDA DEFINE PROGRAM(ZBAFPRnn)
GROUP(SHE00nn)
EXECKEY(CICS) CONCURRENCY(REQUIRED)
JVM(YES) JVMSERVER(ZEUJVM05)
```

```
JVMCLASS(zeusbank.she00nn.AFCheck)
```

```
CEDA DEFINE PROGRAM(ZBAFWCnn)
  GROUP(SHE00nn)
  EXECKEY(CICS) CONCURRENCY(REQUIRED)
  JVM(YES) JVMSERVER(ZEUJVM0S)
  JVMCLASS(zeusbank.she00nn.CustomerForm)
```

```
CEDA DEFINE PROGRAM(ZBAFWTnn)
  GROUP(SHE00nn)
  EXECKEY(CICS) CONCURRENCY(REQUIRED)
  JVM(YES) JVMSERVER(ZEUJVM0S)
  JVMCLASS(zeusbank.she00nn.TellerForm)
```

```
CEDA DEFINE BUNDLE(ZBAFBUnn)
  GROUP(SHE00nn)
  DESC(ZEUSBANK ANTI-FRAUD BUNDLE FOR SHE00nn)
  BUNDLEDIR(/u/SHE00nn/zeusbank_bundle_she00nn_1.0.0)
```

Now install those resources into the active CICS with

```
CEDA INSTALL GROUP(SHE00nn)
```

or the CICS Explorer equivalent. You could also install them one by one by expanding your group with

