CICS, VSAM, ICETOOL/DFSORT and Worklight: Part 1 of 2

Sheffield University COM3310 2014-2015 Assessed Course Project

Dr Malcolm Beattie, IBM UK

30 October 2014. Rebased on COM3015 version from 14 March 2014. Changes for SHE.
6 November 2014. Confirmed submission deadlines. Updated submission process to include MOLE.

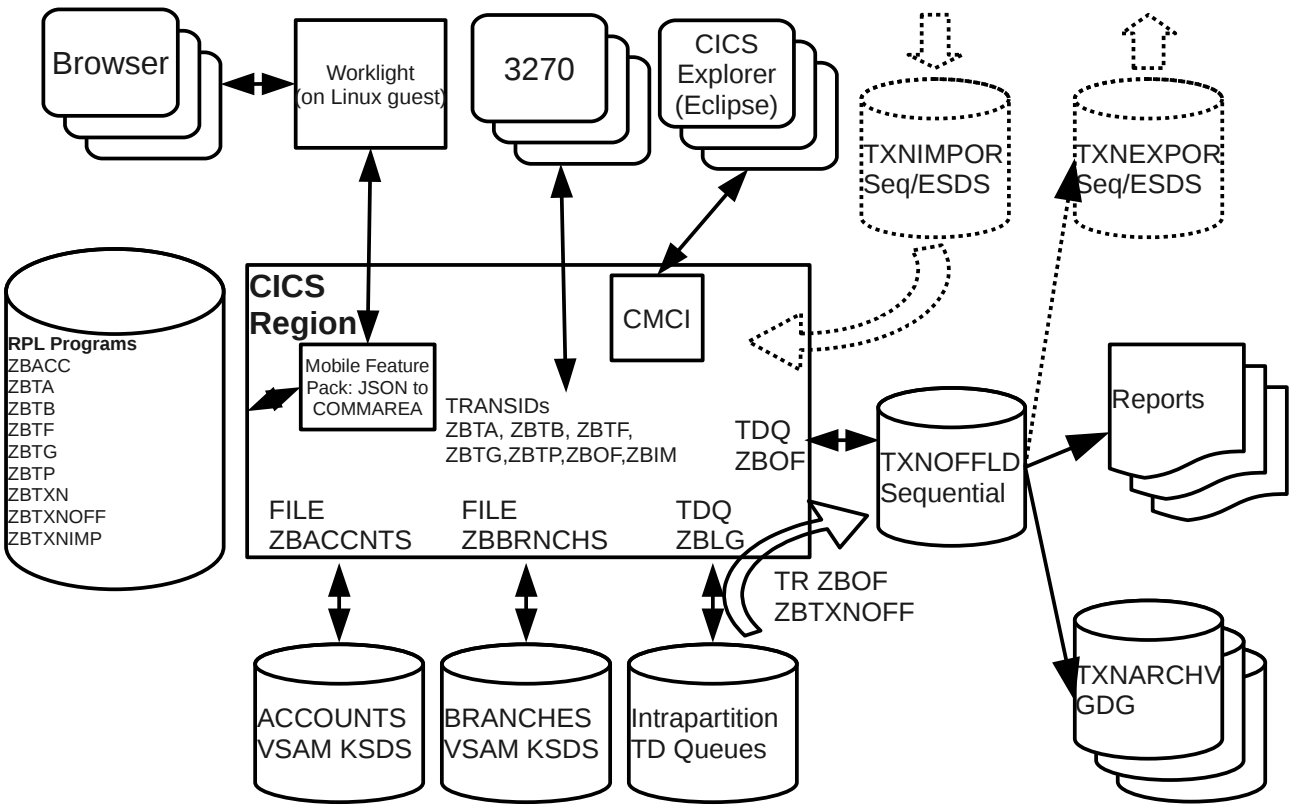
Objectives

This project gets you using CICS, VSAM datasets, ICETOOL/DFSORT and some JCL to provide some reporting and transaction processing for ZeusBank, a simplistic CICS/VSAM-based bank account environment. In the second part of the project you will be using the CICS Mobile Feature Pack and IBM Worklight to create a mobile web front-end to a ZeusBank “voucher” service.

