Accelerate your digital transformation with the power of IBM Application Discovery & Delivery Intelligence (ADDI)

IBM Application Discovery and Delivery Intelligence (ADDI)

A Proof of Technology

Hands-on Lab Exercises
Contents

ACKNOWLEDGMENTS ........................................................................................................................................... 4
INTRODUCTION .................................................................................................................................................... 5
OVERVIEW .......................................................................................................................................................... 6
LAB 1 – JKE BANKING APPLICATION ANALYSIS .......................................................................................... 7
1.1 IBM APPLICATION DISCOVERY ECLIPSE INTERFACE OVERVIEW ....................................................... 9
1.2 DRAW A TRANSACTION CALL GRAPH FOR THE JKEBANK APPLICATION ........................................... 9
1.3 UNDERSTAND A PROGRAM’S LOGIC BY CREATING A PROGRAM FLOW CHART ................................ 16
1.4 INVOKE COBOL VARIABLE USAGE FOR JKEMLIST ............................................................................... 21
1.5 CREATE AN IMPACT ANALYSIS REPORT ON A PROGRAM VARIABLE ............................................... 25
1.6 ANNOTATING ARTIFACTS WITH APPLICATION DISCOVERY .............................................................. 28
1.7 SWITCH TO IDz DEVELOPER PERSPECTIVE AND OPEN JKEMLIST (OPTIONAL) .......................... 31
1.8 SUMMARY .................................................................................................................................................. 35
LAB 2 – HOSPITAL BATCH APPLICATION ANALYSIS .................................................................................. 36
2.1 PRODUCE A CYCLOMATIC COMPLEXITY REPORT ...................................................................... 37
2.2 SHOW PROGRAM FLOW FOR THE MOST COMPLEX PROGRAM .......................................................... 39
2.3 SHOW DATASET USAGE IN JOBS ............................................................................................................. 41
2.4 SHOW DATASET USAGE IN PROGRAMS ................................................................................................. 44
2.5 SHOW SQL USAGE .................................................................................................................................... 47
APPENDIX A. NOTICES ................................................................................................................................. 50
APPENDIX B. TRADEMARKS AND COPYRIGHTS ......................................................................................... 52
Acknowledgments

This Hands-on lab workbook is a refinement and modification of the following materials:

- The ‘IBM EZSource Demo Guide’, created by Roberto Calderon.
- The ‘Discovering the value of IBM Developer for z and IBM Application Discovery Workbook’, published by Joseph Taffe-Atkins.
Introduction

The information in these opening sections of this document provide preparatory information for this Proof of Technology (PoT) Hands-on lab session.

This Hands-on lab takes students through typical scenarios encountered in their organizations. Many industry sectors are going through major changes to increase profitability and comply with new regulatory requirements in an ever-changing business climate. In addition, these businesses must become more customer-focused, which drives the need to evolve the enterprise infrastructure away from technology “silos” and towards integrated, transparent, customer-focused business solutions. At the same time, the new economic environment is driving another wave of mergers and acquisitions. The challenges of integrating different business models and the supporting technology add further complexity and risk to modernization efforts.

This situation is aggravated by the fact that an increasing share of the allocated budgets is spent on maintaining the existing core system applications, which in turn reduces capital available for investment for rebuilding or evolving them to meet changing business opportunities.

To help with these challenges, IBM® Application Discovery automates application discovery and analysis to enable users to:

- Unlock application understanding. Rapidly analyze and visualize relationships between application components, data, and jobs to make changes quickly, safely, and efficiently.
- Empower developers. Improve productivity of new and existing developers through knowledge transfer and automated documentation.
- Appraise quality and assist application modernization efforts.
Overview

IBM® Application Discovery operates across multiple languages and environments, producing consistent understanding and measurement. Through its ability to parse and compile the different component types (for example, source code, OS scripts, include files, screen maps, transactions, messaging system data, application server data, customization data, etc.) it can then visualize the cross applications, internal structure and dependencies. Using rich static code analysis functionality, it gives users the opportunity to select and configure their own analysis start point to navigate across or drill down as dictated by the IT development, maintenance or support task in hand.

