Introduction to Rational System Developer for System z

(also known as RDz)

Module Objectives:

* Become functional with the RDz environment for a number of Perspectives including Remote System, Java, and Database Development
* Be operationally proficient in the Remote System Perspective
* Be able to create partitioned datasets; copy, move, edit, upload, and download files
* Be able to use selected utilities including IDCAMS to create VSAM files
* Be able to run simple programs that utilize files and JCL on the mainframe
* Be able to run a Cobol program against DB2

**Workbench fundamentals**

The workbench is the user interface for Rational® Developer for System z which is based on the Eclipse open source platform. The workbench features a graphical integrated development environment with customizable *perspectives* that support role-based development. The workbench provides a common way for all members of your project team to create, manage, and navigate resources easily. It consists of interrelated *views* and *editors*. Views provide different ways of looking at the resources you are working on. Editors allow you to create and modify code. More than one workbench window can be open on the desktop at any given time. To access the Rational Developer workbench we must first establish connections to the system.

**Create a z/OS connection to the University of Arkansas (UA) System z**

To use RDz, your must first start the software and then create a connection to some part of a System z—generally to a z/OS LPAR. RDz provides a GUI option for the ISPF commands that have been used for so many years on IBM mainframes. As you will learn, you can also connect to DB2 on a System z, create Websites and many other features. Many of these features are accessed via **Perspectives**—more on this later. The following illustrates connecting to the System z at the University of Arkansas which is accomplished via Remote Desktop.

See [http://enterprise.waltoncollege.uark.edu/files/Remote\_Desktop\_Connection.pdfOpen](http://enterprise.waltoncollege.uark.edu/files/Remote_Desktop_Connection.pdfOpen%20) for information on using Remote Desktop for access to the UA System z. From the remote desktop, you can start RDz by double-clicking the IBM Rational Developer..icon or via the Start menu.





Click on the RDz selection shown above and accept the default Workspace: location by clicking the OK button. If this is the first logon for the account, you will have a help system displayed—click the x on the tab to delete it. Depending on the configuration and whether you have logged on before, you may be asked to enter a password immediately.

Assume this if the first time you have opened RDz, the default location for the Remote Systems will most likely be the upper right-hand corner as shown below—note that New Connection has been expanded.



 To create a new z/OS connection, either double-click or right-click z/OS and select New Connect as shown below.



The New Connection dialog opens to allow you to enter appropriate information. The Host name is the only critical one—you must enter 130.184.26.148 as that is the LPAR you will be using. This illustration uses the default Parent profile name and I recommend using your ACCTID for the connection name. An entry for Description is optional.



Click the “Finish” button.

You may receive a popup screen that says “Connection 130.184.26.148 has not been secured using SSL. Proceed anyway?” You may answer “Yes” to proceed. Now we are logged into the z/OS system and should have an entry similar to the right.

Notice that the z/OS UNIX files and shells are listed first and then followed by MVS files. Expanding the MVS Files will lead display My Data Sets which can be expanded to show all the files MVS files for this account. You will find that you can do everything that we have already done in the ISPF environment in RDz!

Click on My Data Sets—most likely, you will be asked for an ACCTID and password. Following successful entry of an ACCTID and password, click the “Yes” button if prompted by an information dialog. All the Partitioned Datasets (PDSs) for the account will be listed. Also, expand My Jobs under JES (Job Entry System). See example to the right.

Local files



MVS Files

PDS

UNIX files and Shells

By opening the JES tab and the MVS tabs, you will see the SDSF output from your class and homework assignments. By opening MVS Files🡪 My Data Sets you will see the PDS’s that you created previously and the members in those PDS’s. But look at what we can also see at the top! Files from your workstation! Instead of having to go through that tedious upload/download process, which we did in the ISPF environment, we can now simply drag and drop our content into PDS members.

To illustrate how easy it is to create a PDS and member, right-click on MVS Files and select New and then select Allocate Partitioned Data Set.

Jobs that have been run



The file Type of a PDS takes on very important meanings in RDz which has been designed to sense a number of file types. For example, a file type of Cobol or JCL will automatically check syntax for you. Because you may already have a number of PDSs created, this example will create PDSs that are meant to be deleted after doing some exercises with them. Create a PDS named ACCTID.TEMP.COBOL.