Situation

The Sheffield CICS region on Zeus ZOS113 system has been set up with a (very simplistic) model of “ZeusBank”, a bank environment that holds and processes bank branch and account information. Programs and transactions are provided which allow for creation of accounts and branches, for payments between accounts belonging to ZeusBank or other banks and for offload and import of transaction data.

Architecture



VSAM Datasets

The branch and account data is held in VSAM KDSDS datasets ACCOUNTS and BRANCHES. All dataset names in the diagram are just given as a single qualifier with an implicit prefix of USER.CICS.Z032.ZEUSBANK. So, for example, the full dataset name of the TXNOFFLD dataset is USER.CICS.Z032.ZEUSBANK.TXNOFFLD and the full dataset name of the ACCOUNTS dataset is USER.CICS.Z032.ZEUSBANK.ACCOUNTS as known to VSAM (which, since it is a KSDS, has data and index components USER.CICS.Z032.ZEUSBANK.ACCOUNTS.DATA and USER.CICS.Z032.ZEUSBANK.ACCOUNTS.INDEX respectively).

Each record in the BRANCHES KSDS is of the form

- 4-byte binary unsigned integer representing a 6-digit decimal branch sortcode (e.g. 420101)
- 1-byte EBCDIC flag:
  - EBCDIC 'A' means the branch is owned by ZeusBank and all its accounts are held in the ACCOUNTS KSDS
  - EBCDIC 'B' means the branch is not owned by ZeusBank so we do not hold its account information in the ACCOUNTS KSDS
- 32-byte EBCDIC character branch name (e.g. “ZEUSBANK FOOFORD BAR STREET” or “OTHERBANK QUUXHAM BAZ ROAD”), blank padded on the right

Each record in the ACCOUNTS KSDS is of the form

- 4-byte binary unsigned integer representing a 6-digit decimal branch sortcode (e.g. 420101)
- 4-byte binary unsigned integer representing an 8-digit account number (e.g. 12345678)
- 4-byte *signed* integer holding the account's current balance
  - For this simplistic bank account model, there is no particular currency associated with this number, it is integral and we do not worry about overflows or underflows
- 32-byte EBCDIC character account owner name (e.g. “PAT JONES”), blank padded on the right

Transactions which change the database also log a record to the Transient Data queue ZBLG. These are 80-byte records are of the form

- 8-byte EBCDIC decimal transaction id number
- 8-byte EBDIC transaction date in the form yyyymmdd
- 6-byte EBCDIC transaction time in the form hhmmss
- 8-byte EBCDIC teller userid (defaults to the z/OS userid of the CICS user entering the transaction), blank padded on the right
- 1-byte EBCDIC character operation (see below)
- 49-byte operation-specific data, blank padded on the right

Operations and their operation-specific data are:

- Add an Account
  - op character 'A'
  - 6-byte EBCDIC sortcode
  - 8-byte EBCDIC account number
  - 32-byte EBCDIC account owner
- Add a Branch
  - op character 'B'
  - 6-byte EBCDIC sortcode
  - 1-byte EBCDIC flag

        - EBCDIC 'A' means the branch is owned by ZeusBank and all its accounts are held in the ACCOUNTS KSDS
        - EBCDIC 'B' means the branch is not owned by ZeusBank so we do not hold its account information in the ACCOUNTS KSDS
    - 32-byte EBCDIC branch name
- Make a Payment
    - op character 'P'
    - 6-byte EBCDIC sortcode of “pay from” branch
    - 8-byte EBCDIC account number of “pay from” account
    - 6-byte EBCDIC sortcode of “pay to” branch
    - 8-byte EBCDIC account number of “pay to” account
    - 11-byte EBCDIC decimal of the payment amount between 0 and 2^31
        - Currently, the application allows the amount to be negative (hence why 11 bytes and not 10) but I've decided to stick with non-negative payments for this project.
    - 1-byte EBCDIC flag of “pay from” branch
    - 1-byte EBCDIC flag of “pay to” branch
        - EBCDIC 'A' means the branch is owned by ZeusBank and all its accounts are held in the ACCOUNTS KSDS
        - EBCDIC 'B' means the branch is not owned by ZeusBank so we do not hold its account information in the ACCOUNTS KSDS

Online Transactions

The following transactions can be used at 3270-terminals signed on to CICS.