Many different options are provided to visualize the analysis according to requirements and results. For example, a particular analysis may reveal complex inter-dependencies with different types of components, so we enable user definable dependency levels, filtering and visualization layouts in order to achieve optimum understanding. In all stages in the analysis, IBM® Application Discovery synchronizes with the current source code version so developers can understand exactly how functions are performed.

In these Hands-on labs we will focus on a small subset of IBM® Application Discovery features and functions. Some of the capabilities utilized are:

- Use of Eclipse-based tooling to visualize and navigate around mainframe applications.
- Impact analysis, tracing an application error back to its root cause.
- The provision of automated documentation.
- Complexity analysis.

Two z/OS applications will be analysed in this workshop:

**JKE Banking CICS Application**

JKE is a COBOL CICS based application that is accessed via a traditional green screen (3270) user interface.

**Hospital Batch Application**

The hospital application is a z/OS batch application. Applications such as these are very common with z/OS data centers. Although it has no online components it is often critical to be able to understand how batch jobs relate to the programs being executed in the JCL, as well as to understand the datasets being used for input and output of the batch jobs.

Requirements

When using the VMware built for this Hands-on lab session, please use the following credentials:

User: **Administrator**

Password: **passw0rd** ← 0 is a zero
Lab 1 – JKE Banking Application Analysis

The IBM Application Discovery Eclipse client is started with the IDz Eclipse client shortcut on the taskbar.

Open the IDz UI and when prompted, make sure C:\Users\Administrator\IBM\rational\dp\workspace is the workspace specified.

If not there, switch to the Application Discovery Browser perspective at this point to get ready for the lab.

Hands on Lab Overview

The JKE Bank application is a CICS, COBOL based application for which the primary interface today is traditional green screen. The mainline transaction for this application is JKEP.

The customer needs to clearly understand the application flows to safely make application changes and enhancements that have been requested by the business units such as mobile and API enablement. Among others, some key IBM Application Discovery features that will be explored are:

- Show Transaction call graph
- Show graphing – objects, relationships, overview, inventory, different layouts
• Show screen layouts for input screen and output screen – gives overview of Business functionality

• Show IBM Application Discovery to IDz interface

• Show COBOL program variable Impact Analysis

• Show IBM Application Discovery reporting capabilities
1.1 IBM Application Discovery Eclipse Interface Overview

After opening the Application Discovery Eclipse interface there will be several tabs already opened that highlight the main features of the IBM Application Discovery solution.

The initial screen on the Application Discovery Analysis browser contains 4 main tabs which represent the 4 main features of the solution. The first is the AD Explore tab. It contains a list of the projects that have been created in Application Discovery. A project is a logical Application Discovery construct that can represent an application, a subset of an application, specific business functions or whatever the user decides is suitable for them.

The second tab is the AD Graphs tab. As the name implies, it will contain graphs of the application being analyzed. An example of such graph could be a Transaction Call Graph that we will show in the next section.

The third tab is the AD Reports tab. It will contain reports generated for a application analysis. An example of such report is an Impact Analysis report which will also be explored.

The fourth tab is the AD Usage tab. It will contain usage information for a particular component of a source being analyzed such as a usage analysis for a COBOL variable. This feature will also be explored.

1.2 Draw a transaction call graph for the JKEBank Application

The mainline transaction for this application is JKEP.

As a developer needing to make a change to a critical business application, you need to understand the upstream and downstream flow. The first step in getting a better understanding of a complex business application is to see a graphical representation of the various programs, transactions, and other components that make up the application.

A transaction call graph makes it easy to understand the overall structure and flow of the application before attempting to make a change.
_1. Open the Application Discovery Browser Perspective from Windows > Open Perspective > Other > Application Discovery Browser. Then navigate to the AD Explore/Explore projects tab in the Eclipse client.

_2. You will see a list of applications already populated into the Application Discovery database and attributes describing the scanned contents of each application. Single click on the JKEbank application to select it.

_3. In the right pane, you will see a list of functions available to analyse the application selected.

Double click on the **Transaction Callgraph** under **Mainframe Graphs**.
__4. The **Transaction Callgraph Analysis** dialog will appear. Select the **JKEP** transaction from the list of **Available Transactions**, click on the arrow icon pointing to the right to move JKEP to the **Selected Transactions** list.