Notice that the high level qualifier is your ACCTID—thus, you do not need to enter it again. Click the “Next” button to open the Specify data set characteristics page.

The default option for this page is specify characteristics by usage type. For many PDSs, this would be the best selection as the system knows that Cobol PDSs should be fixed 80 character records, etc. However, for this illustration, click the Specify characteristics (Advanced Allocation) as shown. Then click the “Next” button.

Use a Volume Serial: provided by your instructor--DB1469 is used in this example--and match the remaining entries as shown. The drop down box for the Data Set Type: includes PDSE—Partition Data Set Extended. Ensure that you select PDS and not PDSE as we have had some problems with PDSE.

In order to illustrate mapping of file types and file extensions, create a PDS named ACCTID.TEMP.DATA with FB 80 character records. Then right-click on the PDS and select Properties from the drop-down menu. In the left pane, click mappings and for the file extension, click the Other option and key in txt; click the Other option for Transfer and use the drop down box to select text.

You should do this before copying files into the PDS.



To see how easy it is to transfer files between the System z and a local computer, create a file in notepad with the following text:

RDz is so much fun!

It is far easier to create PDS members and transfer files.

Hooray for RDz!

Save the file to a location on the remote desktop with the name RDZISFUN. In the remote systems panel, simply right click on the local file that you just created and saved. Scroll down and find the ACCTID.TEMP.DATA PDS you just created. Right-click the PDS and select Paste—verify that the file has been placed into the PDS. Double-click the file to open it in an editor—this is not a line editor.



**Creating and Running a COBOL program in RDz**

**Single Job stream:**

In the ISPF environment, you created an ACCTID.WORK.COMPILES PDS to which you copied a single job stream named SIMPLE. Double-click that file to open it in an editor—most likely will be in the middle of the workspace.



Scroll down to the Paragraph 1000-PROCESS-IT and intentionally create an error—for example delete the letter “E” on one of the MOVE statements. Nothing noticeable happens as the PDS Type Complies has not been set to provide a context sensitive editor. If the Cobol code was in a PDS of Type COBOL, a syntax error would have been highlighted for you. Also, note that the tab (upper left) now has an \* which means a change has been made but not saved. The easiest way to save is to click the disk icon on the menu bar. Click the X on the tab to close the file in the editor.



Reverse the intentional error so that the program will run correctly. Copy this SIMPLE file from the ACCTID.WORK.COMPILES PDS to your ACCTID.WORK.JCL PDS – changing its name to SIMPLEC to indicate it is the complete stream—via a copy and paste. Right-click the SIMPLE file in the ACCTID.WORK.COMPILES PDS, select Copy and then right-click the ACCT.WORK.JCL PDS and select Paste. If a file with the same name exists in the PDS, then you will get a Duplicate Name Collision dialog—click the Rename option and rename the file SIMPLEC. Click the “OK” button.



Double-click the SIMPLEC.jcl file in the ACCTID.WORK.JCL PDS to open it in the context sensitive editor. Intentionally make an error in a JCL statement and notice that the error being highlighted. Reverse to intentional error.

Right-click on the file and select Submit.



A Job submission confirmation dialog provides the job number for you to find in the JES (Job Entry System) list of jobs. Click the “OK” button.



Scroll down and expand JES and My Jobs to see a list of jobs. Right-click on My Jobs and select Refresh—this is necessary to refresh the list of jobs. The last job should be at the top of the list. You can expand to view pieces of the job or double-click the job to open it.

For this illustration, double-click the job and the JES Job Log provides the output for the job stream. Scroll to the bottom and notice the Message Summary Report shows NONE for all error types—thus, the job ran successfully. The output should be in the ACCTID.WORK.OUT PDS – verify this by double-clicking the COBOUT file and the results will be in the editor as shown below.



**Unwrapping the single job stream into appropriate PDS types**

The single stream approach has advantages over the ISFP approach as all the resources needed are within the same interface and does not require changing a number of screens. However, this single stream approach does not maximize the benefit of RDz which has a number of context sensitive editors as well as interactive debugging.