ZBTP - Make a Payment - Prompts for “from” and “to” accounts (each a branch sortcode and account number), amount to pay and a confirmation field. Branches and accounts are looked up on field entry, where possible. Accounts which are held at Zeusbank branches will show the account owner; accounts held at other banks will not (because we do not hold the account information).

ZBTF - Lookup account information given branch sortcode and account number

ZBTG - Lookup branch information given branch sortcode

ZBTA - Create a new account

ZBTB - Create a new branch

Transaction Offload

All logged transactions are written to the ZBLG transient data queue transactionally. The contents of this TDQ is offloaded to the sequential dataset TXNOFFLD by using the transaction ZBOF. This transaction can be invoked from a terminal (or browser) but does not require one. The ZBLG TDQ uses the ATI (Automatic Transaction Facility) feature to trigger the ZBOF transaction to run whenever there are more than a certain number of entries logged in the ZBLG queue.

ZBOF invokes program ZBTXNOFF which opens the ZBOF TDQ (an extra-partition TDQ pointing at the TXNOFFLD dataset), appends the contents of ZBLG to the dataset, closes the dataset and, on success, empties the ZBLG queue.

Reports (this is where you come in)

The following reports are required. Create a PDS SHEnnnn.ZEUSBANK.CNTL (with your userid as HLQ SHEnnnn). Write JCL jobs as members with the given names which invoke ICETOOL/DFSORT to generate the following reports:

1. BRANCHES: List of branches

Using the BRANCHES VSAM KSDS as input, generate a list of branches (sortcode, flag and branch name) in a report written to a sysout named REPORT. Ensure each sort code is displayed as exactly 6 characters with leading zeroes where needed.

Hint: Invoke ICETOOL and use a DISPLAY statement. The branch number in the input dataset is 4 byte unsigned binary so choose an appropriate DSFORT input format specification. For numeric formatting, see the paragraphs on Edit Masks and Leading Zeros in the Using Formatting Items section of Chapter 11 “Using the ICETOOL Utility” of the DFSORT: Getting Started book. Example output:

```
LIST OF ZEUSBANK BRANCHES

SORTCODE   FLAG   BRANCH NAME
--------   ----   -------------------------------
  420101   A      ZEUSBANK UNUNTON
  420102   A      ZEUSBANK UNBITON
  420577   A      ZEUSBANK FOOFORD BAR STREET
  420708   A      ZEUSBANK SEPTOCTITON
  770404   B      THIRDBANK BEIGETON HIGH ST
  881122   B      OTHERBANK QUUXHAM BAZ ROAD
```

2. ACCOUNTS: List of Accounts

Using the ACCOUNTS VSAM KSDS as input, generate a list of accounts (account, balance, account owner). “Break” on branch, i.e. cause ICETOOL to generate a new page for each branch and sum up the account balances for each branch. Include a grand total balance of all accounts. Ensure sort codes (respectively account numbers) are displayed as exactly 6 (respectively 8) characters with leading zeroes where needed. Ensure non-account columns are not subtotalled.

Hint: Invoke ICETOOL and use a DISPLAY statement. Use BTITLE/BREAK/BTOTAL to implement the break. Use NOST to avoid unwanted subtotals. The balance field is 4 byte signed binary so choose an appropriate DSFORT input format specification.

Example output:

```
LIST OF ZEUSBANK ACCOUNTS BY BRANCH

SORTCODE:  420101

 ACCOUNT            BALANCE   OWNER
--------   ----------------   --------------------------------
12345678              64539   FRED SMITH
22332233                -234   BOBBY WILLIAMS
44554455                -120   A NONYMOUS
87654321             155367   PAT JONES

BRANCH TOTAL:        219552
LIST OF ZEUSBANK ACCOUNTS BY BRANCH

SORTCODE:  420102

 ACCOUNT            BALANCE   OWNER
--------   ----------------   --------------------------------
12123434              80000   JAN TAYLOR
...
GRAND TOTAL:         319971
```

3. BYTELLER: Payments by teller.

Using the TXNOFFLD dataset as input, generate a report by Teller, summed over all payments (i.e. columns for Teller and for the sum of the Payments made by that Teller). “Break” on each bank flag pair to give a separate report section for “AA” (intra-bank payments within ZeusBank), “AB” (outoing payments from ZeusBank) and “BA” (incoming payments to ZeusBank”).