![Available transactions](image)

__5. Click **Finish**

The JKEP transaction relationship to the COBOL programs is derived from a CICS CSD report which is processed by Application Discovery.
The **Graph Overview pane**, and the **AD Graphs** will be filled with the JKEBank application call flow.

6. In the **AD Graphs** main tab, select the interactive zoom tool icon.
While holding the left mouse button down, scroll down to zoom in, and up to zoom out. You will notice that the area shown in the main tab is highlighted with a blue square in the **Graph Overview** pane.

You can also use the mouse pan tool to move the blue box around in the overview pane.

Zoom in and out moving the blue box in the overview pane highlights different areas of interest in the main graph window.
7. Zoom in on the JKECMORT COBOL program.

Notice the CICS related object associated to it. Hover over JKECMORT-JKEMENU and you will notice its a MAP, in other words a BMS map screen with input and output. You may be interested in seeing the layout and variables associated with the BMS map. Right click over the BMS map and select Layouts, then Screen Layout.

Application Discovery will draw what the user would see on a 3270 screen when executing this application. We can see that this is a mortgage calculator program that is asking to enter a monthly payment amount.
8. Close the JKEMORT-JKEMENU Screen Layout tab and go back to the AD Graphs tab. Navigate down to the JKEMLIS-JKEMLIS object and notice that it is an output only screen. Right click over the BMS map and select Layouts, then Screen Layout.

9. Notice the Interest Rate field. Let’s assume that is the variable in our production application that we are trying to perform analysis on. We would like to find out how the variable is acted on by our JKEBank CICS application.
Go back to the Transaction Call Graph so you can continue to the next section of the lab.

1.3 Understand a program’s logic by creating a program flow chart.

Once the developer has gained a clear understanding of the overall application flow and structure, and once a program has been identified as needing a change, the next step is to understand the flow of the program in question.

In this section, the developer not only sees a graphical representation of the program flow, but is also able to narrow down on the specific pieces of COBOL code that are impacting the variable that the developer is interested in.

1. From the main graph tab, right click over JKEMLIST and select Mainframe Graphs followed by Program Flow. The result will be a graphical representation of the program flow of JKEMLIST.

2. The resulting program flow, with the output CICS BMS map, is presented in this diagram.
3. Now we want to isolate the program logic/flow that leads to the BMS map output, which has the interest rate field. First click on the ‘Hide all’ edges icon on the toolbar.

The resulting map will look like the one above without any of the line connectors.

4. Now right click over the JKEMLIS-JKEMLIS BMS map icon and select Filter then Show backward and hide unconnected.
5. Select the Change layout to the 'hierarchical button' on the toolbar. The resulting graph shows ONLY the program flow/logic that leads to the BMS map output.
6. The next step would be to now understand where in the source code, for this particular JKEMLIST program, are the CICS commands that lead to the BMS map being displayed so you can see the data areas associated with the map display. Hover over the down arrow pointing to the JKEMLIS-JKEMLIS, right click over the arrow and select View analysis source...

7. In the resulting dialog, select the first source line reference (line 256) and then click on Go to source.
After selecting Go to source, Application Discovery will display, at the bottom of the Eclipse UI on a tab called **JKEMLIST**, the exact CICS call that will lead to the display of the BMS map on the terminal window.

Using the scroll bar on the window, scroll down a few more lines to see the data area which is being used to populate the data in the EXEC CICS SEND MAP command. The data area is called **JKEMLISO**.
At this point we have identified the data area that contains the variable we are trying to focus on. Our next step is to drill one more level down and create a **Variable Usage analysis** to understand more details about the variable **JKEMLIST**. We will demonstrate this feature in the next section.

### 1.4 Invoke Cobol Variable Usage for JKEMLIST

Now that we have identified the variable we are interested in, before making any changes it is important to understand how the variable is read or updated in the actual COBOL program. A **COBOL Variable Usage** report will allow the developer to know all the details about the variable.

1. From the main graph tab, right click over **JKEMLIST** and select **Usage in Programs** followed by **COBOL Variable Usage**.

   ![Variable Usage Report](image)

   The resulting display shows the COBOL Variable Usage analysis.