Thus, the next step is to unwrap the single job stream into the PDS types, JCL, COBOL, and DATA. Because you have a job stream that runs, it is mostly a matter of copy/paste/delete to get the correct file into the appropriate PDSs. You may have already done this in ISPF; but if not, copy the SIMPLE program in the ACCTID.WORK.COMPILES into the ACCTID.WORK.DATA, ACCTID.WORK.COBOL and ACCTID.WORK.JCL PDSs. For the SIMPLE copied file, do the following

* Delete everything except the COBOL code in the ACCTID.WORK.COBOL PDS
* Delete everything except the data in the ACCTID.WORK.DATA PDS and name it SMPLDATA
* Delete everything except the JCL in the ACCTID.WORK.JCL PDS and alter the JCL to reference the Cobol program and the input data

The JCL should be like below except you will need to include your correct accitid.

//COBJOB JOB (COBOLPROG),'UOASxxx',NOTIFY=UOASxxx,

// CLASS=A,MSGLEVEL=(1,1),TIME=1,MSGCLASS=A

//STEP1 EXEC PROC=IGYWCLG

//COBOL.SYSIN DD DSN=UOASxxx.WORK.COBOL(SIMPLE),DISP=SHR

//GO.DISKFL1 DD DSN=UOASxxx.WORK.OUT(COBOUT),DISP=SHR

//GO.READER DD DSN=UOASxxx.WORK.DATA(SMPLDATA),DISP=(OLD,KEEP)

COBOL program is saved as SIMPLE member in PDS -UOASXXX.WORK.COBOL(SIMPLE):

IDENTIFICATIONDIVISION.

PROGRAM-ID.COBTEST.

ENVIRONMENTDIVISION.

INPUT-OUTPUTSECTION.

FILE-CONTROL.

SELECT CARD-IN ASSIGNTO UT-S-READER.

SELECT DISK-FILE1 ASSIGNTO UT-S-DISKFL1.

DATADIVISION.

FILESECTION.

FD DISK-FILE1 LABELRECORDSARESTANDARD

RECORDING F

BLOCKCONTAINS0RECORDS.

01 DSK-RCD1.

02FILLERPICX(80).

FD CARD-IN LABELRECORDSOMITTED

RECORDING F.

01 CRD-REC.

02 CRD-NAME PICX(30).

02FILLERPICX(23).

02 CRD-U-ID PICX(7).

02FILLERPICX.

02 CRD-PASSW PICX(6).

02FILLERPICX(13).

WORKING-STORAGESECTION.

01 EOFPICXVALUE'N'.

01 HOLD-INST.

02 CRD-CRSE PICX(30).

02FILLERPICX(23).

02 CRD-INST-ID PICX(7).

02FILLERPICX.

02 CRD-PWD PICX(6).

02FILLERPICX(13).

01 OUTPUT-INST.

02 CRD-OUT-ID PICX(7).

02FILLERPICX(2)VALUESPACES.

02 CRD-OUT-PWD PICX(6).

02FILLERPICX(65)VALUESPACES.

PROCEDUREDIVISION.

OPENINPUT CARD-IN

OUTPUT DISK-FILE1

PERFORM 2000-RD-CRD-RT

PERFORM 1000-PROCESS-IT

UNTIL EOF ='Y'

CLOSE CARD-IN

 DISK-FILE1

STOPRUN.

2000-RD-CRD-RT.

READ CARD-IN INTO HOLD-INST

ATEND

MOVE'Y'TO EOF.

1000-PROCESS-IT.

MOVE CRD-INST-ID TO CRD-OUT-ID

MOVE CRD-PWD TO CRD-OUT-PWD

WRITE DSK-RCD1 FROM OUTPUT-INST

PERFORM 2000-RD-CRD-RT.

Data file saved as SMPLDATA member -- UOASXXX.WORK.DATA(SMPLDATA):

ANDERSON ESSABIA ANNIECE XXXXXXXXX ISYS4283 002 EANDERS PWD001

ANIPA EKO A XXXXXXXXX ISYS4283 002 EAANIPA PWD002

BLOMBERG BETTY ANN XXXXXXXXX ISYS4283 002 BBLOMBE PWD003

CORDIERO SHELDON JOHN XXXXXXXXX ISYS4283 002 SCORDEI PWD004

NESTRUD CHRISTOPHER CHARLES XXXXXXXXX ISYS4283 002 CCN PWD005

RODRIGUEZ SARAH MARTHA XXXXXXXXX ISYS4283 002 SRODRIG PWD006

SIMMONS KIMBERLY M XXXXXXXXX ISYS4283 002 KMS06 PWD007

WIDEMAN JOHN RANDELL XXXXXXXXX ISYS4283 002 JWIDEMA PWD008

After copying each of your files into the appropriate PDS, you will want to check each one to make sure that no “anomalies” have occurred when moving or copying the files. For example, perhaps you eliminated the space on the 2nd line after the //--you will get the error shown below.