```
CEDA EXPAND GROUP(SHE00nn)
```

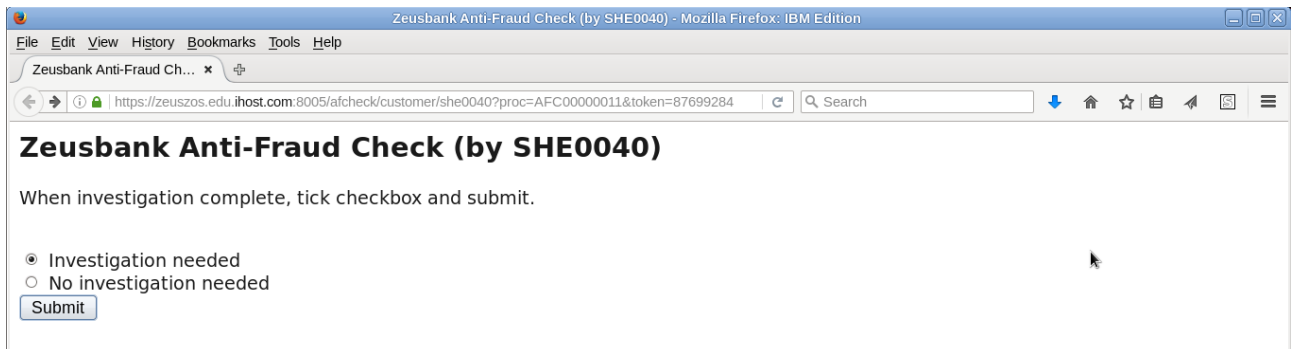
and using the “I” (letter “i” for “Install”) command against the appropriate line in the expanded list.

You will find that the ZBAFU<sub>nn</sub> bundle will not install successfully since you have not yet exported a bundle of your code to the specified directory. As you work on the project below, you will discard/reinstall (not define) that bundle each time you want to install/deploy and test a version of your application.

## Working on the project

Create, edit and rebuild the zeusbank\_java\_she00<sub>nn</sub> code as needed to follow the three Requirement paragraphs above.

You are aiming to get the CustomerForm web page to look something like the following:



The screenshot shows a web browser window titled "Zeusbank Anti-Fraud Check (by SHE0040) - Mozilla Firefox: IBM Edition". The address bar shows the URL "https://zeuszos.edu.ihost.com:8005/afcheck/customer/she0040?proc=AFC00000011&token=87699284". The page content includes the title "Zeusbank Anti-Fraud Check (by SHE0040)", a instruction "When investigation complete, tick checkbox and submit.", and two radio buttons: "Investigation needed" (selected) and "No investigation needed". A "Submit" button is located below the radio buttons.

The TellerForm web page should look something like the following:

Zeusbank Anti-Fraud Check - Teller form (by SHE0040)

AFCHECK process id:

Investigation complete: ☐

The AFCCheck program should progress the process as described in the "AFCCheck Process" requirement.

Once your code builds successfully you need to export the bundle to z/OS for deployment.

### Java, OSGi and CICS dependencies relating to the Eclipse projects

Eclipse will only let you export the bundle to z/OS when all the necessary OSGi bundle parts are available. For us, that means a bundle part with symbolic name "zeus.she00nn.zeus" as listed in zeusbank\_bundle\_she00nn's zeusbank.osgibundle file. For your AFCheck program, that bundle part is declared (via a "Bundle-SymbolicName: zeus.she00nn.zeusbank" line and mapped to CICS-MainClass zeusbank.she00nn.AFCheck) in the META-INF/MANIFEST.MF file of the zeusbank\_java project. That means your zeusbank\_java\_she00nn needs to be compiled and built successfully before you can export (and hence install/deploy) the zeusbank\_bundle\_she00nn. Eclipse will automatically update the status of zeusbank\_bundle\_she00nn when the zeusbank\_java\_she00nn project successfully compiles/builds. You can tell whether the bundle project is happy with the dependent Java project by opening the twisties in Package Explorer to show the icon for the zeusbank\_bundle\_she00nn/zeusbank.osgibundle file. If the icon has a cross overlaying the left-hand side then there is a problem with the Java project/code which will prevent the bundle from being exported and thus deployed.

In the CICS SM perspective of Explorer, look at the Host Connections tab and ensure you have an active CICS Explorer CMCI connection to the ZEUSCICS CICSplex on Zeusplex and an active z/OS FTP connection to Zeusplex. Make a note of the connection names you have given to them.

Back in the Java perspective, go to the "Package Explorer" panel, right-click on zeusbank\_bundle\_she00nn (not zeusbank\_java\_she00nn) and choose "Export Bundle Project to z/OS UNIX File System...". Choose radio button "Export to a specific location in the file system." and click Next>. For the "Connection" dropdown, ensure your active Zeusplex connection is selected. For the "Parent Directory", enter your z/OS Unix home directory which is of the form "/u/SHE00nn" (replacing SHE00nn with the *upper* case version of your z/OS userid). For "Bundle Directory", verify that it then auto-fills-in itself as /u/SHE00nn/zeusbank\_bundle\_she00nn\_1.0.0 or else manually change it to that. (This value needs to match the BUNDLEDIR attribute you set for the ZBAFBUnn CICS BUNDLE resource that you created earlier.) Double-check that the "Bundle Directory" field does not display your actual home directory or the name of any other non-empty directory whose contents you want to keep because the bundle upload process is about to remove the directory and all its contents before recreating it. Tick the "Clear existing contents of Bundle directory" checkbox then click Finish to upload the bundle.

In CICS Explorer (i.e. via the CICS SM Perspective of Eclipse), go to the CICSplex Explorer tab of the panel on the left hand side of the screen and open the twisty by the (one and only) CICSplex labelled ZEUSCICS. This should show an entry for each of the CICS regions on Zeusplex that is an active part of the CICSplex. Click on the CICSZ032 region to select it as the context for your operations. When using CICS Explorer for this project, always ensure that the CICSZ032 region is highlighted in this CICSplex Explorer panel so that you will be operating on the right CICS region. With the region highlighted, from the menu bar select Definitions -> Bundle Definitions to display or switch to the "Bundle Definitions" tab in the main panel. This tab shows the bundles defined to the region. Note: do *not* use the Operations -> Bundles view instead (or the corresponding tab titled simply "Bundles" because this displays only the bundles which are actually installed (a.k.a. deployed).

You now need to install or re-install bundle ZBAFBUnn where nn are the two final digits of your z/OS userid.



If you have installed the bundle from CICS before then you need to disable and discard it before re-installing. To do this, go to the Operations -> Bundles tab and click the "refresh" icon (yellow pair of circulating arrows) if your ZBAFBUnn does not already appear.

Right-click ZBAFBUnn and select Disable. Click OK in the resulting confirmation dialog box. Then right-click the now-disabled-but-still-installed ZBAFBUnn entry and select Discard.

Click OK in the resulting confirmation dialog box.

If you are trying to discard the bundle and you have forgotten to disable it first, you will get an error with RESP(INVREQ) and RESP2(5) - in that case, go back and Disable the bundle first.

Now to install the bundle, right-click on bundle definition ZBAFBUnn and choose Install... .

In the resulting "Perform INSTALL Operation" dialog box, click to select the (one and only shown) region, CICSZ032 (with dropdown at its default of ZEUSCICS), and click OK.

If you have forgotten to disable or discard the bundle and it is already installed then the new attempted installation will fail with an error indicating RESP(INVREQ) and RESP(612).

If you have forgotten to ensure the permissions on the exported bundle files allow CICS to access them then the installation will fail with error RESP(INVREQ) and RESP2(628).

## Testing your application

Once you have installed/deployed your application bundle, you can test it out by making a payment using the ZBTP 3270 transaction as described in Part 1 of the project. From an ssh session to Zeusplex you can do