   ![Variable Usage Display](image)

   - **Project(s):** JKEBank. **Conditions:** Program(JKEMLIST, 10) Variable(*)
   - **Results:** JKEMLIST

2. Click on the arrow next to **Project(s)** to display the attribute filters. Notice that the filter has been pre-filled to match the program name on which we started the variable analysis.
Since we are looking for the **JKEMLISO** variable, type that name into the search box in the middle of the page. The results pane will highlight the COBOL variable data area as a 01 JKEMLISO.
3. The icon next to 01 OUTMAP shows that it is a COBOL REDEFINE of the variable right above it which is JKEMLISO. If you click on the arrow to open 01 OUTMAP, you will see a 03 OUTMAP-REPEAT. Open up 03 OUTMAP-REPEAT to reveal the 05 OUTMAP-RATE.
4. Expanding **05 OUTMAP-RATE** reveals details about the variable such as if the variable is being read from or written to, the type of COBOL statement used to alter the variable, in this case a COBOL MOVE command, and the actual line in the JKEMLIST source member where the MOVE command is being executed.

5. Double-click on the JKEMLIST (Line 223) to have Application Discovery take you to line 223 in the source code where the MOVE statement is being executed.
At this point we have been able to drill down all the way from the business application level down to a specific variable which holds special interest.

1.5 Create an Impact Analysis report on a program variable

Now that the developer understands all the details of the COBOL variable in question, before making any changes it is key that we not only understand how the variable is used in the COBOL program we are looking at, but more importantly how a change to this variable definition may potentially impact OTHER programs that use the value represented by this variable. An Impact Analysis Report will show the developer in depth detail of how other programs in the business application use this variable and therefore its potential impact on the overall business. Note that depending on the amount of impacts and expansion depth selected, some reports may take a while to run.
1. Click on the **05 OUTMAP-RATE** field in the search results window.

   ![Load Save Enable levels limit](image1)

   ![JREMLISO](image2)

   Results:
   - 05 FILLER #143
   - 05 OUTMAP-COMPANY
   - 05 FILLER #144
   - 05 OUTMAP-PHONE-NUM
   - 05 FILLER #145
   - 05 OUTMAP-RATE

2. In the right pane, the available reports for the variable selected will be displayed. Under **Mainframe Reports**, expand **Impact Reports** and double click **Impact Analysis Report**.

   ![Quick filter](image3)

   - Mainframe Reports:
     - Impact Reports:
       - Field Expansion Report
       - Field Usage Report
       - Impact Analysis Report
     - Usage in Programs
     - Cobol Variable Usage

3. In the **Impact Analysis report Wizard**, since the OUTMAP-RATE is an output field, select **backward** as the impact direction. Do not limit expansion depth and click **Next**.

   ![Impact Analysis report Wizard](image4)

4. In the **Exclude variables** dialog, do not exclude any common variables in the next dialog and click **Finish**.

   The Impact Analysis report for OUTMAP-RATE will be displayed in the Application Discovery Reports tab.
Impact Analysis Report

Organization: -
Project Name: JKEBank
Variable(s): OUTMAP-RATE (program: JKEMLIST)
Options: backward
Date: Jan 5, 2017

Here we can see an example of the report where the impact of OUTMAP-RATE is traced back to understand how other variables and programs impact the contents of OUTMAP-RATE

Impact analysis of variable WS-CALC-INTEREST
Def. Source: \127.0.0.1\EZSource\EZSource Build\Sources\Mainframe Library Members\ZCOSDEV\POS_MVS\EMPOT.Demo.DEV.COBOL\JKEMPMT(line 64)
Context program: JKEMPMT

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
<th>Impact type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTMAP-RATE (JKEMLIST)</td>
<td>WS-FORMAT-NRMBR(1, 5)</td>
<td>COBOL MOVE</td>
</tr>
<tr>
<td></td>
<td>JKEPROM-RETURN-MONTH-PAYMENT</td>
<td>COBOL MOVE</td>
</tr>
<tr>
<td></td>
<td>DFHCMMAREA(Linked Section)(469, 472)</td>
<td>Called program</td>
</tr>
<tr>
<td></td>
<td>JKEC9MRT (program)</td>
<td>Program argument</td>
</tr>
<tr>
<td></td>
<td>DFHCMMAREA(Linked Section)(469, 472)</td>
<td>Direct dependency</td>
</tr>
<tr>
<td></td>
<td>JKEPROM-RETURN-MONTH-PAYMENT</td>
<td>COBOL MOVE</td>
</tr>
<tr>
<td></td>
<td>JKEPDATA-RETURN-MONTH-PAYMENT</td>
<td>Parent Record Dependency</td>
</tr>
<tr>
<td></td>
<td>JKEPDATA(18, 21)</td>
<td>Called program</td>
</tr>
<tr>
<td></td>
<td>JKEMPMT (program)</td>
<td>Program argument</td>
</tr>
<tr>
<td></td>
<td>JKEPDATA(Linked Section)(18, 21)</td>
<td>Direct dependency</td>
</tr>
<tr>
<td></td>
<td>JKEPDATA-RETURN-MONTH-PAYMENT</td>
<td>COBOL COMPUTE</td>
</tr>
<tr>
<td></td>
<td>WS-CALC-INTEREST</td>
<td></td>
</tr>
</tbody>
</table>
Notice that OUTMAP-RATE is the target of a COBOL MOVE in program JKEMLIST as we saw previously, but also we can see how WS-CALC-INTEREST impacts the content of OUTMAP-RATE not only in the JKEMLIST but also on other called programs. This is an invaluable tool to help developers understand how to trace back the contents of a variable to fix a problem, or how to understand the potential impact of a change to a variable on other programs and variables of the business application.

1.6 Annotating artifacts with Application Discovery

Application Discovery allows for developers that are making changes to any application artifact to annotate their changes in order to collaborate with other developers.

1. Go back to the AD Graphs tab and look again at the JKEMLIST program. Notice that there is a pencil icon on the top left corner of the program icon.

2. Right click over the JKEMLIST program and select Annotations then Add Annotation.

3. In the Create Annotation dialog, add a Title and a Main Text as described below or choose your own.
4. To add searchable keywords for this annotation, click on the Manage button.

5. In the Manage keywords dialog, select all the keywords on the Available keywords window and add them to the Selected keywords window using the >> and click OK.

6. Click Save.
7. You can now again right click over the **JKEMLIST** module and select **Annotations** then **View Annotations**

8. Remove **JKEMLIST** from the **Resource Name** field and click **Search** to see the annotations.

9. Any of the annotations can be modified or deleted using the icons above the **Annotations** box.
1.7 Switch to IDz Developer perspective and open JKEMLIST (Optional)

**Note:** This section is optional and requires a connection to a z/OS system. Please read through to get an idea of the integration with IBM Developer for z Systems (IDz).

Though not required to use Application Discovery, integration into the overall IBM DevOps solutions is important. When developers are ready to make the necessary application changes to support the business initiatives, they need to be able to seamlessly navigate, in context, from Application Discovery to IBM Developer for z Systems, and from there to the rest of the DevOps tools. In this section we will show the Application Discovery transition to IDz interface.

__1. Open up the Enterprise Developer perspective by clicking on the Open perspective icon on the toolbar__

__2. Select the Enterprise Development perspective and click OK__

__3. In the Remote Systems tab, under the previously defined Application Discovery dataset filter, locate the member JKEMLIST under the EMPOT.DEMO.DEV.COBOLO dataset.\n
4. Right click over it and select Application Discovery, then Analyze from the pop up menu.

5. Since JKELIST belongs to the JKEBank project, select it and click on the right arrow to move it to the Selected projects list and click Next.
6. On the Select available analysis dialog single click the JKEBank project to reveal the list of Available Analysis.

7. At this point the analysis can continue as if you had just started within the Application Discovery Analysis perspective. For example, under Mainframe Graphs, select Program Flow and click Finish.
8. You will be asked if you want to switch to the Application Discovery Analysis Browser perspective. Click Yes.
9. The resulting graph will be the same one you saw in step 1.3 (Understand a program's logic by creating a program flow chart) and analysis can continue as if you had started there.

This shows how seamlessly we can move from IBM Developer for System z to Application Discovery in order to analyze a program that will eventually be altered and implemented back into your source code manager and eventually production with the IBM DevOps solutions.

1.8 Summary

In this lab, you have:

- Learned about the main Application Discovery features as seen in the Eclipse interface
- Created a Transaction Callgraph for the JKEBank application at the highest level starting at the mainline transaction
- Drilled down into a program to analyze its variable usage
- Performed an Impact Analysis report on a specific COBOL program variable
Lab 2 – Hospital Batch Application Analysis

Hands on Lab Overview

The Hospital application is a batch COBOL based application. It also contains some DB2 usage. We will highlight how Application Discovery can also be a valuable analysis solution for batch workloads.

The data that applications process can come from various sources such as batch generated input datasets or databases. Getting a clear understanding of how programs and program flow relate to datasets being used in batch jobs is important in understanding how to make changes to applications, to improve application performance or to debug applications.

Application Discovery can also help analyze applications to understand how complex they are and therefore better estimate the time needed to modify them.

The features we will highlight in this lab are:

- Produce Cyclomatic Complexity Report
- Show Program Flow for MSTRUPDT
- Invoke Dataset Usage in Programs
- Invoke Dataset Usage in Jobs
- Show SQL Usage
- Show SQL Field Usage
2.1 Produce a Cyclomatic Complexity Report

1. Go back to the AD Explore tab and single click on the Hospital Application.

2. On the right pane you will notice that available reports and graphs appear for this application. Under Mainframe Reports expand Complexity Reports and double click on Cyclomatic Report.

   - Mainframe Reports
     - Complexity Reports
       - Cyclomatic Report
       - Halstead Report
       - Heuristic Complexity
       - Maintainability Index

3. In the resulting dialog move all the modules from the Available programs window to the Selected programs window by using the >> button and click Finish.
4. The report will be displayed in the AD Reports tab. Remember that this report is at an application level and is looking at the complexity of all the modules that were selected in the previous step.

5. Using the Next Page button on the toolbar (a yellow arrow), go to page 4 to see the Complexity percentage distribution graph. This graph gives an idea, to the developer, of how complex or not the overall application is.

Complexity Score Summary
Number of programs: 42
6. Now go to page 5 and see a complexity summary for all the modules that were analyzed. Let’s say that we really want to focus on the **MSTRUPDT** COBOL program. The name is a hyperlink to detail page so click on it.

### Program complexity scores

<table>
<thead>
<tr>
<th>Program</th>
<th>Score</th>
<th>Program</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADNTMSGH</td>
<td>113</td>
<td>getMemberInfo</td>
<td>1</td>
</tr>
<tr>
<td>MSTRUPDT</td>
<td>69</td>
<td>lstMemberContainer</td>
<td>1</td>
</tr>
<tr>
<td>TRTMNT</td>
<td>67</td>
<td>lock</td>
<td>1</td>
</tr>
</tbody>
</table>

7. You will be taken to a summary for the MSTRUPDT program that shows the complexity score as well as the meaning of the complexity score value.

Since the Cyclomatic report has advised us that this **MSTRUPDT** program is a difficult one we may want to go back and do further analysis, for example see the program flow of this module.

### 2.2 Show program flow for the most complex program

1. Switch back to the **Explore projects** tab and single click on **Hospital Application** to reveal the available reports on the right pane. Under **Mainframe graphs** double click on **Flow Chart**.

<table>
<thead>
<tr>
<th>Project</th>
<th>type</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hospital Application</td>
<td>z/OS</td>
<td>MVS; (ASSEM...</td>
</tr>
<tr>
<td>JKEBank</td>
<td>z/OS</td>
<td>MVS; (ASSEM...</td>
</tr>
</tbody>
</table>
2. Select MSTRUPDT from the available programs list and click Finish.

3. The resulting graph will be displayed in the AD graph tab. This graph is down to the actual program code and can be the first step for the developer to analyze and reduce the complexity of this particular COBOL program.
4. You can zoom in using the tools described earlier in the lab to focus on specific areas that could be simplified. We can see that there is a complex EVALUATE statement with multiple WHEN clauses assigning a numeric value based on the USA State 2 letter code. A more efficient way to process this requirement, for example some API integration, could lead to a simpler and more maintainable program.

2.3 Show dataset usage in jobs

Because this is a batch application, we could also look at some of the other Application Discovery analysis capabilities for batch impact such as dataset usage in jobs.

1. Switch back to the Explore projects tab and single click on Hospital Application to reveal the available reports on the right pane. Under Usage in Jobs double click on Dataset Usage in Jobs.

2. The AD Usage tab will be populated with the Dataset Usage in Jobs dialog. Let's say that we are looking for datasets that have RPT as part of the dataset name. In the Filter box, enter *RPT* and click Apply.
3. The Results dialog will show the datasets that matched the filter.

4. By clicking on the Save button, we can save this particular report for later use.

5. Expanding on the datasets in the search window, you can see the job using the dataset, the step in the job using the dataset, and the DD name using the dataset.
6. Click on the data set to reveal the reports available for this resource. Double click Dataset Flow from the right pane to see a dataset flow graph.

7. Because this is an output dataset, in the Data Flow dialog select Backward as the data flow direction and click Finish.
8. The resulting graph shows the target dataset where we requested the flow from, and all the other datasets that contribute to the contents of the report dataset through the batch Hospital Application batch application.

2.4 Show dataset usage in programs

Like the dataset usage in jobs demonstrated above, we can also get a view of how the datasets are being used by programs. This can give us a good indication on how the contents of the datasets are being created and will allow a developer to know what programs may need to change in order to change the data in a dataset.

1. Switch back to the AD Usage tab.
2. Click on the previous data set to reveal the reports available for this resource. Double-click **Dataset Usage in Programs** from the right pane to see a dataset flow graph.

The Results show, in detail, the data set in question; the DD name, the logical name for the DD in the COBOL program, the program name, the COBOL verb using the data set, and the actual COBOL line number in the COBOL program where the statement can be found.
3. You can double-click on the Line number, in this case MSTRUPDT Line 357, to reveal the actual COBOL code.

```cobol
 900 CLOSE-FILES.
 354 MOVE "900-CLOSE-FILES" TO PARA-NAME.
 356 CLOSE PATINS, PRSMSTR, PATERR, PATRPT.
 357 DISPLAY "FILES CLOSED".
 359 UNQ SET DISCONNECT-ACTION TO TRUE.
 360 UNQ CALL 'MQPUTSUB' USING LN-MQPUTSUB-FARN.
 361 UNQ IF LN-RETURN-CODE = 0
```
2.5  Show SQL usage

The Hospital Application also has some DB2 table usage. Just as with the dataset usage above, a DB2 Table usage report can help a developer understand, and pinpoint, where a change may have to be made in order to change how a COLUMN in a DB2 table is updated.

__1.  Switch back to the Explore projects tab and single click on Hospital Application to reveal the available reports on the right pane. Under Usage in Programs double click on SQL Table Usage.

__2.  In the resulting AD Usage tab, just click on the Apply button to reveal all the DB2 tables in use by the Hospital Application.

__3.  The results will be displayed by DB2 table name in alphabetical order.
4. We may be interested in re-ordering the results to get a different perspective of the data being displayed. By clicking on the **Reorder Current Results** icon, you can do just that.

5. Click on **Statement Type** and move it up to the top of the list and click **OK**.

6. The results will show the SQL statement type, the DB2 Table name associated with the SQL call, the program category (in this case COBOL), the program name and finally the line number in the source where the SQL statement can be found.

7. Expand the **DDS0001.DIAG_CODES** SQL Table.
8. You can double click on the COBOL line number for program **DALYEDIT** (in this case Line 545) to reveal the actual source code.

```
541 ****** EXEC SQL to get info from DB2
542   MOVE PRIMARY-DIAGNOSTIC-CODE TO
543   DIAG-CODE IN DCLDIAG-CODES.
544
545 EXEC SQL
546   SELECT DIAG_CODE INTO :DIAG-CODE
547   FROM DDS001.DIAG_CODES
548   WHERE DIAG CODE = :DIAG-CODE
549 END-EXEC
```

**END OF LAB**
Appendix A. Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been
estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Appendix B. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

<table>
<thead>
<tr>
<th>IBM</th>
<th>Application Discovery</th>
<th>CICS</th>
<th>Rational</th>
<th>System z</th>
<th>COBOL</th>
</tr>
</thead>
</table>

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product and service names may be trademarks or service marks of others.
© Copyright IBM Corporation 2017.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others.

回收标志