Hint: Invoke ICETOOL and use two statements: a SORT (which invokes DFSORT) then a DISPLAY. The SORT statement should use control statements in a sysin which tells DFSORT to include only payment transactions and to sort/sum the appropriate fields. The SORT statement can write its output

TO(SORTTEMP), a temporary dataset that can be created with the same allocation size and structure of the input dataset by coding a DD statement after the DD statement for the input dataset (assumed here to be DD name TXNOFFLD):

```
//SORTTEMP DD DSN=&&SORTTEMP,DCB=*.TXNOFFLD
```

The absence of a DISP specification triggers the default of (NEW,DELETE) so a temporary dataset will be automatically created for the duration of the job and deleted afterwards. The TXNOFFLD dataset has its payment field in text format, not binary, so the appropriate format is “ZD”. Do not worry about the sum overflowing the field size. The ICETOOL DISPLAY statement should read from SORTTEMP to generate its report. Using formatting code A0 is one way to ensure that the display of a ZD field is more human-readable.

Example output:

```
       PAYMENTS BY TELLER

       TRANSFER TYPE  AA

       TELLER              PAYMENTS
       --------   ----------------
       BEATTIE                176
       BEATTIEA               132
       SHE0001             234579
       SHE0002                585
       SHE0030              80023
       SHE0030A               128

       SUBTOTAL            315623
       PAYMENTS BY TELLER

       TRANSFER TYPE  AB

       TELLER              PAYMENTS
       --------   ----------------
       SHE0030              80021
...
```

4. BYHOUR: Payments by hour

Using the TXNOFFLD dataset as input, generate a report of payments summed over each hour. Columns should be Date, Hour and Payments. Break by bank flag pair again to make a separate section for “AA”, “AB” and “BA” payments. Do not worry about the sum overflowing the field size.

Example output:

```
       PAYMENTS BY HOUR

       TRANSFER TYPE  AA

       DATE       HOUR           PAYMENTS
       --------   ----   ----------------
       20130309   23                  256
       20130310   00                  180
       20130310   18                   23
       20130315   18               235164
       20130321   21                80000

       SUBTOTAL                    315623
       PAYMENTS BY HOUR

       TRANSFER TYPE  AB

       DATE       HOUR           PAYMENTS
       --------   ----   ----------------
       20130310   19                  20
       20130425   14                80001
```

Part 1 Deliverables

What I need from you is the following:

1. Your PDS ***SHEnnnn*.ZEUSBANK.CNTL** with its four members: BRANCHES, ACCOUNTS, BYTELLER and BYHOUR. In other words, leave that dataset in place on ZOS113 and do not remove or edit it after you have submitted part 1 of your project.
2. A “cover note” email which includes
    - your name and university id number
    - your ZOS113 userid of the form SHEnnnn
    - the contents of the PDS members from (1)—cut-and-paste in text form is fine.
    - Unless there are special circumstances, this cover note should simply be an email to me at beattiem@uk.ibm.com , and a file containing it should be submitted to the MOLE dropbox for this part of the assignment.

Deadline

The deadline for submission of part 1 is 3pm on Monday 1 December and for part 2 is 3pm on Friday 19 December.

Documentation

The z/OS base manuals and books are all available in PDF form from http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/r13pdf/

Of those, the two you are most likely to find useful for the DFSORT/ICETOOL parts of the project are the following two manuals, both of which are referenced in links in the above page:

- the *DFSORT Getting Started* manual. The direct link to the PDF is http://publibz.boulder.ibm.com/epubs/pdf/ice1cg60.pdf
- the *DFSORT Application Programming Guide* manual. The direct link to the PDF is http://publibz.boulder.ibm.com/epubs/pdf/ice1ca61.pdf