```
tail -f /u/SHECICS/CICSZ032/CICSZ032/ZEUJVM05/CURRENT.STDOUT
```

to watch the notification and debugging output appear "live" that you write to System.out. Use Ctrl-C to exit from the tail command.

Use your browser to verify the behaviour of at least the following situations:

- Customer uses web page to respond before timeout and asks for investigation
- Customer uses web page to respond before timeout and asks for no investigation
- Customer does not respond before the timeout happens
- Customer attempts to respond after timeout and is refused
- Teller responds

You can use transaction CBAM and hit Enter against the AFCHECK processtype (the one and only row) to see still-present business processes AFCnnnnnnnnn . Once the customer has responded that investigation is not needed or the teller has marked the investigation as complete, your code should have caused the business process to finish and disappear. If your code is not yet fully working and the business process stays around (perhaps because your program abends), please use transaction BTCA to cancel it:

BTCA AFCHECK AFCnnnnnnnnn

This will avoid leaving old business processes around clogging up the repository where others will have to scroll through an increasingly long list in CBAM to troubleshoot their own ones.

## Part 2 Deliverables

You need to export your Eclipse zeusbank\_java project (*not* the zeusbank\_bundle or zeusbank\_util projects) as a compressed tar file to send to me. To do this, select zeusbank\_java in the Eclipse "Project Explorer" panel, right-click and choose Export. In the resulting dialog box, choose General | Archive File and click Next>. Ensure zeusbank\_java (and only zeusbank\_java) is ticked in the checkboxes on the left-hand panel (and all its content files on the right-hand panel will thus be ticked too). In the Options section, choose the "Save in tar format" radio button (*not* the "Save in zip format" one) and ensure the "Compress the contents of the file" checkbox is ticked. Fill in a location and filename on your local filesystem in the "To archive file" field and click Finish. This will export a tar.gz file of your project.

What I need from you is

1. An email which includes
  - your name
  - your University Registration number

- your Zeusplex userid of the form SHEnnnn
  - An attachment of that .tar.gz file exported from your zeusbank\_java Eclipse project.
2. Unless there are special circumstances, this email should be sent to me at [beattiem@uk.ibm.com](mailto:beattiem@uk.ibm.com)
  3. Upload the same information from (1) to MOLE in the assessment dropbox for this module
  4. Leave the resources (bundles, programs etc) on CICS and do not alter them after submission.

## Deadline

The deadline for submission is 17:00 on Monday 10 December 2018.

## Documentation

The z/OS 2.2 base manuals and books are all available in PDF form from the Knowledge Center at

<http://www-03.ibm.com/systems/z/os/zos/library/bkserv/v2r2pdf/>

or in a form intended for online reading in another part of the KnowledgeCenter at

[http://www.ibm.com/support/knowledgecenter/SSLTBW\\_2.2.0/com.ibm.zos.v2r2/en/homepage.html](http://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2/en/homepage.html)

The complete set of CICS 5.3 documentation is available online in the KnowledgeCenter for online browsing at

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.ts.home.doc/welcomePage/welcomePage.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.home.doc/welcomePage/welcomePage.html)

and can be downloaded in PDF form from

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.proddoc.doc/topics/PDF.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.proddoc.doc/topics/PDF.html)

For detailed explanations of error response codes (RESP and RESP2) generated by the EXEC CICS APIs (as logged in error messages from the zeusbank\_util Java BTS wrapper classes, for example), see the CICS Application Programming Reference manual available in the set above or via the following direct link to the PDF manual: <http://publibfi.dhe.ibm.com/epubs/pdf/dfhp4h00.pdf>

The detailed Javadoc JCICS API documentation is available online at

[https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.3.0/com.ibm.cics.ts.jcics.javadoc/overview-tree.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.jcics.javadoc/overview-tree.html)

but not available for PDF download.