Double-click the SIMPLE Cobol program in your ACCTID.WORK.COBOL to invoke the Cobol syntax editor. Again, scoll to the bottom and remove the Letter “E” on one of the move statements—this command is flagged to its left margin as shown.



Move the cursor over the icon to reveal a message. Thus, RDz provides a great code syntax editor.



Reverse this error to ensure the program will run. In the ACCTID.WORK.JCL PDS, right-click on Simple and select Submit. As before a Job submission confirmation dialog appears with the job number—JOB01177 in this case. Click the “OK” button.



As before, find this job in the list of My Jobs in the JES folder after refreshing the job list.



Notice that you still see the same SYSPRINT, JESMSGLG, JESJCL and JESYSMSG files that we saw before in the ISPF environment. Double click on the message to open it in the central editor panel. Again, we need to look for COND CODE 0. . . . .

**RDz tip—double-click any table to maximize it; double-click it again to return to initial size.**

**DB2 –Create a Table:**

As previously mentioned, RDz has a number of perspectives—one is a Database

Development perspective which allows working with databases and, in particular, DB2. This illustration opens the Database Development perspective and creates a connection to DB2 which allows you to create and query DB2 tables.

To open the Data Development perspective, click Window on the main menu and select Open Perspective.



Click Other to get the complete list of perspectives and select Database Development.



Click the “OK” button and locate the Data Source Explorer tab. Expand the Database Connections folder and locate the ZUAF [DB2 Alias] entry. Double-click this entry to open the Properties for ZUAF dialog. Accept Other Driver Default for the Drivers: entry and enter your ACCTID and password. **It is important that the database be ZUAF in upper case.**



Click the OK button.

Expand the ZUAF entry (and Schemas) until you find your ACCTID which serves as the name of your DB2 database.



Expand the Tables entry and then expand a table. In this case, the table name is FIRST.



RDz provides rich features for working with a database—most every option is possible via right-clicking. For example, right-click the table, select Data and then Edit. A window opens which allows direct entry into the table. Remember to save.

RDz also provides a very powerful Query Builder feature. At the top of the Data Source Explorer is an icon that opens a SQL Script set of windows. Click this icon and if needed select the ZUAF connection.



A blank script window appears – you can enter any legitimate SQL statements directly in this window. The SQL statements can be executed by click the green go arrow on the Toolbar or right-click and select Run SQL. Try an example and notice that it lists your tables?



The results of running a query are shown in the SQL Results tab toward the bottom of the workspace. The left pane provides the status of the run-successful or not. It not, then the error message will be in the right pane. If the SQL runs successfully, any result rows are displayed in the right pane.



The SQL create statements for the First table are below:

CREATE TABLE FIRST (

 FNAME CHAR(10) NOT NULL,

 LNAME CHAR(10) NOT NULL,

 UPPICE DECIMAL(7 , 2) WITH DEFAULT NULL,

 QTY SMALLINT WITH DEFAULT NULL

 )

 AUDIT NONE

 DATA CAPTURE NONE

**Using the Query Builder Feature**

Delete the SQL Script tab if needed and click the icon at the top of the Data Source Explorer that Opens a scrapbook to edit SQL statements.



Once in the SQL Scrapbook editor, right-click in the open window and Select Edit in SQL Query Builder. In the middle left pane, right-click and add tables. The bottom pane uses tabs to lead you through the steps of selecting the columns, conditions, groups and group conditions.



As you are building the SQL statement in the bottom pane, the actual SQL statement is shown in the top pane. In this case, it will select all the records with a last name of the letters “OU”. After building the SQL Statement, right-click in the upper pane and select Run SQL.

As an example, create a query that lists all the rows where the first name starts with “Geor”.



The resulting answer is:

