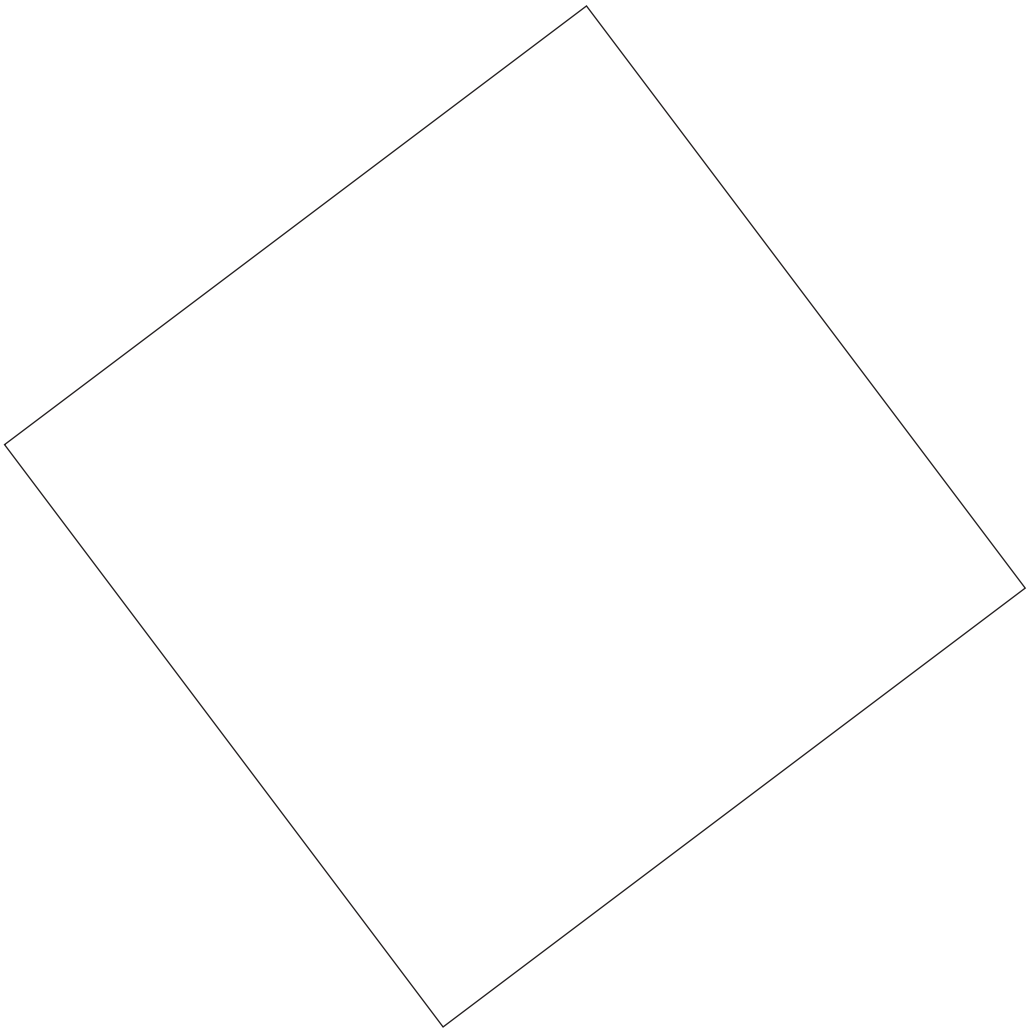
VisualAge for COBOL for OS/2

GC26-8749-00

Introducing Redeveloper Installing and Getting Started

Version 1 Release 2



IBM

VisualAge for COBOL for OS/2

GC26-8749-00

Introducing Redeveloper Installing and Getting Started

Version 1 Release 2

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page vii.

First Edition (July 1996)

This edition applies to Version 1 Release 2 of VisualAge for COBOL, Professional for OS/2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, W92/H3
P.O. Box 49023
San Jose, CA, 95161-9023
U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices . . . vii
Trademarks . . . viii

About This Book . . . ix
How to Use this Book . . . ix
Online Information . . . x

Chapter 1. Redeveloper - Product Overview . . . 1
Analysis and Maintenance Process . . . 1
Who Uses This Product . . . 1
What is Redeveloper? . . . 2
What is Application Understanding? . . . 4
What is Program Understanding? . . . 4
What is Program Conversion and Structuring? . . . 5
What is Redeveloper Integration? . . . 7
What is a Redeveloper Project? . . . 7
A Common Scenario . . . 8
Modifying the Program to Meet New User Requirements . . . 8

Chapter 2. Installing Redeveloper . . . 9
Before You Begin . . . 9
Installing Redeveloper . . . 10
Adding or Deleting Components . . . 13
If Something Goes Wrong . . . 13
If You Get an Error Message . . . 13
If the Component Requires Another Component . . . 13
If You Press Stop . . . 13
If the Installation Fails . . . 14
If You Can't Delete an Object Because It's in Use . . . 14
If Redeveloper Doesn't Appear in the Installation Utility . . . 14
If You've Tried Everything and It Still Doesn't Work . . . 15
If You Updated VisualAge for COBOL, Standard for OS/2 V1.2 . . . 15
Installation Verification . . . 15
Before Using Application Understanding . . . 16
Defining the Database and Binding Application Understanding . . . 18

Chapter 3. Getting Started Using Redeveloper Integration . . . 21
Redeveloper Integrated Project Environment . . . 21
Redeveloper Project Installation Requirements . . . 21
Application Understanding Project Setup Verification . . . 22
Program Understanding Project Setup Verification . . . 22
Creating Redeveloper Projects . . . 23
Adding Files to a Redeveloper Project . . . 24
Adding MVS Data Sets to the New Project . . . 24
Adding OS/2 Files to the New Project . . . 26

© Copyright IBM Corp. 1996

Using Actions in a Redeveloper Project 26
Using Application Understanding Actions 28
Using Program Understanding Actions 32
Using Program Conversion and Structuring Actions 36

Chapter 4. Understanding Redeveloper Tools 42
Accessing Redeveloper Tools 42
Starting Application Understanding from the Works Folder 44
Populating Your Inventory Database 45
Scanning the JCL Libraries 45
Retrieving the JCL Scan Output 46
Loading the Database with the JCL Scan Output 47
Viewing the Contents of Your Inventory Database 48
Opening an Inventory Window 48
Setting a Qualifier 49
Opening a Structure View of Components 49
Modifying Your Inventory Database 50
Creating a New Component 50
Deleting a Component 50
Starting Program Understanding from the Works Folder 50
Analyze Flow 53
Create Flow Graph 53
Show Flow Graph 53
Show Expanded Source 54
Show Control Flow Graph 55
Create and Show Flow Graph 56
Starting Program Conversion and Structuring from the Works Folder 56
Converting Your Program 58
Analyzing Your Program 59
Preparing Your Program to Improve Structuring 60
Structuring Your Program 61
Modularizing Your Program 61
Testing Your Program 62

Appendix A. Updating the CONFIG.SYS File 63

Glossary 64

Index 67

Figures

1. Redeveloper tools . . . 3
2. Redeveloper Initial Installation and Instructions window . . . 11
3. Application Understanding - Setup window . . . 19
4. Create New Redeveloper Project window . . . 23
5. Sample Redeveloper Project window . . . 27
6. Scan JCL action example . . . 29
7. Retrieve JCL action example . . . 30
8. Load Database action example . . . 31
9. Find in Inventory action example . . . 32
10. Create and show flow graph action on host-resident file . . . 33
11. Create and show flow graph action on workstation-resident file . . . 34
12. XSC file-type action example . . . 35
13. EU file-type action example . . . 36
14. Conversion and Structuring actions . . . 37
15. Works folder showing Redeveloper icon objects . . . 43
16. Application Understanding window . . . 44
17. Scan Request window . . . 46
18. Retrieve Request window . . . 47
19. Load Request window . . . 48
20. Program Understanding Folder -Icon View . . . 51
21. Basic steps for displaying graphical views using Program Understanding tool 52
22. Control flow graph view . . . 54
23. Expanded source view . . . 55
24. Program Conversion and Structuring main folder window . . . 57
25. Basic PCS steps for reengineering COBOL programs . . . 58
26. Convert options notebook . . . 59
27. Analyze/Structure options notebook . . . 60

© Copyright IBM Corp. 1996

Tables

1. Redeveloper Workstation Components Disk-Space Requirements 10
2. Examples of File Types and Redeveloper Actions You Can Perform On Them 28

© Copyright IBM Corp. 1996

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
555 Bailey Avenue, W92/H3
P.O. Box 49023
San Jose, CA 95161-9023

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

© Copyright IBM Corp. 1996

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

OS/2	IBM
VisualAge	CICS
DB2	DB2/2
MVS/ESA	

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

Welcome to the Redeveloper Installing and Getting Started book. This book will guide you through the installation process and help you get started analyzing and maintaining your COBOL code and applications.

How to Use this Book

This book is divided into chapters that correspond to the different tasks you, as a Redeveloper user, need to understand. These chapters include:

Chapter 1. Redeveloper - Product Overview
This chapter presents an overview of Redeveloper and it's features.

Chapter 2. Installing Redeveloper
This chapter describes Redeveloper installation tasks.

Chapter 3. Getting Started Using Redeveloper Integration
This chapter describes the tasks you are likely to do when analyzing and maintaining COBOL applications and program code in the Redeveloper integration environment.

Chapter 4. Understanding Redeveloper Tools
This chapter describes the individual tools that make up Redeveloper and explains their functions in greater detail.

Appendix A.
The appendix contains information on updates required to your CONFIG.SYS file before starting Redeveloper.

Glossary.
The glossary contains vocabulary terms used in this book..

In addition to this Installing and Getting Started book, online materials will help you use Redeveloper.

© Copyright IBM Corp. 1996

Online Information

Redeveloper Installing and Getting Started Online Book

Provides you with an online copy of this Installing and Getting Started book.

Online Tutorals

Provides you with step-by-step lessons on how to use Redeveloper. The following tutorials are available:

- Application Understanding Online Tutorial
- Program Understanding Online Tutorial
- Program Conversion and Structuring Online Tutorial

Online Help

Provides you with information on all functions and instructions for performing specific tasks. All Redeveloper components are supported by online help.

Task Helper

Provides you with a scenario on how to use the Redeveloper tools to perform a specific task.

Chapter 1. Redeveloper - Product Overview

Redeveloper is a product that works with IBM VisualAge for COBOL, Standard for OS/2 V1.2 (VisualAge for COBOL). With Redeveloper, you can re-engineer existing programs to produce applications better suited for future tasks. And when used with VisualAge for COBOL, Redeveloper aids in the analysis, maintenance, and reuse of your existing inventory of COBOL code. You can complete the full analysis and maintenance process using Redeveloper.

Redeveloper does not support object-oriented programming.

Analysis and Maintenance Process

The analysis and maintenance process generally begins with an evaluation of your applications to find the most suitable choices for redevelopment. You can use the Application Understanding tool to identify good redevelopment candidates and to predict the impact of changes you may want to make.

Next, you can use the Program Understanding tool's analysis functions to understand your programs. You can locate data flow in and between programs in executable units and see a program's relative complexity.

Your maintenance work may involve upgrading from older COBOL versions to the ANSI X3.23-1985 standard (the American National Standard for Programming Language COBOL, also known as ISO International Standard 1989-1985). The Program Conversion and Structuring tool automates much of the upgrade process by converting most of the COBOL syntax for you and converting EXEC CICS commands.

You may also want to structure a program because years of maintenance may have obscured whatever structure the code originally had. The Program Conversion and Structuring (PCS) tool helps you clean up code and identify dead code, which reduces a program's size and makes the code easier to maintain. You can structure your program either before or after converting. Program Conversion and Structuring gives recommendations for structure changes. You can allow PCS to structure the program for you, or do it yourself.

Redeveloper is a collaboration of tools that can be used separately or together. Each tool has its own icon in the Works folder (see Chapter 4 for information on accessing Redeveloper tools individually).

Who Uses This Product

The primary user of Redeveloper is someone whose job is to maintain legacy applications, for example, fixing 'Year 2000' problems. Typically, these applications work but need to be modified to handle future tasks or need to be upgraded to a higher level and structured. Redeveloper helps you analyze and understand these programs.

© Copyright IBM Corp. 1996

What is Redeveloper

What is Redeveloper?

Redeveloper is a development environment for analyzing and maintaining legacy COBOL programs and applications.

As depicted in Figure 1 on page 3, Redeveloper consists of tools that allow you to fully leverage existing COBOL resources in a client/server environment. This environment takes advantage of distributed processing to use the power of the host for gathering of host-based applications and the ease of use and convenience of the workstation for analysis, understanding, conversion and structuring of COBOL program code. Redeveloper consists of the following tools:

- Application Understanding (AU) tool

  This tool helps you understand your existing MVS batch jobs. For more information on the Application Understanding tool, see “What is Application Understanding?” on page 4.

- Program Understanding (PU) tool

  This tool helps you understand COBOL program logic, control flow, and data flow. For more information on the Program Understanding tool, see “What is Program Understanding?” on page 4.

- Program Conversion and Structuring (PCS) tool

  This tool helps you convert older COBOL programs to higher levels of COBOL and produce structured programs that are easier to understand, update, and maintain. For more information on the PCS tool, see “What is Program Conversion and Structuring?” on page 5.

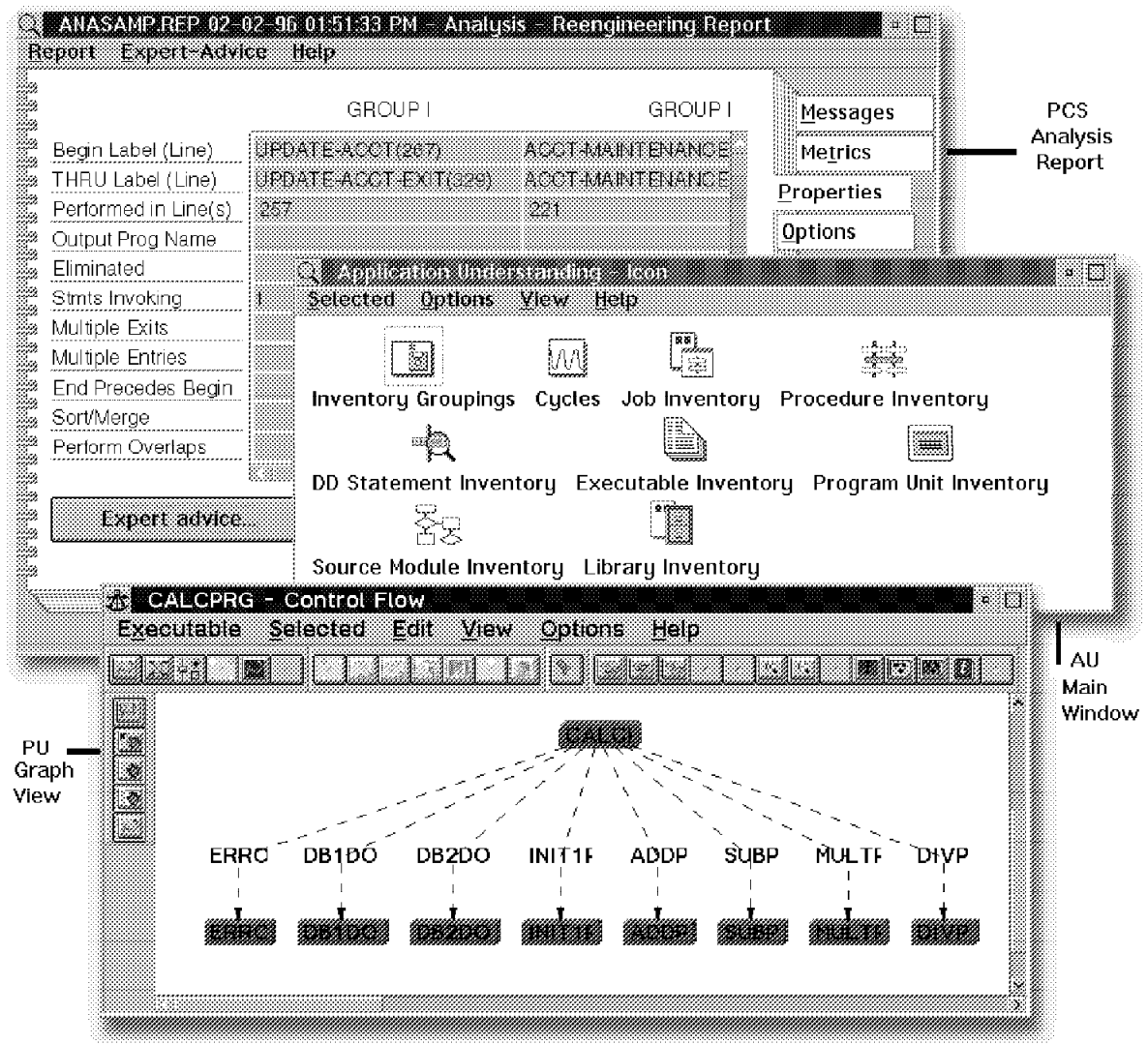


Figure 1. Redeveloper tools

With Redeveloper, COBOL program code and applications can be analyzed both before and after converting and structuring.

Redeveloper helps you maintain COBOL program code and applications by:

- Helping you understand the resources, structure, and flow of an application.
- Helping you understand the logic and flow of an application program.
- Providing a robust set of program views that you can quickly request and see. You can view programs as graphical nodes, code listing, and data element listings.

What is Redeveloper

- Enabling you to quickly create new program views and to save them.
- Helping you perform impact analysis on proposed changes to the program code and tracking impacts through data redefines and procedure parameter passing.
- Helping you convert COBOL programs. to higher levels of COBOL.
- Helping you structure COBOL programs.

What is Application Understanding?

When maintaining applications, it is useful to understand how they are constructed, where they used, and what files they process. The Application Understanding (AU) tool helps you do this by providing a capture facility to extract key information from your existing inventory of applications and a view facility that presents this information to you in easy to understand graphic windows. Using these facilities, you can better understand the impact of a suggested change.

With the capture facility, you can initiate the scan of JCL in you MVS job libraries. If you have installed the remote capability, the request is sent directly to MVS and you are notified when the scan is complete. Then you can load the JCL information into the Inventory database on your workstation.

With the view facility, you can see all the jobs/jobsteps that process a particular file or execute a particular program, for example. Or you can see all the jobs that make up an Inventory Group, or all the job steps in a given job. From one view you can also go directly to another. For example, if you are viewing a job, you can expand the view to see the job steps and then expand again to see the `dd` statements of a job step.

Note: Application Understanding requires the allocation of datasets on MVS before the capture facility can be properly executed. See “Before Using Application Understanding” on page 16.

What is Program Understanding?

The Program Understanding (PU) tool helps you understand complex COBOL software applications. You can analyze and extract information about an entire application and examine it on the workstation using an OS/2 graphical user interface to show both graphical and textual views of the physical design of the application.

Graphically, you can explore various views of the physical design of the program. At the highest level, you can discover the relationships between the programs that compose the entire application. A mid-level view displays the calling structure between the routines within a program or executable, including routines external to a program. At the lowest level, you can examine the internal control flow of each program.

At any time, you can reduce the scope of your analysis by restricting the analysis to only view areas of interest to you.

By default, multiple views are linked. A linked view depicts the association between the compiled source code and the graphical control flow representations of the program.

Textually, you can be presented with hypertext cross-reference information that details program usage, data structures, symbol usage, and COPY library usage. You can use the expanded source code browser view as a launching pad for navigation within and across programs.

You can request dataflow analysis on any user defined variable at any point in the program. You can also view all instructions that use the selected variable.

Program Understanding technology can:

- Provide complete user defined high-, medium-, and low-level information about large-scale software applications.
- Provide dataflow analysis.
- Provide program slicing to view business logic.
- Reduce the time application programmers need to become familiar with large-scale applications.
- Facilitate the migration of applications to the latest technology, such as client/server computing.
- Reduce errors that occur when maintenance tasks are performed manually on complex applications.
- Improve the productivity of software maintenance programmers.

Program Understanding does not support IMPLICIT transfers of control to COBOL declaratives. It does, however, support EXPLICIT transfers of control.

What is Program Conversion and Structuring?

The Program Conversion and Structuring (PCS) tool helps streamline the maintenance of COBOL programs by helping you convert your program code to higher levels of COBOL and transform unstructured programs into structured form automatically. The objective of Program Conversion and Structuring is to extend the life of your COBOL programs by making them easier to understand, debug, modify, and reuse.

The PCS tool is based on the technology that is currently available in the CCCA and COBOL/SF products.

Conversion

With Program Conversion and Structuring, you can automate the conversion of the following input languages:

- DOS/VS COBOL
- OS/VS COBOL LANGLVL(1)
- OS/VS COBOL LANGLVL(2)
- VS COBOL II Release 1 and Release 2
- VS COBOL II Release 3 (CMPR2)

You can convert any of the input languages listed above to any of the following output languages:

What is Redeveloper

- VS COBOL II Release 3
- COBOL for MVS & VM, VisualAge for COBOL

The Conversion function eliminates user-defined name and reserved word conflicts, flags language elements it cannot convert, and produces a log of information that helps you assess the conversion and track statistics about converting your COBOL inventory.

Conversion reports consist of the following:

- Convert Log, which contains the following pages:

  Status page
  Shows the status of each converted program, including the date each was most recently converted.

  Options page
  Shows the options used for each converted program.

  Files page
  Lists a cross reference of what data files are used by what COBOL programs, indicating those that do not need conversion and those that will need to be defined or checked for migration.

  Copy Library (CopyLIB) page

  Provides a cross reference of what copy modules are used by what COBOL programs; can be sorted by copy names or by program names. This page is only available if you convert a program containing COPY statements.

  Call page
  Provides a cross reference of what call statements are used by what COBOL programs; can be sorted by call names or by program names. This page is only available if you convert a program that has CALL statements.

- Convert Results file

  The Convert Results file provides a detailed report of the converted COBOL program.

Analysis

With Program Conversion and Structuring (PCS), you can analyze your program before structuring it to find and fix programming errors, and to identify and remedy complex PERFORM procedures before you run the final generation. PCS reports on the complexity of your program, and provides expert advice that tells you how to prepare your program for structuring.

Analysis reports include:

Reengineering Report
A detailed analysis of the PERFORM logic in the input program, as well as a set of input program complexity metrics.

Complexity Metrics Summary
A summary of the complexity metrics of all programs you have analyzed or structured.

Structuring

Using the information from analyzing your COBOL program, you can optimize the structuring of your program. Then you can structure and produce reports that help you understand the structured output program and how it relates to the input program.

Structuring reports include:

Cross Reference Browser
A window for cross-referencing between the Input COBOL Report and the Output COBOL Report after Program Conversion and Structuring finishes restructuring your input program.

Reengineering Report
A detailed analysis of the PERFORM logic in the input program, a set of complexity metrics, structure charts of the output program, advice to help modularize the output program, and sections describing various properties of the output program.

Complexity Metrics Summary
A summary of the complexity metrics of all programs you have analyzed and structured.

What is Redeveloper Integration?

You can integrate Redeveloper with VisualAge for COBOL through the project environment provided by WorkFrame. The basic maintenance functions provided by Redeveloper tools can be integrated and made available to you as actions invoked on objects in projects. This provides a seamless way to access these tools from the same kind of project view as used in VisualAge for COBOL.

What is a Redeveloper Project?

A Redeveloper project encapsulates and organizes the different actions, types, and variables needed for Redeveloper tools operation.

Redeveloper allows you to create projects that match your working environment. You can create a single project that organizes all the actions for all Redeveloper tools; or you can create different projects that organize individual Redeveloper tool actions or create projects that work with specific file types. For example, you may want to create a project that organizes Redeveloper tools' actions for working with host-resident files only. Likewise, you may want to create another project that just works with workstation-resident files.

For additional information on creating a Redeveloper project, see “Creating Redeveloper Projects” on page 23.

A Common Scenario

This section contains a common Redeveloper scenario. This scenario provides step-by-step information on how to approach and solve the reengineering task using the Redeveloper tools in conjunction with VisualAge for COBOL.

Modifying the Program to Meet New User Requirements

To modify a program to meet new user requirements:

1. Identify impact of affected application program:
   a. Perform impact analysis, using Application Understanding to identify application program parts affected by the new user requirement for the current production environment.
   b. Understand the COBOL modules and programs that make up the affected application program parts, using Program Understanding.
2. Prepare to update program:
   a. If needed, convert the affected programs, using Program Structuring and Conversion.
   b. If needed, analyze the affected programs to understand how to best structure them and then structure them using Program Conversion and Structuring.
   c. Understand the converted/analyzed modules and programs, using Program Understanding:
3. Make changes:
   a. Edit, compile, debug the programs identified as impacted by the fix, using other VisualAge for COBOL components.
   b. Test the programs identified as impacted by the fix, using other VisualAge for COBOL components.
   c. Possibly structure the programs identified by the fix again, using Program Conversion and Structuring.
   d. Promote to production.

The online Redeveloper Task Helper provides a demonstration of a way to use Redeveloper in an integrated environment with VisualAge for COBOL to add new function to legacy code.

Chapter 2. Installing Redeveloper

This chapter describes the installation process for Redeveloper. We recommend that you read this chapter before installing Redeveloper. It contains the following sections:

- Before You Begin
- Installing Redeveloper
- Adding or Deleting Components
- If Something Goes Wrong
- Installation Verification
- Before Using Application Understanding

Before You Begin

Before you begin to install Redeveloper, make sure that your workstation meets the following hardware and software requirements.

Processor
An Intel 80486-based minimum (Pentium recommended) 66 MHz IBM workstation or equivalent.

Display
VGA minimum (SVGA recommended)

RAM
24MB minimum (32MB recommended)

Disk Space
60MB; additional disk space required for VisualAge for COBOL, Standard for OS/2 V1.2 and DB2/2.

Operating System
OS/2 Warp Version 3.0 or subsequent releases.

Other Software

- VisualAge for COBOL, Standard for OS/2 V1.2 including the WorkFrame component and Remote Edit/Compile/Debug component.
- DB2/2 Version 2.1 (FixPak 8090 required) with APAR JR08999
- Communications Manager Version 1.11 or later
- MVS/ESA System Product - JES2 or JES3 Version 5 or later

Specific Redeveloper component disk space requirements are summarized in Table 1 on page 10.

Installing Redeveloper

Table 1. Redeveloper Workstation Components Disk-Space Requirements

REDEVELOPER WORKSTATION COMPONENT	DISK SPACE REQUIRED (MB)
Program Conversion and Structuring	20
Application Understanding	5
Program Understanding	10
Redeveloper Integration	4
Application Understanding Tutorial	3
Program Conversion and Structuring Tutorial	3.1
Program Understanding Tutorial	5

Note: If you install to a FAT partition, the space required for each component may be larger than the amount shown.

You can install Redeveloper to any drive and directory on your workstation. By default, the components are installed on the **c** drive.

Installing Redeveloper

This section provides instructions for installing Redeveloper. Make sure you have read "Before You Begin" on page 9.

If you get error messages during the installation procedure, select the **Help** push button in the message window for help on what to do. See "If Something Goes Wrong" on page 13.

Install Redeveloper from the CD-ROM as follows:

1. Insert the Redeveloper CD-ROM.
2. From an OS/2 command line, enter the following command:

   `d:\install`, where `d` is the drive containing the Redeveloper CD-ROM.

   As depicted in Figure 2 on page 11, the Redeveloper initial installation and instructions window appear.

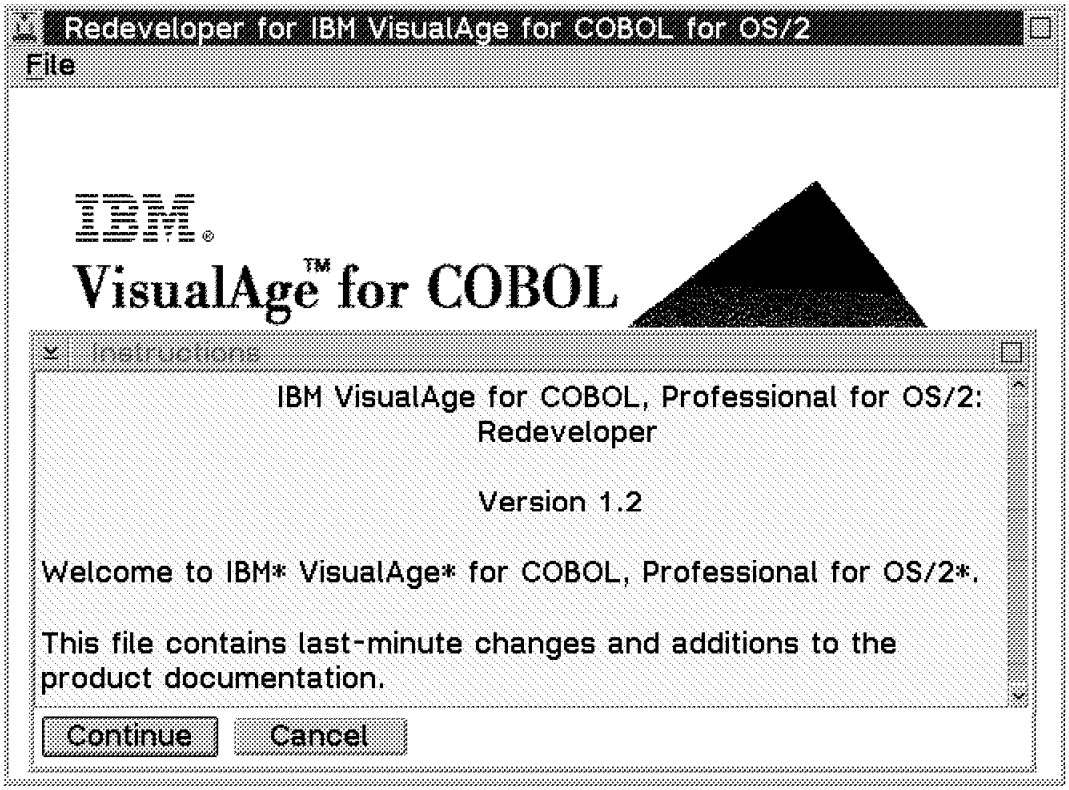


Figure 2. Redeveloper Initial Installation and Instructions window

3. Scroll through the **Instructions** window to read all the text in the READ.ME file. This file contains the latest information about Redeveloper changes or restrictions, including any that affect installation.

   If the *READ.ME* file differs from this install section, the *READ.ME* file is correct. You may need to move the instructions window to see the installation window.

   Note: The *READ.ME* file is located in the same directory as the install program. It can also be accessed using the OS/2 system editor.

4. Select **Continue** in the Instructions window, and follow the installation instructions as they are displayed on the monitor.

   You can select the components you want to install; they are:

   - Program Conversion and Structuring
   - Application Understanding
   - Program Understanding
   - COBOL Redeveloper Integration
   - Application Understanding Tutorial
   - Program Conversion and Structuring Tutorial
   - Program Understanding Tutorial

Installing Redeveloper

You do not have to install all the components at once; you can run the install program again at a later time. You must select at least one component to continue with the install program.

Allow the installation program to automatically update your CONFIG.SYS file. This is the default setting. If you do not tell the installation program to update your CONFIG.SYS file, you must make the changes yourself before you reboot and run Redeveloper. See Appendix A for information on updates required to your CONFIG.SYS file before running Redeveloper.

Note: The installation program creates a file named **CONFIG.ADD** for your use if you did not allow the installation program to automatically update your CONFIG.SYS file. You can examine the CONFIG.ADD file to make the necessary changes to the CONFIG.SYS file.

Although not required, you can install Redeveloper in the same drive and directory (IBMCOBOL) as VisualAge for COBOL, Standard for OS/2 V1.2.

Some components require other Redeveloper components for them to work correctly. The following lists components that require other Redeveloper or VisualAge for COBOL, Standard for OS/2 V1.2 components:

- Redeveloper Integration component requires one of the Redeveloper components (Application Understanding, Program Understanding, or Program Conversion and Structuring) and WorkFrame.

  We recommend that you install the Redeveloper Integration component. This provides a project-oriented maintenance solution function for Redeveloper (see Chapter 2 for information on how to get started using Redeveloper in the project environment). WorkFrame can be installed from the VisualAge for COBOL, Standard for OS/2 product.

The following lists components that require other IBM products:

- Application Understanding requires DB2/2.

  To run Application Understanding, DB2/2 must be installed, a database created, and the BIND application must be run on the database. After installing Application Understanding and DB2/2, you will have to create the database and BIND application before running Application Understanding. See “Defining the Database and Binding Application Understanding” on page 18.

- Program Conversion and Structuring may require the VisualAge for COBOL, Standard for OS/2 V1.2 (VisualAge for COBOL) compiler.

  If you plan to convert programs with (CICS) BLL cells in the LINKAGE SECTION, the VisualAge for COBOL compiler must be installed.

  Note: This means that a database must be available for the conversion tool to connect to. If you can manually comment out the DB2 content in the LINKAGE SECTION where BLL cells are present, this connection to a database is not required.

5. When the install program has finished, shut down your system and restart it so installation changes can take effect.

Adding or Deleting Components

Once Redeveloper has been installed, you can go back to the install program to install additional components or to delete components. If you need to reinstall a component, you can delete it and then add it again.

You can use one of two ways to add or delete components:

1. Run the original install program interactively.
2. Use the Installation Utility icon object from the VisualAge for COBOL **Works** folder.

If you are adding components, make sure you have access to the CD-ROM directory where the Redeveloper installation files reside.

Use the original installation program supplied on the CD-ROM directory and follow the instructions as they are displayed on the monitor to add or delete components. See “Installing Redeveloper” on page 10

Note: If you are deleting components, you may get a message indicating that some files could not be deleted because they were in use. Select **OK** to continue. When you shut down and restart your system, these files will be deleted.

If Something Goes Wrong

This section describes what to do if you have problems running the install program. Online help is also available for error messages, both from the **Help** push button on the message window, or from the index or contents of the install help.

If You Get an Error Message

Select the **Help** push button in the error message window for information on how to correct the problem. Also note the message number and the file name where the error occurred, if one is indicated.

If the Component Requires Another Component

Some of the Redeveloper components require that other components be installed for them to work correctly. If you select a component to install without selecting its prerequisite, a message may prompt you to select the prerequisite also.

Select the prerequisite component, as indicated by the message. Note that the prerequisite component may itself have prerequisites.

If You Press Stop

If you press the **Stop** button on the Install progress window, the install program immediately stops and displays a window that asks if you want to back out of the installation so far. You cannot resume the install program from this point. You must end it and start again.

If Something Goes Wrong

In the confirmation window:

- Select **Yes** to undo everything that the install program has done up to that point. We recommend you choose this option. Any desktop objects are deleted.
- Select **No** to leave the files that have already been installed.

  Caution: Selecting **No** may cause some systems to become inoperable.

Once the program has ended, you can restart it from the beginning.

If the Installation Fails

If the installation fails, a window is displayed that asks whether you want to back out of the installation:

- Select **Yes** to undo everything that the install program has done up to that point. We recommend you choose this option.
- Select **No** to leave the files that have already been installed.

  Caution: Selecting **No** may cause some systems to become inoperable.

The most likely reason for the failure is that not enough disk space is available. To solve this problem:

- Make more room on your target drive.
- Choose a different target drive.
- Select fewer components to install.

If disk space is not a problem, run CHKDSK on the:

- Drive where OS/2 is installed
- Target drive to ensure there are no file system problems.

If you have tried all of these suggestions and the install program still fails, contact IBM Service and Support.

If You Can't Delete an Object Because It's in Use

When you delete a component, if any objects or files are in use, they are not deleted. A message informs you when this happens. The rest of the component and its desktop objects are deleted.

After you shut down and restart your system, these objects should be deleted automatically.

If Redeveloper Doesn't Appear in the Installation Utility

If you start the Installation Utility and Redeveloper is not listed in the window, select **Current Catalog** from the **View** menu to display it.

If You've Tried Everything and It Still Doesn't Work

If the install program continues to fail after you've tried everything suggested by the error message help and this section, contact IBM Service and Support.

Be sure to inform them what error messages you see, and what file names, if any, are given in those messages.

If You Updated VisualAge for COBOL, Standard for OS/2 V1.2

If you have updated or reinstalled VisualAge for COBOL, Standard for OS/2 V1.2 (VisualAge for COBOL) after you installed Redeveloper, you may need to recreate the **COBOL Redeveloper Project** entry in your **COBOL Project Smarts** catalog by running DCZWFINS.CMD. This command file is in the drive and directory containing Redeveloper. You can run this command at the prompt of an opened OS/2 window.

Installation Verification

To determine that the installation process successfully installed your Redeveloper components, verify the following:

Note: All the following verification checks are accomplished from the opened **VisualAge for COBOL - Icon view** window. You can open this window by double-clicking with the mouse on the **VisualAge for COBOL** icon located on the desktop.

- All Redeveloper components and tutorials reside in the VisualAge for COBOL **Works** folder. Double-click on the **Works** folder to open it and verify that the following Redeveloper components exist for the components you installed:

  Application Understanding

  Application Understanding Setup

  Program Conversion and Structuring

  Program Understanding

  Application Understanding Tutorial

  Program Conversion and Structuring Tutorial

  Program Understanding Tutorial

  Redeveloper READ.ME file

- If you installed the Redeveloper Integration component, verify that:

Before Using Application Understanding

  - the **Create New Redeveloper Project** has been created.
  - the **Master Project** for each Redeveloper component you installed is in the **Works** folder.
  - the **Redeveloper Project** is in the **Templates** folder.
  - the Redeveloper **Sample Redeveloper Project** is in the **Samples** folder.
  - from inside the **Works** folder, there is a COBOL Redeveloper Project entry in the **COBOL Project Smarts** catalog folder.

- for Redeveloper online information, verify that:
  - the **Task Helper** contains Redeveloper Task Helper online help information.
  - the online Redeveloper Getting Started information and component-specific help information for each component installed is in the **Information Notebook**
  - the Redeveloper How To Start information is presented when you click on the **How To Start** icon.

Before Using Application Understanding

The host component of Application Understanding is shipped with the workstation parts. These parts must be manually uploaded to the host. These parts do not need to be uploaded to the host again once done successfully.

The **MVS** subdirectory on the drive and under the main directory that you installed Application Understanding contains the following host parts:

ARDV000.CLI
This clist must be added as member ARDV000 to the userid.ARD.CLIST PDS of users wanting to submit JCL scan from the workstation. This PDS must be available to the APPC transaction defined for IBMCOBOL.

ARDV000.JCL
This must be added to a JCL PDS available to users wanting to run the JCL scan in batch on MVS.

ARDV000.LOD

This file is a packed loadlib. It must be sent to MVS and unpacked to a PDS. This load library must be available to the APPC transaction defined for the drive containing Redeveloper. It is also used to run JCL scan manually.

1. **From the MVS Environmment**

   Allocate a seq file on the MVS host to receive the packed part. The datasets are Record Format **FB**, Record Length **1024**, and Blocksize **6144**.

   Note: If you allow the send command to allocate the MVS file for you, it may be allocated with a blocksize unacceptable to the unpack utility.

2. **From the Workstation Environment**

   To send the packed part (ardv000.lod) to MVS, open an OS/2 window and position to the drive and **MVS** directory by entering: `d:`, where `d` is the drive containing Redeveloper.

   ```
   cd ibmcobol\mvs
   ```

   The above command will make **MVS** the current directory.

   Then issue the following command:

   ```
   send ardv000.lod b:'userid.packed.ardv000.lod'
   ```

   where:

   ```
   send
   ```

   is the Communication Manager command

   ```
   ardv000.lod
   ```

   is the part from the MVS directory,

   ```
   b:
   ```

   is your Communication Manager session for MVS, and

   ```
   userid.packed.ardv000.lod
   ```

   is the MVS seq file you allocated in step 1

3. **From the MVS Environmment**

   To unpack ardv000.lod, use the PDF option **U** of the **Terse - File/Expansion** command. The dataset name you use for Output Dataset Name must be a PDS.

   Note: If your output dataset does not already exist, use the **Output Explicit Alloc ==> YES** option to allocate it before the unpack is done.

   Use the default values to allocate the PDS (for example, Record Format=**U**, Record Length =**0**, Blocksize=**32760,** ,Device=**3380**, Space=**Tracks**, Primary Quantity=**7**, Secondary Quantity=**15**, Directory Blocks=**20**).

The following MVS datasets must be allocated before the JCL scan is run:

Note: The value .yyyy used in the following instructions represents 0 or more qualifier used in your MVS dataset names.

Before Using Application Understanding

CDIF PDS

userid.yyyy.CDIF (VB, 255, 23476) The first time you run the Application JCL Scan request you will need to specify this PDS name. (Size is determined by how many scan the user wishes to save. Each scan will generate a member that can be several times larger than the JCL being scanned because the JCL is exploded to include the JCL from called PROCs.)

ProcLib List

userid.yyyy. ARDLIB sequential data set (FB, 80, 3200) (1 track should be sufficient). Before you run a JCL scan, you must edit this dataset creating 1 line for each procedure library used by the JCL being scanned.

The following is an example file:

```
SYS1.ADPROC
SYS1.PP.PROCLIB
USERID.ABC.TESTPROC
```

Message Log PDS

userid.yyyy.MSG (VB, 255, 23476) The first time you run the Application JCL Scan request you will need to specify this PDS name. (Same size as userid.yyyy.CDIF)

The default number of DB2/2 applications you can have running at one time is 8. With Application Understanding, it is possible to have open as many windows as you wish. Each one is a DB2/2 application. If you wish to have more than 8 Application Understanding windows open at once, you must increase the maximum active applications value (maxappls) parameter of DB2/2. The methods for doing this is DB2/2 release dependent. See your DB2/2 documentation for instructions.

Defining the Database and Binding Application Understanding

Before running Application Understanding, you must do the following:.

1. Double-click on **VisualAge for COBOL** icon on your desktop and then double-click on **Works** folder.
2. Double-click on **Application Understanding Setup** folder.

   As depicted in Figure 3 on page 19, the **Setup** window appears.

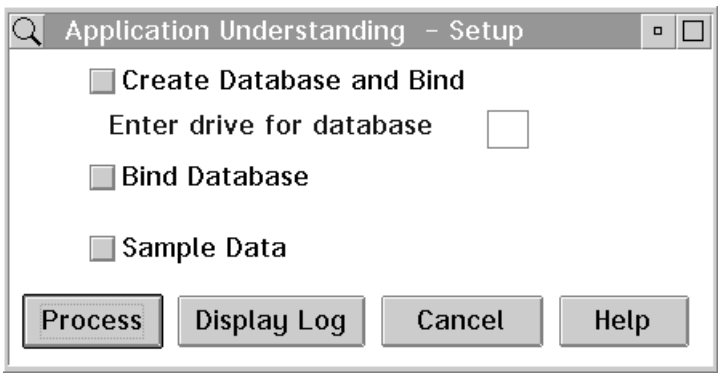


Figure 3. Application Understanding - Setup window

3. You can define the DB2/2 database by clicking on the **Create Database and Bind** check box; enter the drive for the database in the **Enter drive for database** entry field and select **Process**. If you do not supply a drive for the database, it will default to the drive where DB2/2 is installed.

   Note: If you have previously installed Redeveloper and defined this database, you can skip this step.

   You will need DB2/2 administrator authority to do this.

4. You can bind Application Understanding to the database defined in step 3 by clicking on the **Bind Database**. check box and select **Process**.

   You must bind Application Understanding to the database even though the database has been previously defined.

   Note: If you deleted Application Understanding at any time and then reinstalled it, you must bind Application Understanding to the database before running Application Understanding.

5. You can populate the database with sample data by clicking on the **Sample Data** check box and select **Process**.

Related Messages

1. Error message **SQL102N SQLSTATE=51007** when you first attempt to open the Application Understanding object.

   Solution: This message displays if you do not create the Application Understanding database and run the necessary bind. Use the Application Understanding Setup utility to create the database and run the bind. See “Defining the Database and Binding Application Understanding” on page 18 for instructions about using Application Understanding Setup.

2. Error message **SQL0805N SQLSTATE=51002** when you first attempt to run Application Understanding after installation or re-installation.

   Solution: This message displays if you do not bind Application Understanding to the database. Use the Application Understanding Setup utility to run the necessary

bind operation. See “Defining the Database and Binding Application Understanding” on page 18 for instructions about using Application Understanding Setup.

3. Information message **CODE PAM0031I No server object shown. You may not have started a CODE connection to the host.**

   Solution: This message displays if you check mark any type of MVS project type such as Application Understanding in the Create New Redeveloper Project window and have not configured APPC on your workstation. The message is only an information message and can be ignored for the moment, but you must configure the APPC connection (under CM/2) before you can use Application Understanding.

Chapter 3. Getting Started Using Redeveloper Integration

This describes the tasks you are likely to do when analyzing and maintaining COBOL applications and program code in the Redeveloper integration environment. It assumes that you have installed the product on your workstation and that you are ready to begin using Redeveloper. If you haven't installed Redeveloper yet, follow the instructions in “Installing Redeveloper” on page 10.

Note: This section assumes that the COBOL Redeveloper Integration component was installed. This component provides an integrated environment using WorkFrame whereby actions for Redeveloper tools can be executed.

What You Need to Know

You need to know how to use a mouse to manipulate windows, icons, and other objects on the OS/2 desktop. The **OS/2 Tutorial** explains how to use the OS/2 operating system and provides a hands-on overview of the basic tasks. It is normally located in the Information folder on your desktop.

Redeveloper Integrated Project Environment

Redeveloper is integrated with VisualAge for COBOL, Standard for OS/2 V2.1 (VisualAge for COBOL) through the project environment provided by WorkFrame. The functions provided by Redeveloper tools are integrated and available to users as actions invoked on objects in projects. This provides a seamless way to access these tools from the same kind of project view as used in VisualAge for COBOL.

Redeveloper Project Installation Requirements

The installation requirements for the Redeveloper project environment are described below.

Redeveloper Integration Component and WorkFrame

You can work in the Redeveloper project environment only if the Redeveloper Integration component is installed. If this is not installed, go to “Installing Redeveloper” on page 10 for installation information. The Redeveloper Integration component works with the WorkFrame component from the VisualAge for COBOL product. If you did not install this component, you must go back to VisualAge for COBOL and do a selective install of WorkFrame.

Working with Host (MVS) Applications and Programs

If you work with MVS applications and programs and want to add PDS members to your project, then the tools can work with MVS data and MVS actions in an integrated project environment. Application Understanding works with MVS data and actions. The Program Understanding tool works with MVS data if you are working with COBOL programs (ADATA files) from the host. Redeveloper tools that work with MVS data require APPC configuration and the **Remote Edit/Compile/Debug** component installed

© Copyright IBM Corp. 1996

from the VisualAge for COBOL for OS/2 product. You must also configure the APPC connection between your workstation and host (under Communications Manager/2) before you can use Application Understanding and Program Understanding working with MVS data.

Working with OS/2 Workstation Applications and Programs

If you only work with OS/2 files and want them added to your project, then the tools can work with local OS/2 workstation programs and files in an integrated project environment. The Program Understanding tool works with COBOL programs (ADATA files) residing on your OS/2 workstation.

Application Understanding Project Setup Verification

Before using the Application Understanding project actions on JCL files accessed from a host, verify the setup of Application Understanding Master Project in the **Works** folder as follows:

1. Open the **VisualAge for COBOL folder**, then open the **Works** folder.
2. Highlight the **Application Understanding Master Project** and select **Settings** from the pop-up menu.
3. Select **Location** in the notebook, then select PAM/MVS at the bottom of the notebook page.
4. Select **Options .....**.
5. In the **Options** notebook, select **Applications**.
6. Type the following three file names into the top entry field and click on the **Add** push button for each, if they are not already in the list box.

```
DCZWFAUE.EXE
DCZWFAUR.EXE
DCZWFAUS.EXE
```

Program Understanding Project Setup Verification

Before using Program Understanding project actions on SYSADATA files accessed from a host, verify the setup of Program Understanding Master Project in the **Works** folder as follows:

1. Open the **VisualAge for COBOL folder**, then open the **Works** folder.
2. Highlight the **Program Understanding Master Project** and select **Settings** from the pop-up menu.
3. Select **Location** in the notebook, then select PAM/MVS at the bottom of the notebook page.
4. Select **Options .....**.
5. In the **Options** notebook, select **Applications**.

6. Type the following file name into the top entry field and click on the **Add** push button for each, if it is not already in the list box.

```
DCZWFHAF.EXE
DCZWFHCS.EXE
```

Creating Redeveloper Projects

Before you start using any of the Redeveloper tools in the integrated environment, you must create a Redeveloper project. A Redeveloper project encapsulates and organizes the different actions, types, and variables needed for Redeveloper tools operation. Use the following steps to create a Redeveloper project:

1. From the VisualAge for COBOL Icon view, open the **Create New Redeveloper Project**.

   As depicted in Figure 4, the **Create New Redeveloper Project** window opens.

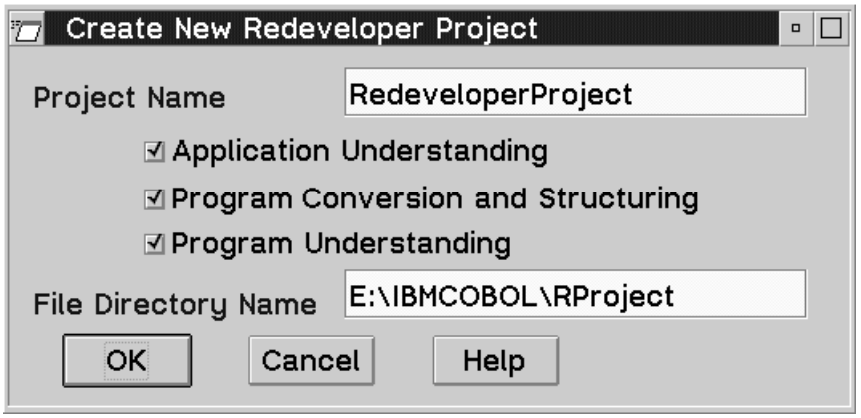


Figure 4. Create New Redeveloper Project window

2. In the **Project Name** field, use the default project name (RedeveloperProject) or enter the desired project name for the new project.
3. Select the check box of the tool or tools you want to use.
4. In the **File Directory Name** field, enter the working directory where your COBOL programs, files, and data are located. As depicted in Figure 4, the default File Name Directory is **E:\IBMCOBOL\RProject**. If your COBOL programs are located in directory **D:\COBOL1**, enter

```
D:\COBOL1
```

5. Select **OK**

   Using the default setup, this will create a project called **RedeveloperProject** on the desktop.

   Note: RedeveloperProject is the default project name. You can choose to call the project by any name.

Adding Files to a Redeveloper Project

A Redeveloper project inherits all the required actions from the COBOL master project and from each Redeveloper master project for the tool which is selected.

Adding Files to a Redeveloper Project

You can add all the files necessary to the project before you perform integrated Redeveloper tool actions in the Redeveloper project. Likewise, you can also add files to the project whenever you need them. You can add the following types of files:

- MVS files (for projects that use MVS PDS members)
- OS/2 files

Note: See "Redeveloper Project Installation Requirements" on page 21 for information on Application Understanding, Program Understanding, and Program Conversion and Structuring types of files.

After opening a Redeveloper project window, perform the steps below:

Adding MVS Data Sets to the New Project

1. Select **Settings** from the **View** toolbar menu.
2. Select **Location** with the left mouse button.
3. Select **PAM/MVS** at the bottom of the notebook page.
4. Use the **Find...** button and select a fully qualified name of a host MVS data set from the **Tree View of Host File System** window (Add, Replace, or Delete). Alternately, if you already know the MVS data set name, you can manually enter a fully qualified name of MVS data set, one per line, in the following format then close the notebook:

   ```
   <ServerName>DataSetName(*)
   ```

   where `<ServerName>` is the name of the connection to your server (host) and `DataSetName` is the name of your PDS data set for the JCL. The asterisk `(*)` indicates all members in the data set.

   Note: If you experience problems connecting to your MVS PDS members, ensure that the **Remote/Edit/Compile** component was installed form the VisualAge for COBOL for OS/2 V1.2 product and that you have successfully configured APPC on your workstation.

   An Example for adding Application Understanding MVS-type files to your Redeveloper project follows:

   ```
   <MVSAPPC>CDXMTS1.DCZ.JOBLIB(*)
   <MVSAPPC>CDXMTS1.DCZ.CDIF(*)
   <MVSAPPC>CDXMTS1.DCZ.MSG(*)
   ```

   In the above example, `<MVSAPPC>` is the server name. `CDXMTS1.DCZ.JOBLIB` is the PDS data set name for MVS JCL. `CDXMTS1.DCZ.CDIF` is the PDS data set name for JCL output on MVS. `CDXMTS1.DCZ.MSG` is the PDS data set name for scan process message.

The above PDS data set should already be created on MVS. In the above example, when Redeveloper is used for the first time, **JOBLIB** data set should have some MVS JCL, **CDIF** data set does not have any member and **MSG** data set does not have any member. Data set qualifier for scan JCL should be **JOBLIB**. For scan JCL output, the qualifier should be **CDIF**. For holding any message of the JCL scan or retrieve operation, the qualifier should be **MSG**.

If the APPC configuration is correct, you will see the icon file name for your host in your project. For example, *JOBLIB(IBMJCL)*.

Note: The above example uses a data set qualifier type called **JOBLIB**. Your MVS host job environment may use other data set qualifier names. You can add a filter for a data set qualifier name used by your MVS host environment by:

- Select **View** from the menu bar of an opened Redeveloper project window, then select the **Tools setup** choice from the drop-down menu.

  The **Tools setup** window opens.
- Select **View** from the menu bar of the opened Tools setup window, then select the **Types** choice from the drop-down menu.
- Scroll to the **MVSJCL** name type and select the **Add..** choice from the pop-up menu. You can enter your data set qualifier as a filter-name in the opened **Add Type** window.

An Example for adding Program Understanding MVS-type files to your Redeveloper project follows:

```
<MVSAPPC>CDXMTS1.IWZ.SYSADATA(*)
```

or

```
<MVSAPPC>CDXMTS1.IWZ.ADATA(*)
```

or

```
<MVSAPPC>CDXMTS1.IWZ.ADT(*)
```

In the above example, `<MVSAPPC>` is the server (host) name and `CDXMTS1.IWZ.SYSADATA` (or `CDXMTS1.IWZ.ADATA` or `CDXMTS1.IWZ.ADT`) is the PDS data set name.

The above PDS should already be created on MVS.

If the APPC configuration is correct, you will see the icon file name for your program in your project.

Note: If you get the message **No Server object shown. You may not have started a CODE connection to the host.**, you may not have a CODE connection between your workstation and host. You can start the CODE connection as follows:

1. Enter the **code** command at the prompt of an opened OS/2 window.
2. In the opened **CODE** window, select **Server list** menu choice from the**Windows** menu.

   The **Servers** window opens.

Using Actions in a Redeveloper Project

3. In the opened **Servers** window, enter the server name (the name of your connection to the host) in the **Server name** field.

   Ensure that the LU (APPC) check box is checked.

4. Close the **CODE** window.
5. After opening a Redeveloper project window, perform the steps below:
   a. Select **Settings** from the **View** toolbar menu.
   b. Select **Location** with the left mouse button.
   c. Select **PAM/MVS** at the bottom of the notebook page.
   d. Select **Options...** push button.
   e. Select **Configure...** push button and, at the **Servers** window, enter the server name in the **Server name** field if not already displayed.
   f. Select a fully qualified name of a host MVS data set from the **Tree View of Host File System** window.

Adding OS/2 Files to the New Project

Normally, all OS/2 files that you want to perform Redeveloper actions on would already reside in a working directory and its location identified when a Redeveloper project was created. However, you can add files to a Redeveloper project that reside in other directories.

After opening a Redeveloper project window, perform the steps below:

1. Select **Settings** from the **View** toolbar menu.
2. Select **Location** from the cascaded menu.
3. If you have set up your project to also work with MVS files, select **OS/2** at the bottom of the notebook page.
4. Enter a fully-qualified path of the location of a file, one per line.

   An example for adding OS/2 files that Program Conversion and Structuring or Program Understanding can perform actions on follows:

   ```
   G:\MYDRCTRY
   ```

   In the above example, `G:\` is a drive and `MYDRCTRY` is a directory where COBOL programs are located.

Using Actions in a Redeveloper Project

As depicted in Figure 5 on page 27 and summarized in Table 2 on page 28, you can perform integrated Redeveloper actions on files in a single Redeveloper project. The actions available for an object in a Redeveloper project are dependent on the type of file the object represents. You can select the appropriate object's icon and the action from the pop-up menu or from the choices in the **Selected** menu. Note that, as in Figure 5 on page 27, you can display all file types in your Redeveloper project by

selecting asterisk (*) in the file filter-type entry field. You can change this entry to only display specific file types. For example, changing the file filter-type to **MVS**, results in only displaying host-resident files.

Note: The **MVS** file filter-type only applies to actions performed by the Application Understanding and Program Understanding tools that work with host-resident files.

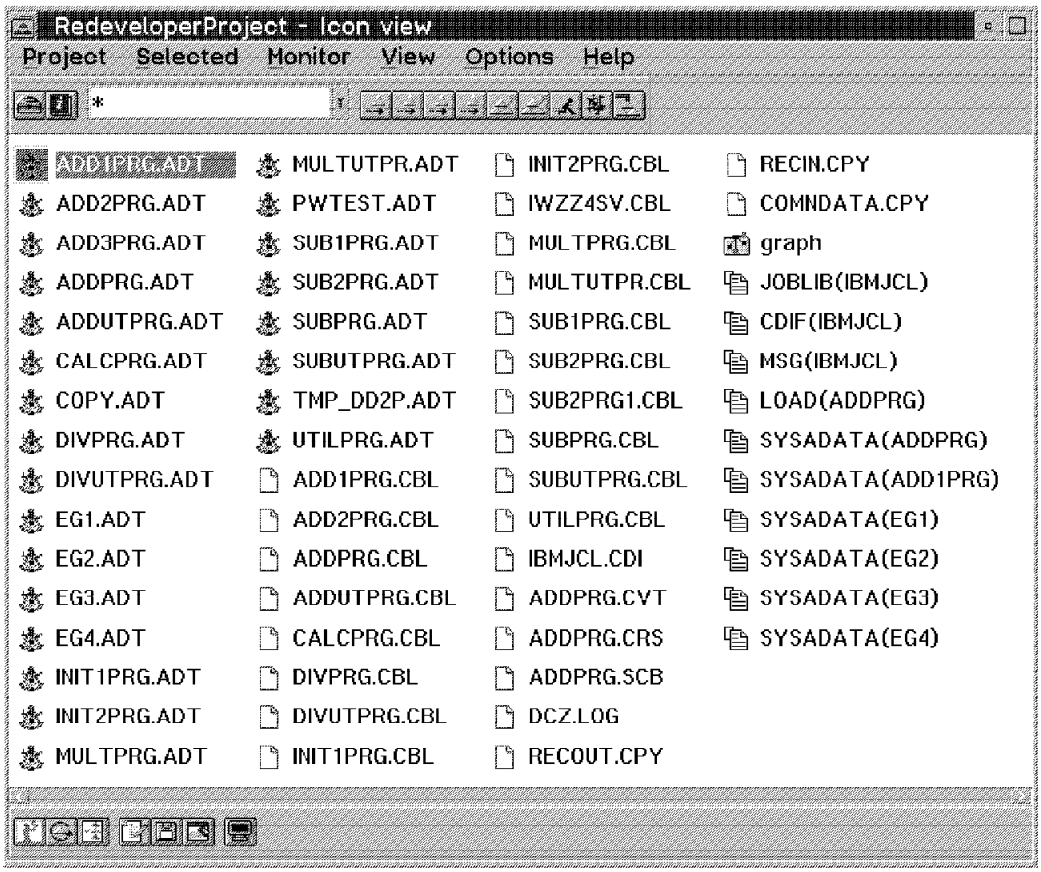


Figure 5. Sample Redeveloper Project window

Note: Before Application Understanding for the first time or if you have deleted and reinstalled Application Understanding, you must run the Application Understanding setup utility as described in “Defining the Database and Binding Application Understanding” on page 18. For Application Understanding and Program Understanding actions on MVS host-resident files, you must also configure APPC on your workstation. If you do not, you may encounter error messages when attempting to use Application Understanding or Program Understanding. See “Related Messages” on page 19 for more information on database, BIND, and APPC configuration messages.

Using Actions in a Redeveloper Project

Table 2. Examples of File Types and Redeveloper Actions You Can Perform On Them

FILE TYPE FILTER EXAMPLE	ACTION
JOBLIB(*) - A JCL file residing on the MVS host	Scan JCL
CDIF(*) - a file on the MVS host that contains the output information from a JCL scan operation.	Retrieve scan JCL output
***.CDI** - A file on the OS/2 workstation that contains JCL scanned output received from an MVS host	Load database
LOAD(*) - A file on the MVS host that was loaded into the AU database	Find in Inventory
***.CBL** - COBOL programs residing on the OS/2 workstation.	Convert, Analyze, Structure (cascaded actions from Conversion and Structuring)
***.CVT** - COBOL programs residing on the OS/2 workstation that have been previously converted	Convert, Analyze, Structure (cascaded actions from Conversion and Structuring)
***.SCB** - COBOL programs residing on the OS/2 workstation that have been previously structured	Convert, Analyze, Structure (cascaded action from Conversion and Structuring)
***.ADT** - COBOL programs residing on the OS/2 workstation generated from compiling a COBOL program with the SYSADATA option	Create and show flow graph, Analyze flow
SYSADATA(*) or **ADATA(*)** or **ADT(*)** - COBOL files residing on the MVS host generated from compiling a COBOL program with the SYSADATA option	Create and show flow graph, Analyze flow

Using Application Understanding Actions

Use the following steps to execute Application Understanding actions with the Redeveloper project. Perform the steps below on an opened Redeveloper project window.

To run different actions, you select the appropriate object's icon and then select the action from the pop-up menu or from the choices in the **Selected** menu.

1. As depicted in Figure 6 on page 29, you can select **Scan JCL** action from the pop-up menu for the **JOBLIB(IBMJCL)** object icon. Note that in Figure 6 on page 29, only MVS host-based files display in the Redeveloper project because the file filter-type **MVS** was selected.

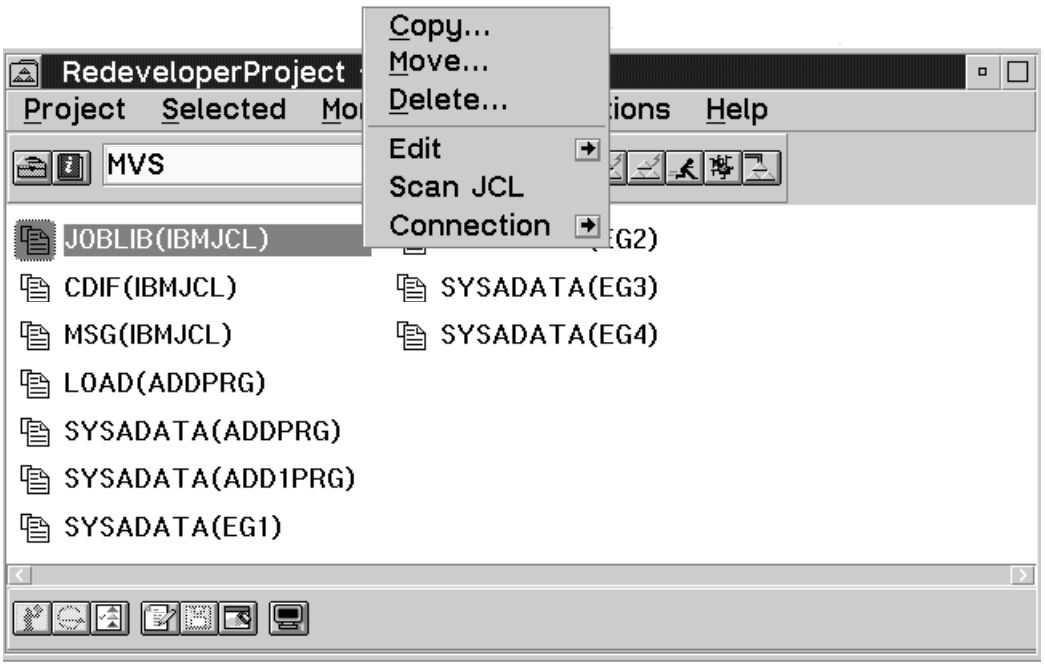


Figure 6. Scan JCL action example

The **Application Understanding Scan Request** window, as depicted in Figure 17 on page 46, opens with all entry fields filled in. By clicking on **Scan JCL** button, JCL on MVS will be scanned. If the process is successful, the **CDIF(IBMJCL)** icon and the **MSG(IBMJCL)** icon appears in the project.

Note: The example shown in Figure 6 uses a data set qualifier type called **JOBLIB**. Your MVS host job environment may use other data set qualifier names. To add a file filter-type to match a data set qualifier name used by your MVS host environment, see the note on "Adding MVS Data Sets to the New Project" on page 24.

2. As depicted in Figure 7 on page 30, you can select the **Retrieve scan JCL output** action from the pop-up menu for the **CDIF(IBMJCL)** object icon.

Using Actions in a Redeveloper Project

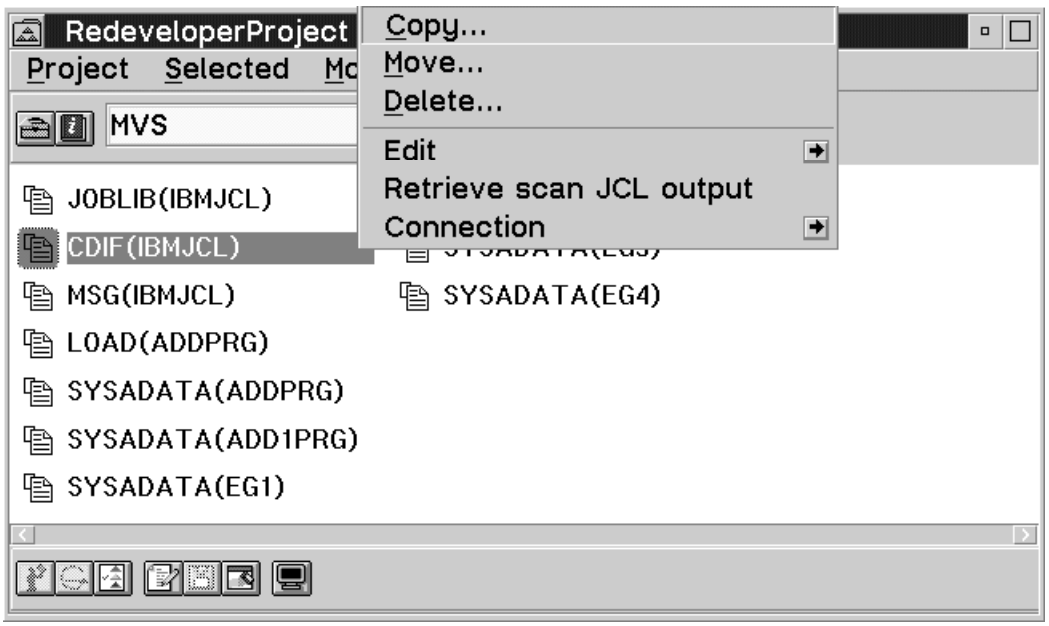


Figure 7. Retrieve JCL action example

The **Application Understanding Retrieve Request** window, as depicted in Figure 18 on page 47, opens with all fields filled in.

The **Path** should be the fully-qualified name of WorkFrame working directory which is the name in the **Working directory** in the **OS/2 Files** location notebook.

If the process is successful, the retrieved SCAN output is shown in the project as **IBMJCL.CDI** icon.

3. As depicted in Figure 8 on page 31, you can select **Load database** action from the pop-up menu for the **IBMJCL.CDI** object icon. Note that in Figure 8 on page 31, only CDI file types display in the Redeveloper project because the file filter-type **CDIType** was selected.

   Be sure DB2/2 database manager has started. If DB2/2 is not running, you start DB2/2 by selecting the **Data Tools** choice from the **Selected** menu from the opened Redeveloper project window.

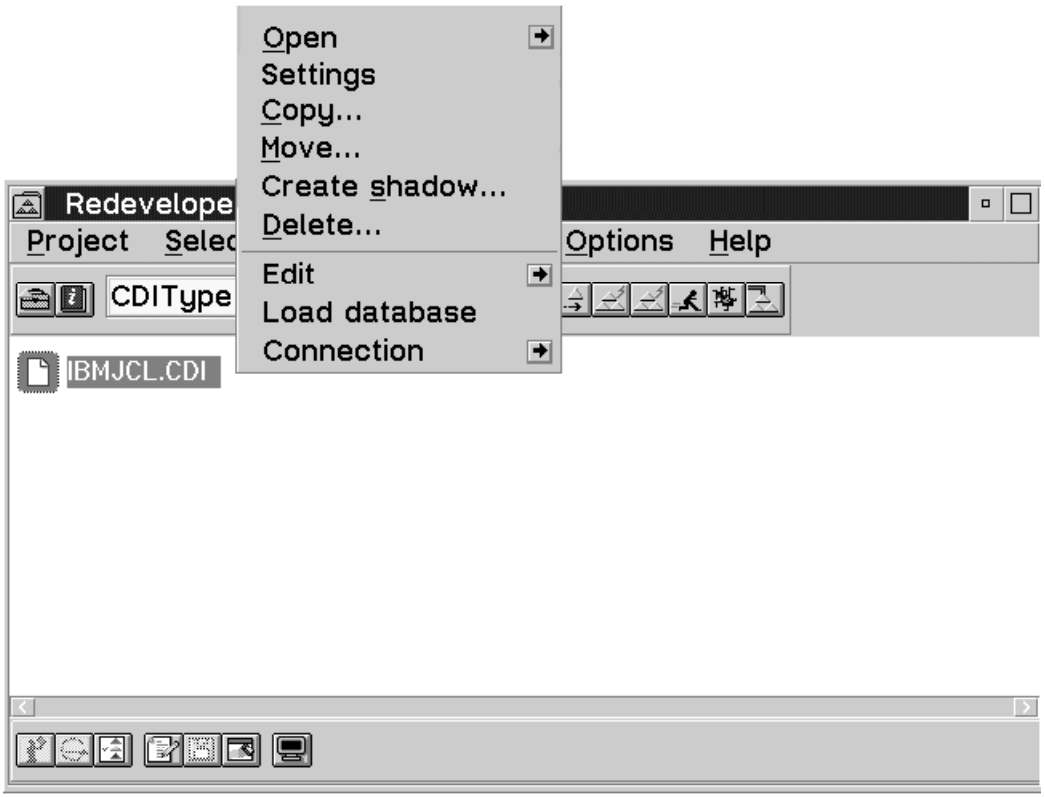


Figure 8. Load Database action example

If the process is successful, a window displays which shows the process completed.

4. As depicted in Figure 9 on page 32, you can select **Find in inventory** action pop-up menu for the **LOAD(ADDPRG)** or **EXEC(ADDPRG)** object icon. This action brings up an Application Understanding Executable Inventory window showing object icons for all PDS members named **ADDPRG** in either **LOAD** PDSs or **EXEC** PDSs stored in the Application Understanding database. For example, **CDXMTS2.DCZ.LOAD(ADDPRG)**, **CDXMTS3.DCZ.LOAD (ADDPRG)**.

Using Actions in a Redeveloper Project

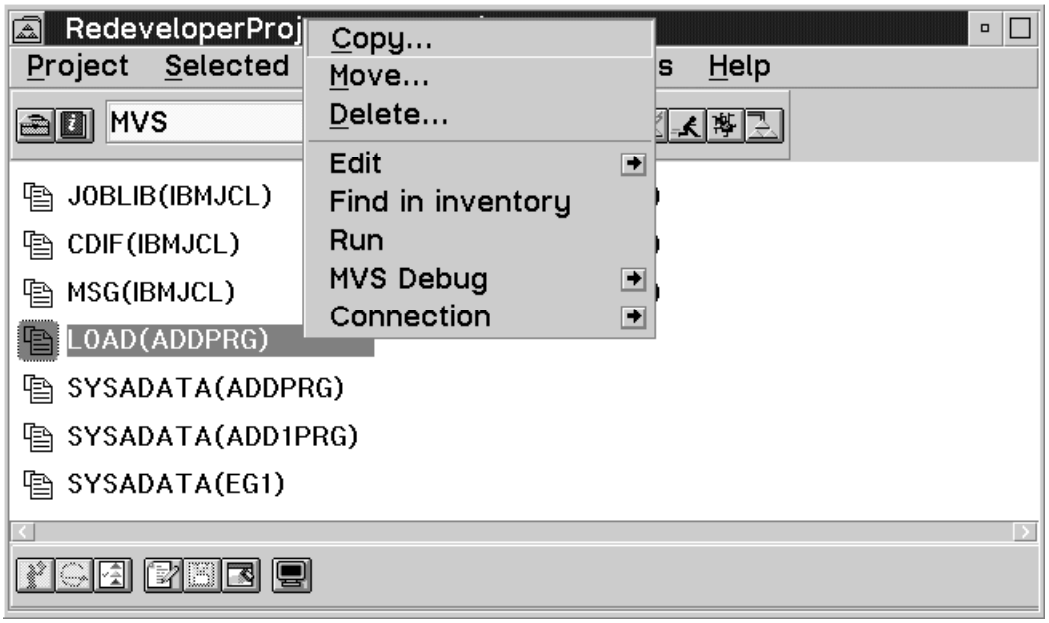


Figure 9. Find in Inventory action example

From the Application Understanding **Executable inventory** window, you can access other Application Understanding functions.

Using Program Understanding Actions

Use the following steps to execute Program Understanding actions with the Redeveloper project. Perform the steps below on an opened Redeveloper project window.

To run different actions, you select the object's icon and then select the action from the pop-up menu or from the choices in the **Selected** menu.

Note: The COBOL programs are listed with a CBL file name extension; or CVT or SCB file name extension if the COBOL programs have been converted/structured by the Program Conversion and Structuring tool. Program Understanding only works with COBOL compiler output files using the SYSADATA option. In order to use Program Understanding to understand COBOL programs, you need to compile the COBOL programs and create files with the SYSADATA file name extension. You can do this by selecting an appropriate object icon name that has a CBL, CVT or SCB file name extension, then select **Compile** from the pop-up menu. Program Understanding works with the resulting COBOL SYSADATA output files from the compiler. For example, you can use the **MVS Compiler** on COBOL files originating on an MVS host to produce SYSADATA(*) or ADATA(*) or ADT(*) files. Ensure that your MVS host can compile COBOL programs that produces ADATA files. If not, you can download your MVS host COBOL programs into an OS/2 directory first, then compile them on your workstation with the ADATA option. For workstation-resident COBOL files, you can use the **OS/2 Compiler** to produce files that have an ADT file name extension.

1. As depicted in Figure 10 on page 33, select a COBOL program (SYSADATA file) that you want to understand, then select the **Create and show flow graph** choice from the pop-up menu.

   This will produce an executable unit (EU) graph of the COBOL program. More than one selection can be made for workstation-resident files. You can only select one file at a time for host-resident files. Figure 10 illustrates a selection made on a SYSADATA file residing on a host. Note that only host SYSADATA file types display in the Redeveloper project because the file filter-type **SYSADATA** was selected. Figure 11 on page 34 illustrates a selection made on a SYSADATA file residing on the workstation. Note that only ADT file types display in the Redeveloper project because the file filter-type **ADT** was selected.

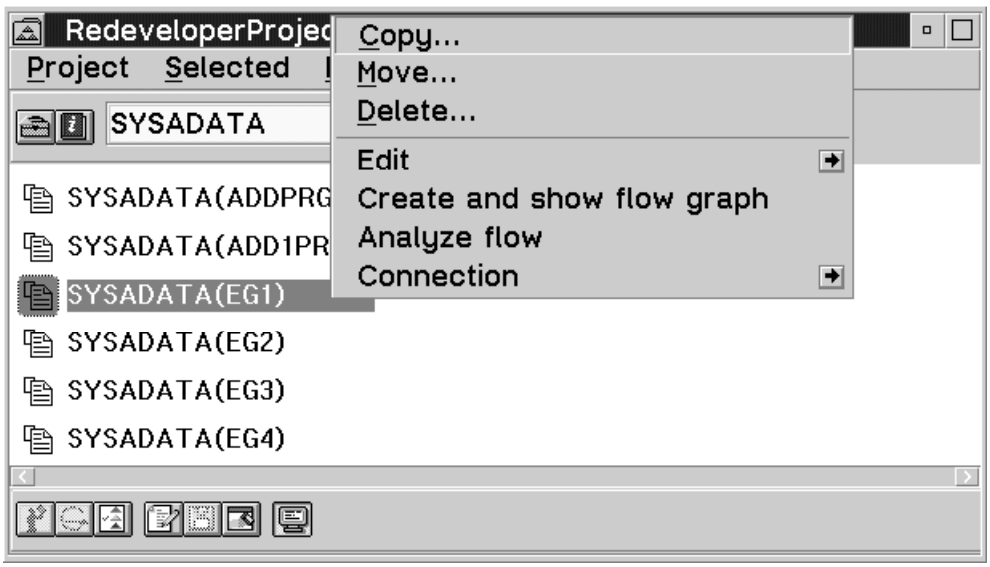


Figure 10. Create and show flow graph action on host-resident file

Using Actions in a Redeveloper Project

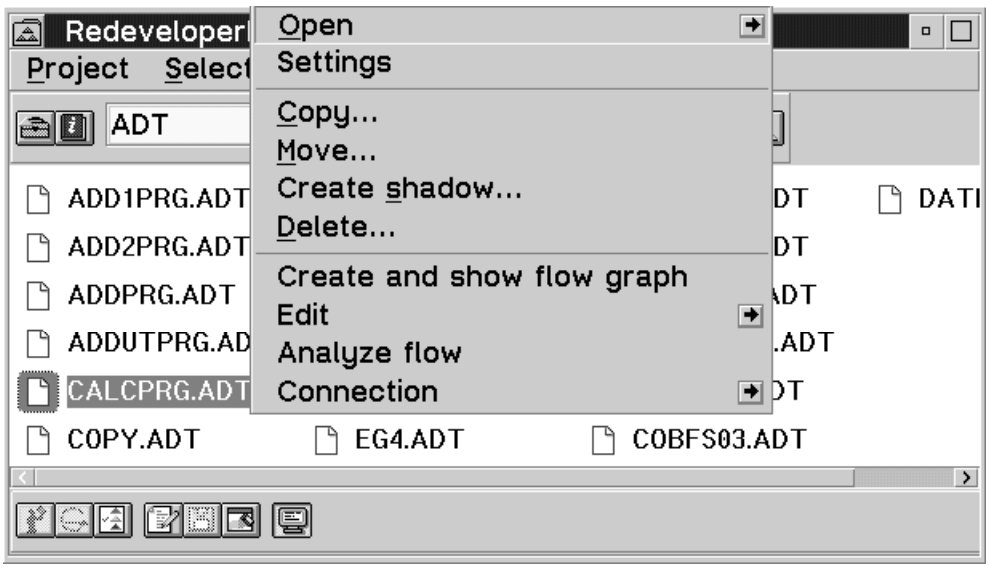


Figure 11. Create and show flow graph action on workstation-resident file

The **Create and show flow graph** function analyzes the selected COBOL program(s), creates a subproject that contains the output of the analyzed program(s), and displays a graph of the executable units of the COBOL programs on your screen.

The default name of the subproject is called **graph**. If the project exists, a **Project - Status** message window opens. On the project status message window, select **Yes** to replace its contents. Select **Overlay** to merge files into it. You can leave the subproject contents unchanged and exit to the previous window by selecting **Cancel**. The displayed graph is the **Executable Unit Graph**. To only analyze the COBOL SYSADATA file and create a subproject containing objects (files) that reflect program structure for each program, select **Analyze flow** choice. You can use this choice to add or replace parts of an executable unit made up of several program units.

Working in the Graph Subproject

The **graph subproject** provides a view of files in a working directory that reflect program structure for each program from the **Analyze flow** or **Create and show flow graph** action on SYSADATA files. Program Understanding enables further browsing on expanded source code files (files with **XSC** extension) listed in the graph subproject. Actions that can be performed on these file types are:

- Show flow graph (EU file name extension only)

  Displays the graph of the executable unit graphical view of the previously analyzed COBOL programs

- Show control flow graph (XSC file name extension only)

Displays a graphical view of the COBOL executable unit with focus on a specific program unit.

- Show expanded source (XSC file name extension only)

  Displays an expanded source view of the specified COBOL program. This allows you to start browsing the source code.

- Create flow graph (XSC file name extension only)

  Creates new files by linking the previously analyzed output file(s) and creates a new file with the EU file name extension in the graph subproject. The **Show flow graph** action can be executed on this file to display a new graph.

Use the following steps to start viewing the expanded source or executable units viewed in the subproject:

1. Open the **subproject**

   The **subproject** (Graph - Icon view) window opens.

2. As depicted in Figure 12, to show the expanded source code you select an object icon that has an **XSC** file name extension, then select the arrow on the **Show** choice from the pop-up menu or in the **Selected** menu.

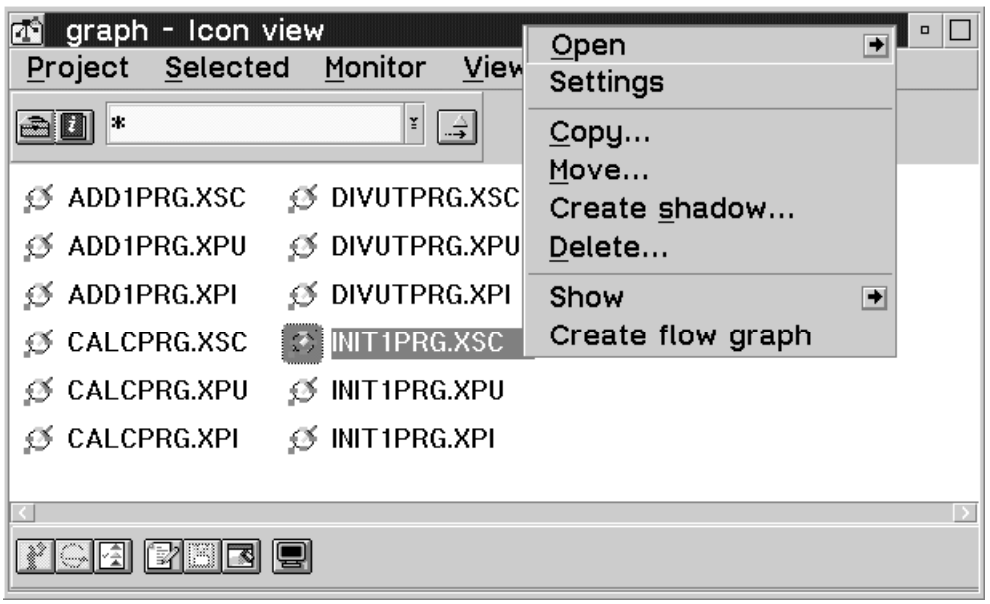


Figure 12. XSC file-type action example

The **Show** function opens up a cascaded window that offers either the **Control flow graph** or **Expanded source choices**. Make your desired selection.

3. As depicted in Figure 13 on page 36, to show the graph of the executable unit you select a file with an **EU** file name extension, then select the **Show flow graph** action from the pop-up menu or in the **Selected** menu.

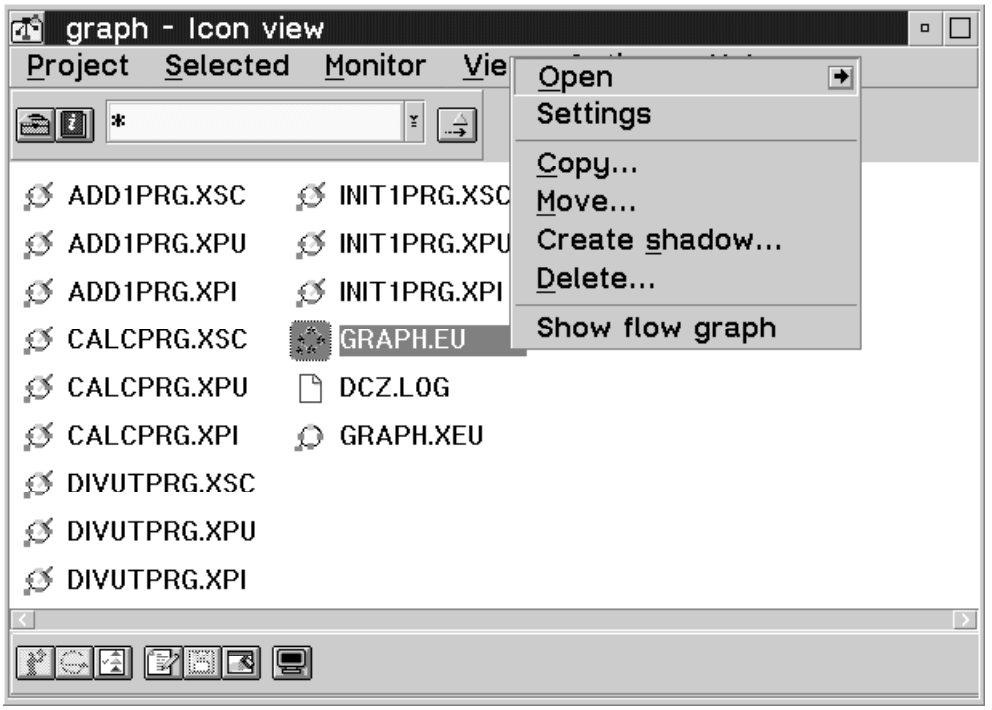


Figure 13. EU file-type action example

Using Program Conversion and Structuring Actions

Use the following steps to integrate Program Conversion and Structuring (PCS) actions with the Redeveloper project. Perform the steps below on an opened Redeveloper project window.

To run different PCS actions, you select the appropriate object's icon and then select the action from the pop-up menu or from the choices in the **Selected** menu.

Converting Your Program

1. As depicted in Figure 14 on page 37, you can select an input COBOL program (for example, a file with a CBL extension) that you want to convert, then select the **Conversion and Structuring** choice from the pop-up menu. Note that in Figure 14 on page 37, all file tyes are displayed in the Redeveloper project because the asterisk (*) was selected in the file filter-type entry field.

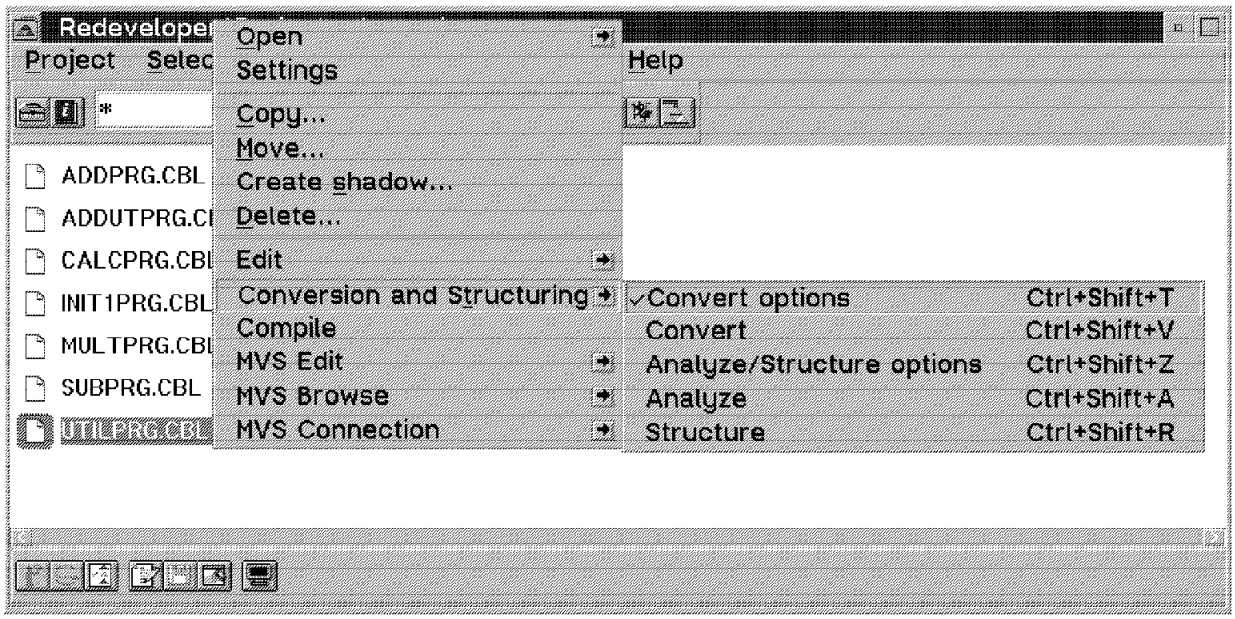


Figure 14. Conversion and Structuring actions

2. From the cascaded menu, choose **Convert options** from the **Conversion and Structuring** menu choice.

   The Convert options notebook appears.

3. After setting the options, select the **Convert** push button in the Convert options notebook to convert your program.

   For more information about Convert option settings, see the online help.

4. If you have already set your options, you can convert your program by highlighting the input program object and selecting **Convert** from the **Conversion and Structuring** menu choice from the **Selected** menu.

   If you have not previously converted your input COBOL program and have not change the destination directory of the converted output COBOL file in the Convert options notebook, a window opens with all entry fields filled in. By selecting the **OK**, a window appears confirming the source of the input file and destination of the output file. By selecting the **OK** button in this window, the program is converted. If the process is successful, a file with a **CVT** extension is created, the Convert Log is created or updated, and a Convert Results report (file with a **CRS** extension) is generated.

   Note: To display the newly created files or reports, refresh the Redeveloper project by ensuring that an asterisk (*) is used in the file filter-type field and by selecting the **Refresh now** choice from the **View** menu.

Using Actions in a Redeveloper Project

Double-click on any report object to open the report. For example, double-clicking on the covert log icon object ☑ for a converted program, opens the convert log file for that program.

Analyzing Your Program

Program Conversion and Structuring analyzes the input program's control flow and provides information about the complexity, understandability, and maintainability of your input program. The reports generated after analyzing your program will help you determine the benefits of structuring your program.

To analyze your program:

1. As depicted in Figure 14 on page 37, you can select an input COBOL program (for example, a file with a **CBL** or **CVT** or **SCB** extension) that you want to analyze, then select the **Conversion and Structuring** action choice from the pop-up menu or from the **Selected** menu. Note that in Figure 14 on page 37, all file types are displayed in the Redeveloper project because the asterisk (*) was selected in the file filter-type entry field.
2. From the cascaded menu, choose the **Analyze/Structure options** from the **Conversion and Structuring** choice.

   The Analyze/Structure options notebook appears.
3. After setting the options, select the **Analyze** push button in the Analyze/Structure options notebook to analyze your program.

   For more information about the Analyze option settings, see the online help.
4. If you have already set your options, you can analyze your program by highlighting the input program object and selecting **Analyze** from the **Conversion and Structuring** choice from the **Selected** menu.

The reports generated after the analysis are created or updated in the project. The reports include:

- Input COBOL (the input COBOL file name with a **ICR** extension)
- Reengineering Report (the input COBOL file name with an **REP** extension)
- Complexity Metrics Log

Note: To display the reports, refresh the Redeveloper project by ensuring that an asterisk (*) is used in the file filter-type field and by selecting the **Refresh now** choice from the **View** menu.

Double-click on any report object to open the report.

Preparing Your Program to Improve Structuring

In about 25% of typical COBOL programs, some manual preparation of the input program before structuring will help you obtain the maximum benefit from the structuring function. After analyzing the input program, you can use the information from the analysis step to improve the input program and decrease the number of complex PERFORMs.

Using Actions in a Redeveloper Project

Program Conversion and Structuring has an Expert Advice feature that tells you how to do the program preparation. This advice is provided for programs with complex performed procedures (Group I) or procedures that have multiple entries and/or exits (Group II). You can use the Expert Advice feature after you have analyzed or structured a program by opening the Reengineering report and selecting **Expert Advice** from the menu bar, then selecting **Open as Expert-Advice...**.

Structuring Your Program

In this step, Program Conversion and Structuring produces a structured program and several reports. You can use these reports to understand the structured output program and how it relates to the input program. The structured output program is equivalent in function to the input program.

To structure your program:

1. As depicted in Figure 14 on page 37, you can select an input COBOL program (for example, a file with a CBL or CVT or SCB extension) that you want to structure, then select the **Conversion and Structuring** action choice from the pop-up menu or from the **Selected** menu. Note that in Figure 14 on page 37, all file types are displayed in the Redeveloper project because the filter type (*) was selected in the file filter-type entry field.
2. From the cascaded menu, choose the **Analyze/Structure options** from the **Conversion and Structuring** choice.

   The Analyze/Structure options notebook appears.
3. After setting the options, select the **Structure** push button in the Analyze/Structure options notebook to structure your program.

   For more information about the Structure option settings, see the online help.
4. If you have already set your options, you can structure your program by highlighting the input program object and selecting **Structure** from the **Conversion and Structuring** choice from the **Selected** menu.

   If you have not previously structured your input COBOL program and did not change the destination directory of the structured output COBOL file in the Structure options notebook, a window opens with all entry fields filled in. By selecting the **OK** button, the program is structured. If the process is successful, a file with an **SCB** extension is created, and report files are created or updated in the project. The reports include:

   - Input COBOL (the input COBOL file name with an ICR extension)
   - Output COBOL (the input COBOL file name with an OCR extension)
   - Reengineering Report (the input COBOL file name with an REP extension)
   - Complexity Metrics Log

   Note: To display the reports, refresh the Redeveloper project by ensuring that an asterisk (*) is used in the file filter-type field and by selecting the **Refresh now** choice from the **View** menu.

Double-click on any report object to open the report.

Using Actions in a Redeveloper Project

Modularizing Your Program

After reviewing the Reengineering Report, you can use Program Conversion and Structuring's modularization advice to find out which procedures represent the best candidates for modularization. This will help you reduce the size of the program and its overall complexity.

To include modularization advice in your Reengineering Report, set the Analyze/Structure options as follows:

1. Highlight the input program object, select the **Selected** menu bar choice, then select the arrow button next to the **Conversion and Structuring** choice.

   A cascaded menu appears.

2. From the cascaded menu, select **Analyze/Structure options** from the **Conversion and Structuring** menu choice.

   The Analyze/Structure options notebook appears.

3. Select the **Reporting** tab.
4. Make sure the box to the left of **Perform modularization analysis** is checked.
5. Click in the **Minimum** setting entry field under **Modularization statement** to set the minimum number of lines a PERFORM should contain to be considered for modularization in the report.
6. Click in the **Maximum** entry field to set the maximum number of lines a PERFORM should contain to be considered for modularization in the report.
7. After setting the options, select the **Structure** push button from the Analyze/Structure options notebook to structure your program and generate the Reengineering report.

You can then look at the Reenginering Report to see the modularization advice:

1. Double-click on the report object in the project.

   The report opens.

2. Click on the **Mod Advice** tab to open the modularization advice page.
3. To see modularization statistics, select the **Statistics table** push button.

   For more information about this report, click on the **Help** push button.

You can use the modularization advice to manually modularize your program. Modularization advice is not provided for programs containing CICS HANDLE commands.

Testing Your Program

The final step is to verify that your structured program is equivalent in function to the input program.

Final Note

Some of these steps are optional depending on your reengineering scenario:

Using Actions in a Redeveloper Project

- Some input programs will not need to be converted to a higher level of COBOL.
- About 75% of typical COBOL programs will not need input preparation before structuring.

Chapter 4. Understanding Redeveloper Tools

This section describes the Redeveloper tools and their functions in greater detail. In addition, there may be situations where you wish to access the functions in one of the Redeveloper's tools independently of the Redeveloper integration environment. This chapter describes the use the tools in that manner.

Accessing Redeveloper Tools

Each tool has its own icon in the **Works** folder. You can access all Redeveloper tools from this folder. To open the **Works** folder:

1. Double-click with the mouse on the **VisualAge for COBOL** icon located on the desktop.
2. From the opened VisualAge for COBOL - Icon View window, double-click on the **Works** icon.

   The **Works - Icon View** window opens. The icon objects pertaining to Redeveloper are illustrated in Figure 15 on page 43.

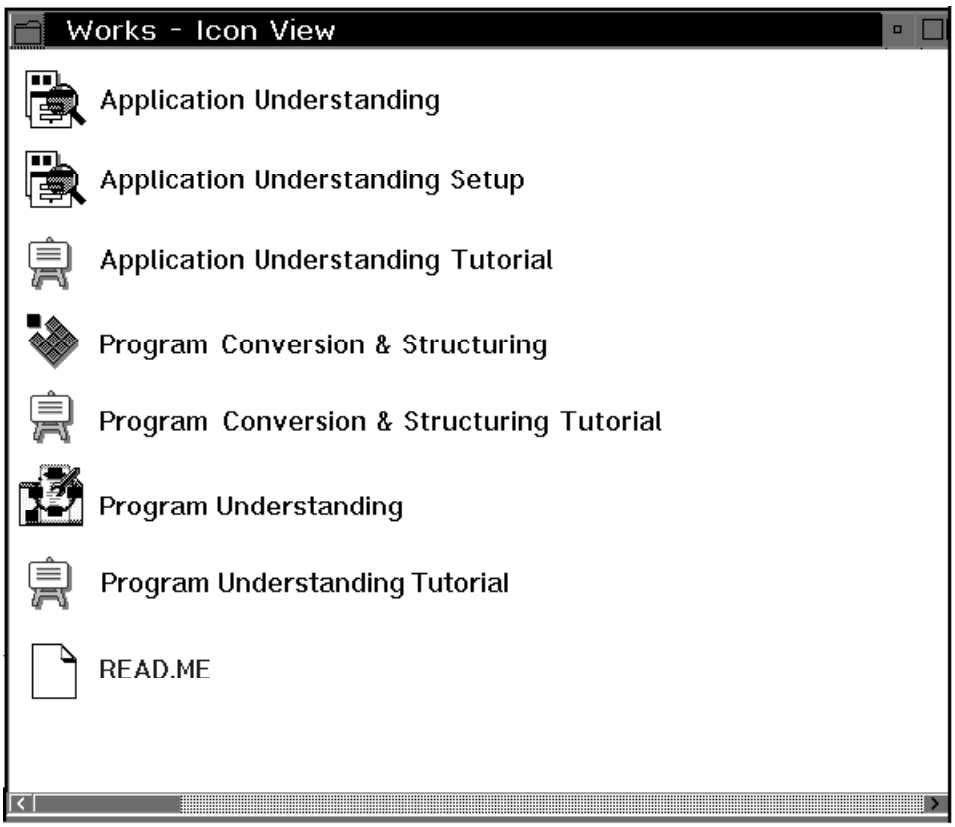


Figure 15. Works folder showing Redeveloper icon objects

The Redeveloper tools are described below.

To analyze and understand your applications:
Use the Application Understanding (AU) tool to capture your existing MVS batch jobs inventory and to help you better understand how applications are constructed, where they are used, and what files they process.

To understand your programs:
Use the Program Understanding (PU) tool to help you understand executable units, associated program units, data structures, and copy libs.

To convert and structure your programs:
Use the Program Conversion and Structuring (PCS) tool to convert older COBOL programs to higher levels of COBOL and to structure unstructured COBOL.

You can access the online tutorials. The online tutorials provide you with step-by-step lessons on how to use the Redeveloper tools. The following tutorials are available:

Starting Application Understanding from the Works Folder

- Application Understanding Online Tutorial
- Program Understanding Online Tutorial
- Program Conversion and Structuring Online Tutorial

Starting Application Understanding from the Works Folder

You can begin to start Application Understanding (AU) by selecting the AU icon object in the Works Folder. As depicted in Figure 16, the AU window displays categories of inventory stored in the database. These categories allow you to select the kind of information you want to work with. You can select a JCL component category and open an Inventory window to view the components or you can set a qualifier to extract the components you want to view.

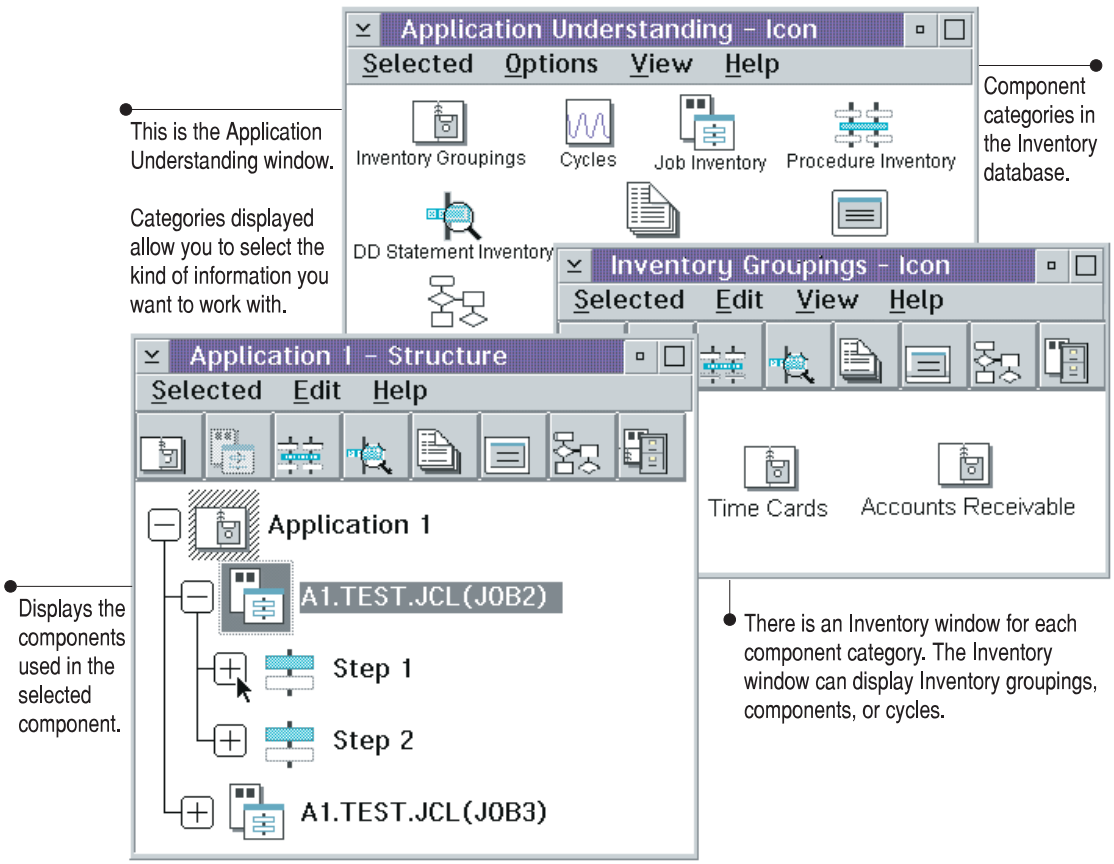


Figure 16. Application Understanding window

The Structure window shows how a component is structured in a top-down tree view

Use the Application Understanding window to set up a JCL library scan, to manually add information to the database, or to access application information already in the database.

Populating Your Inventory Database

To use the facility of Application Understanding, you must have data in your Inventory database. You can either add the data manually or use the capture facility.

You populate your database by scanning JCL libraries on the host, retrieving the JCL scan output to your workstation, and then loading the JCL scan output into the Inventory database.

Scanning the JCL Libraries

You set up a JCL scan to capture information about your application inventory. Information extracted during the scan is written to a file that can be retrieved and then used to populate your Inventory database.

Note: You do not have to be logged on to MVS to initiate a scan from the workstation, but you must have a host-workstation connection. If you do not, you can run the scan on an MVS batch job by modifying the sample JCL included in the installation package. The scan output and source file can be downloaded.

To scan a JCL library:

1. Go to the **Application Understanding** window.
2. Select **Options**, then **Scan JCL** from the menu bar.

   The Scan Request window opens as depicted in Figure 17 on page 46.
3. In the **Application Understanding Scan Request** window, type the required information into the entry fields.

Populating Your Inventory Database

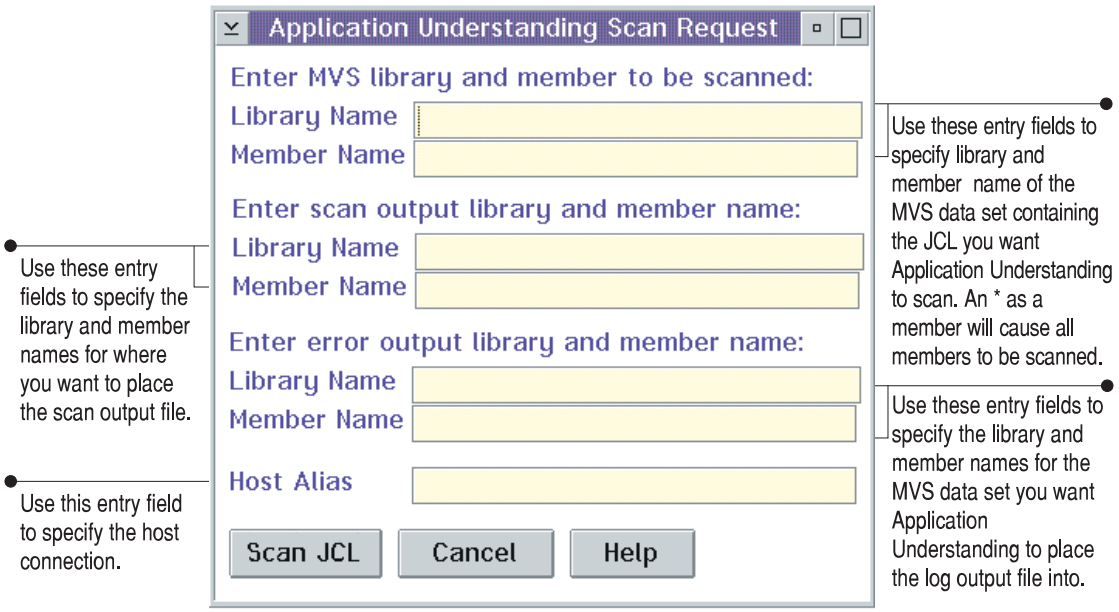


Figure 17. Scan Request window

4. Press the **Scan JCL** push button.

   Note: The JCL scan request runs on MVS and you will be notified when the scan completes.

   Note that the TIME and REGION resource allocations must be large enough to accommodate the number of members to be scanned in a single execution of this job (depending on the number of procs per job and the size of the jobs, you may need to adjust these values). There is an upper limit of 3000 members (JCL scanned and proc members called) that can be processed in a single run of the scan.

Retrieving the JCL Scan Output

After the JCL library has been scanned, you can download the MVS data sets containing scan output to your OS/2 workstation. Doing this allows you to later load the contents into your Inventory database.

Note: You do not have to be logged on to MVS to initiate a retrieve request, but you must have a host-workstation connection.

To retrieve the JCL scan output:

1. Go to the **Application Understanding** window.
2. Select **Options**, then **Retrieve** from the menu bar.

   The Retrieve Request window opens as depicted in Figure 18 on page 47.

3. In the **Application Understanding Retrieve Request** window, enter the required information into the entry fields.

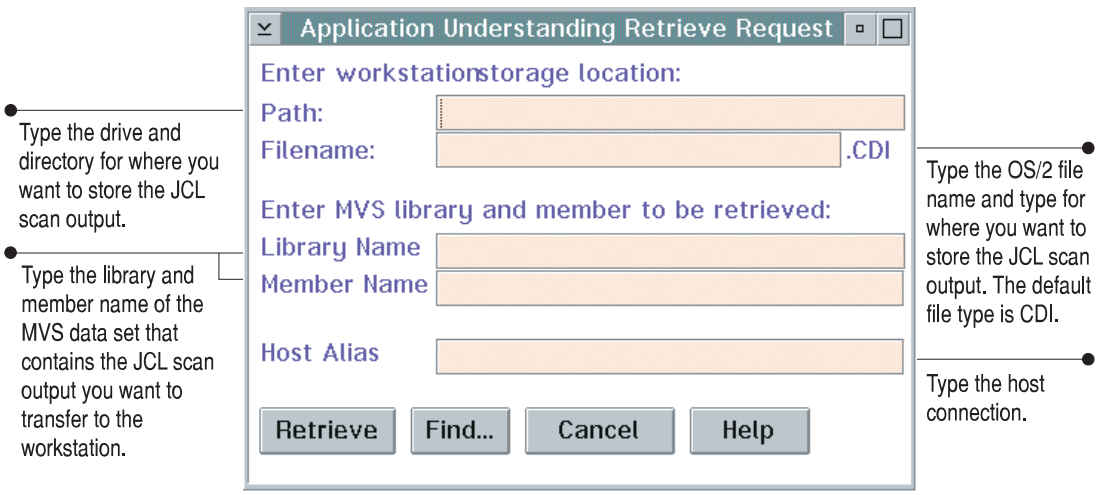


Figure 18. Retrieve Request window

4. Select the **Retrieve** push button to begin retrieving the data set.
5. Select the **Find** push button to select the file to be retrieved from the list of files created. The fields are automatically filled in.

Loading the Database with the JCL Scan Output

You populate your Inventory database by loading the JCL scan output. This allows you to use Application Understanding's query and search functions to understand your application inventory.

Note: Before you can load the JCL scan output, you must have scanned a JCL library and retrieved the output to your workstation.

To load the database with the JCL scan output:

1. Go to the **Application Understanding** window.
2. Select **Options**, then **Load** from the menu bar.

   The JCL Load Request window opens as depicted in Figure 19 on page 48.
3. In the **Application Understanding JCL Load Request** window, type the required information into the entry fields.

   Note: The default file extension of the scan output file is .CDI.

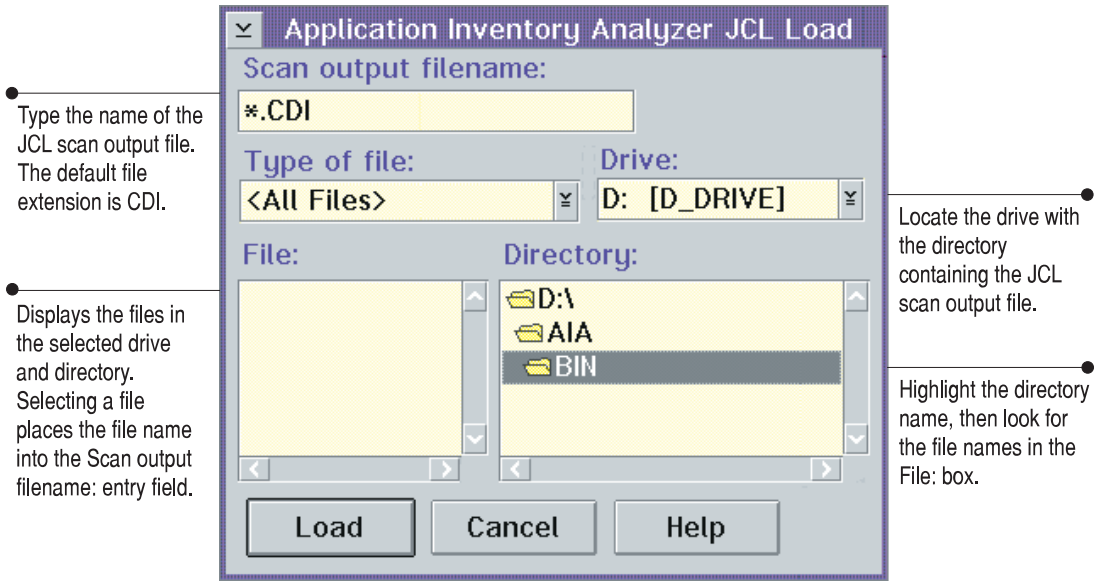


Figure 19. Load Request window

4. Select the **Load** push button to begin loading the JCL scan output file into your Inventory database.

Viewing the Contents of Your Inventory Database

You can view your inventory to see where its various components are used and re-used. You may filter the information you want from the database by setting a qualifier for an Inventory window. You can see how a component is constructed by opening a Structure view.

Opening an Inventory Window

You can select a category from the Application Understanding window and open an Inventory window for that category. The view will contain only information under that category.

To open an Inventory window:

1. Select a category in the **Application Understanding** window.
2. Select **Selected**, then **Open** from the menu bar.

   An Inventory window for that category opens displaying the components of that category.

   Note: You can specify what components you want to appear in the Inventory window by setting a qualifier for the category.

Viewing the Contents of Your Inventory Database

Setting a Qualifier

You set a qualifier to selectively filter what information is displayed in an Inventory window. This allows you to view only the components you are interested in. You can set separate qualifiers for each inventory category. Once you set a qualifier, you can view it by selecting the **Details** view in the **Application Understanding** window.

To set a qualifier:

1. Go to the **Application Understanding** window.
2. Select the category you want to qualify.
3. Select **Selected**, then **Qualifier** from the menu bar.
4. Type an SQL Qualify statement into the Qualify dialog box

   This tells DB2/2 what to display in the window.

   Basic syntax:

   ```
   attribute operator attribute value
   ```

   For more information about the SQL Qualifier statement, see the online help.

   Note: The Qualify statement is not case sensitive, but the value may be.
5. Select the **OK** pushbutton.

   The qualify statement is saved and used in all sessions until you change it.

Opening a Structure View of Components

You open a Structure window to better understand how a component is structured. For example, to see what jobs are in an inventory group or what job steps are in a job. You may want to select a particular sub-component of this component and see if it is used anywhere else.

To open a structure view of a component:

1. In the **Application Understanding** window, select an Inventory category.
2. Select **Select**, then **Open** from the menu bar.
3. In the Inventory window you just opened, select the component whose structure you want to view.
4. Select **Selected**, then **Structure** from the menu bar.

   A structure window opens showing the related components of the selected component.
5. Select the + (plus) to expand a level or select on the - (minus) to contract a level.

   Note: Double-click on a component in the structure to see additional related components.

Starting Program Understanding from the Works Folder

Modifying Your Inventory Database

As you use your Inventory database, you may want to make adjustments to the database by manually adding or deleting components.

Creating a New Component

You may create a new component in any of the inventory categories.

To create a new component:

1. From the **Application Understanding** window, open an Inventory window for the category you want to create a new component for.
2. In the Inventory window, select **Edit**, then **Create** from the menu bar.

   A definition notebook opens containing attributes that describe the component.
3. Update or correct the attribute values as needed, then select the **OK** push button.

   Note: Attributes marked with an asterisk (*) are called Key attributes and cannot be modified. If any one of these attributes contains an incorrect value, the component must be deleted and re-added to correct the value.

Deleting a Component

You clean up your database by manually deleting unwanted components from the Inventory database. When you delete a component, you are deleting all relationships associated with the deleted component.

Note: The component is also deleted from the DB2/2 database.

To delete a component:

1. From the **Application Understanding** window, open an Inventory window for the category that contains the component you want to delete.
2. Select the component you want to delete.
3. Select **Edit**, then **delete** from the menu bar.
4. Select the **Yes** push button to delete the component.

For more advanced tasks, see the Application Understanding Online Help.

Starting Program Understanding from the Works Folder

You can begin Program Understanding (PU) by selecting the Program Understanding icon object in the **Works** Folder. As depicted in Figure 20 on page 51, the initial Program Understanding - Icon View window contains icon objects for the **Analyze flow**, **Create and show flow graph**, **Create flow graph**, **Show control flow graph**, **Show expanded source**, and **Show flow graph** functions.

Starting Program Understanding from the Works Folder

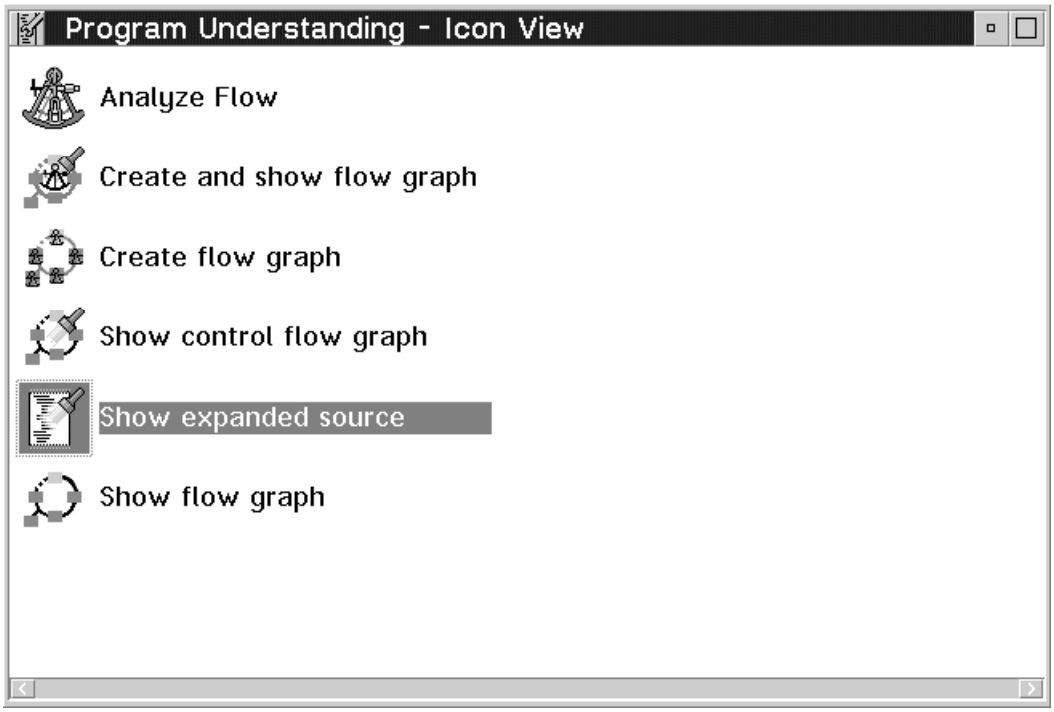


Figure 20. Program Understanding Folder -Icon View

These objects relate to actions you can perform with the Program Understanding tool. You use these actions on COBOL programs (ADATA files) and resulting analyzed files to understand COBOL programs graphically and textually.

Program Understanding works with ADATA files produced by MVS host COBOL compilers and OS/2 COBOL compilers.

Starting Program Understanding from the Works Folder

As depicted in Figure 21, to use Program Understanding actions effectively:

1. Identify COBOL files (files with a CBL extension) that are linked to a module or executable.
2. Create a directory on your OS/2 workstation for files related to that COBOL executable and populate it with ADATA files. You can do this by compiling COBOL programs with the ADATA option either on your OS/2 workstation or on the MVS host. If you do compilation on the MVS host, download the binary ADATA files into this directory. You must use a binary download process.

   Note: Ensure that your MVS host can compile COBOL programs that produces ADATA files. If not, you can download your MVS host COBOL programs into an OS/2 directory first, then compile them on your workstation with the ADATA option. You can do this by selecting an appropriate object icon name that has a CBL file name extension (or SCB or CVT file name extension if a COBOL program has been structured or converted), then select **Compile** from the pop-up menu.
3. Analyze the various program flows by double-clicking on the ADATA files (files with an **ADT** extension).
4. Create the flow graph by double-clicking on any of the **Create flow graph** object icons (files with an **XPI** extension).
5. Graphically view the entire executable by double-clicking on the **Show flow graph** object icon (files with an **EU** extension), or by double-clicking on an individual program **Show control flow graph** object icon (files with **XPU** extension).

   If a COBOL source file has changed, then you can repeat step 3 for the changed ADT file followed by steps 4 and 5 to incorporate and view the changes.

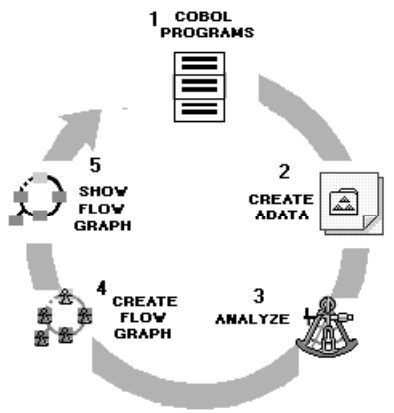


Figure 21. Basic steps for displaying graphical views using Program Understanding tool

Starting Program Understanding from the Works Folder

Analyze Flow

The **Analyze flow** function analyzes the selected COBOL program(s) and creates files that contains the output of the analyzed program(s).

To can analyze the program select one of the following three ways:

1. Double-click on an ADATA object icon (file with an **ADT** extension).
2. Drag an ADT file object icon to the **Analyze flow** action object icon in the opened Program Understanding tools window.

The analyzed output files have an **XSC** and **XPI**, and **XPU** extension. By default, these files are created in the same OS/2 directory that your ADATA files are in. You can change this directory by adding a parameter to the Settings notebook **Working directory** parameter field of the **Analyze flow** icon object to direct the files to another directory. The parameter is /**d***x*, where *x* is the directory to contain the files.

Create Flow Graph

The **Create flow graph** function links the analyzed files together so that cross program impacts may be observed.

To create the flow graph select one of the following:

1. Double-click on an object icon with an **XPU** extension.
2. Select the **Create flow graph** menu choice on a object icon that has a **XPU**, **XPI**, or **XSC** file name extension.
3. Drag an object icon with a file name extension of **XPU**, **XSC**, or **XPI** to the **Create flow graph** object icon in the opened Program Understanding tools window.

The created output files have an **XEU** and **EU** extension. By default, these files are created in the same OS/2 directory that your ADATA files are in. You can change this directory by adding a parameter to the Settings notebook **Working directory** parameter field of the **Create flow graph** icon object to direct the files to another directory. The parameter is /**d***x*, where *x* is the directory to contain the files.

Note: If the files with the **ADT**, **XPI**, **XPU**, or **XSC** file name extension change, the flow graph has to be re-created. The output file with the **EU** file name extension controls the graph content. You can edit this file before creating the flow graph. If it is erased, it is rebuilt using all selected and available files with an **ADT** file name extension.

Show Flow Graph

The **Show flow graph** function displays the analyzed programs linked together is a graph.

Starting Program Understanding from the Works Folder

To show the flow graph, select one of the following:

1. Double-click on an object icon with an **XEU** or **EU** extension.
2. Select the **Show flow graph** menu choice on a object icon that has a **XEU** or **EU** file name extension.
3. Drag an object icon with a **XEU**, or **EU** file name extension from the directory to the **Show flow graph** icon in the opened Program Understanding tools window.

As depicted in Figure 22, the Control Flow graph opens. The graph shows the control flow structure of the previously analyzed COBOL program executable units.

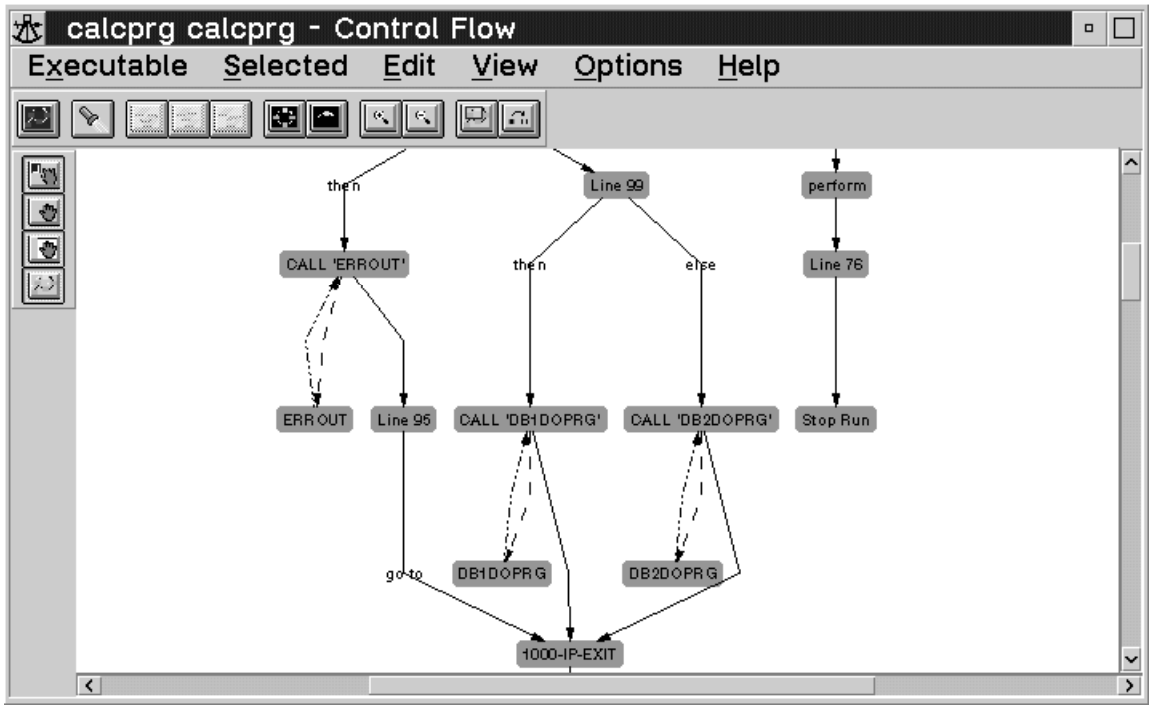


Figure 22. Control flow graph view

Show Expanded Source

The **Show expanded source** function displays the expanded source of a program in the context of an executable. This expanded source listing view is of the previously analyzed COBOL program(s).

To Show the expanded source listing, select one of the following:

1. Double-click on an object icon with an **XPI** extension.

2. Select the **Show expanded source** menu choice on a object icon that has a **XPU**, **XSC**, or **XPI** file name extension.
3. Drag the object icon file name with an **XPU**, **XSC**, or **XPI** extension to the **Show expanded source** object icon in the opened Program Understanding tools window.

As depicted in Figure 23, the Expanded source window opens with a view showing the expanded source listing of the previously analyzed COBOL program.

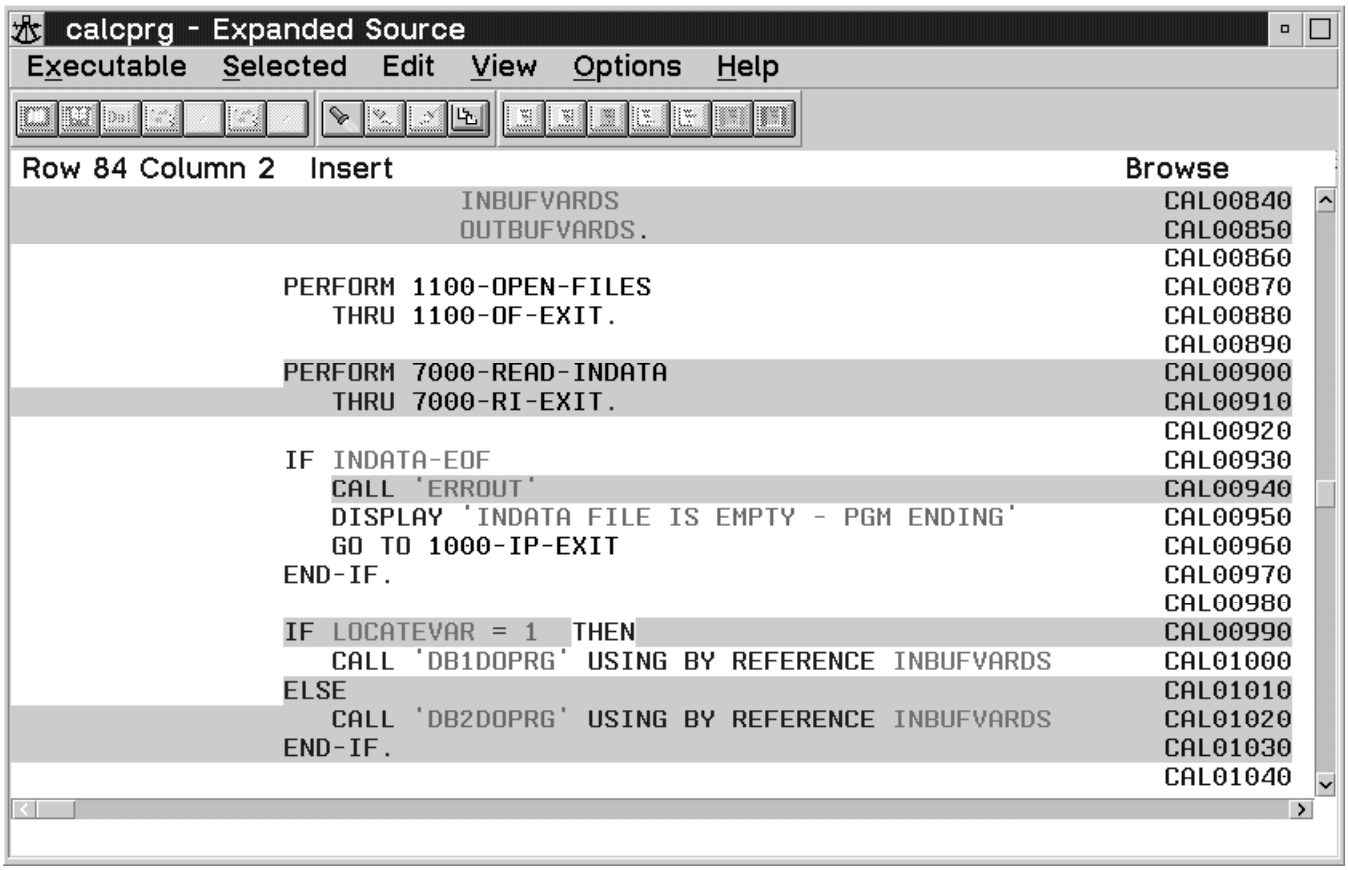


Figure 23. Expanded source view

Show Control Flow Graph

The **Show control flow graph** function displays the graphical view of the COBOL executable unit with focus on a specific program unit. This graphical view is of the previously analyzed COBOL program(s). The Show control flow graph function works on files that have an **XSC** extension.

Starting Program Conversion and Structuring from the Works Folder

To show the control flow graph select one of the following:

1. Double-click on an object icon with an **XSC** extension.
2. Select the **Show control flow graph** menu choice on a object icon that has a **XSC**, **XPU**, or **XPI** file name extension.
3. Drag the object icon with a **XSC** file name extension to the **Show control flow** object icon in the opened Program Understanding tools window.

As depicted in Figure 22 on page 54, the Control Flow window opens with the graph showing the program unit control flow structure of a program executable unit.

Create and Show Flow Graph

The **Create and show flow graph** function analyzes the selected COBOL program, creates files that contains the output of the analyzed program, and displays a graph of the executable units of the COBOL program on your screen.

To create and show the flow graph, select one of the following:

1. Select the **Create and show flow graph** menu choice on a object icon that has a **ADT** file name extension.
2. Drag an object icon with a **ADT** file name extension to the **Create and show flow graph** object icon in the opened Program Understanding tools window.

As depicted in Figure 22 on page 54, the Control Flow window opens. The graph shows the control flow structure of the COBOL program executable units.

Starting Program Conversion and Structuring from the Works Folder

You can start Program Conversion and Structuring (PCS) by selecting the PCS icon in the Works Folder. As depicted in Figure 24 on page 57, the initial PCS window contains icon objects for the Convert Log, Complexity Metrics Log, and a User Profile. Initially, the logs are empty. However, as you convert programs, the Convert Log contains a history of all workstation programs that have been converted. As you structure or analyze programs, the Complexity Metrics Log contains a history of all programs that have been analyzed or structured. The User Profile allows you to specify an editor, settings for Analyze/Structure and Convert functions, and font settings for programs and the Cross Reference Browser reports.

Starting Program Conversion and Structuring from the Works Folder

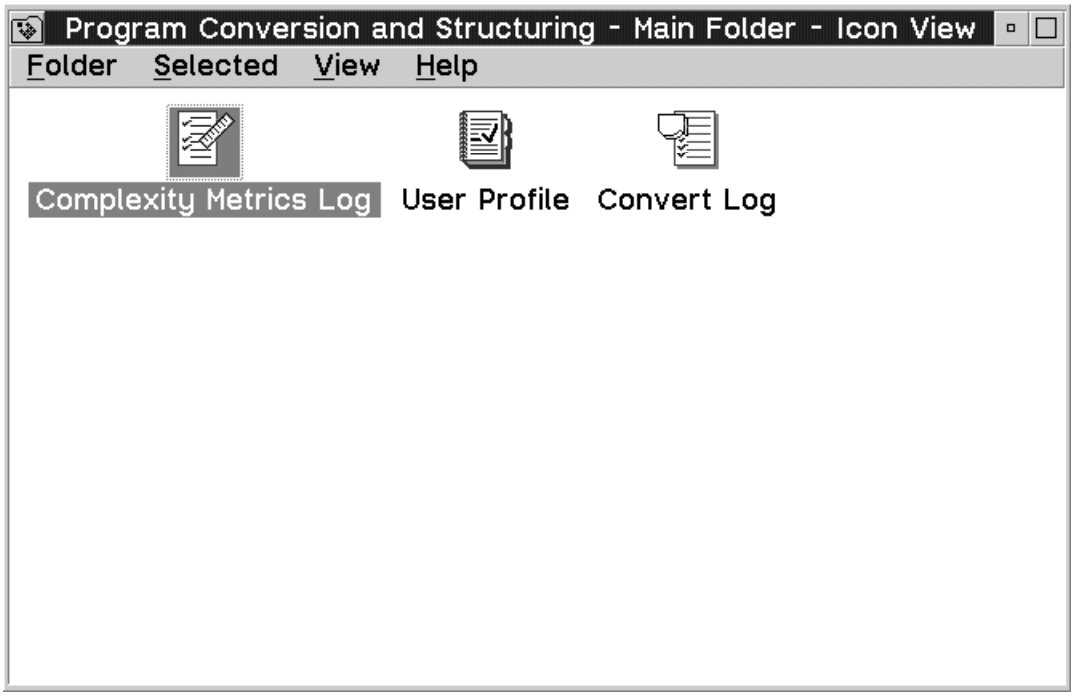


Figure 24. Program Conversion and Structuring main folder window

As depicted in Figure 25 on page 58, reengineering COBOL programs with PCS involves six basic steps. Note that some of these steps are optional, depending on your reengineering scenario:

- Converting
- Analyzing
- Preparing
- Structuring
- Modularizing
- Testing

Converting Your Program

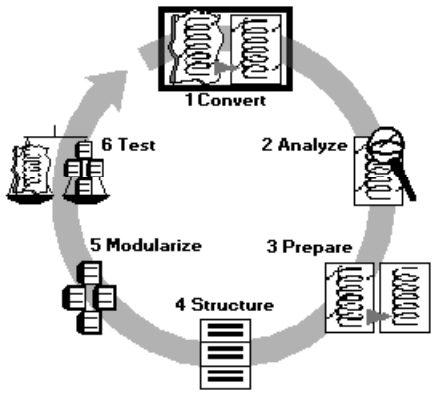


Figure 25. Basic PCS steps for reengineering COBOL programs

Converting Your Program

To initially convert a COBOL program from a lower level to a higher level, you need to add a COBOL file to the PCS folder. You can add it as an input file. You can do this by selecting the **Folder** menu bar choice, then selecting **Add file...**. Type the required information into the entry fields and select **OK**. The COBOL program file appears as an icon in the PCS folder window.

To convert a COBOL program:

1. Select a program in the opened PCS folder.
2. Select **Selected** from the menu bar, then select **Convert options**.

   As depicted in Figure 26 on page 59, the Convert options notebook opens.
3. After setting the options, select **Convert** in the Convert option notebook to convert your program.

For more information about the Convert function, see the online help.

The reports generated after the conversion are displayed as objects in the PCS folder. You can view the information in the reports by double-clicking on the report objects.

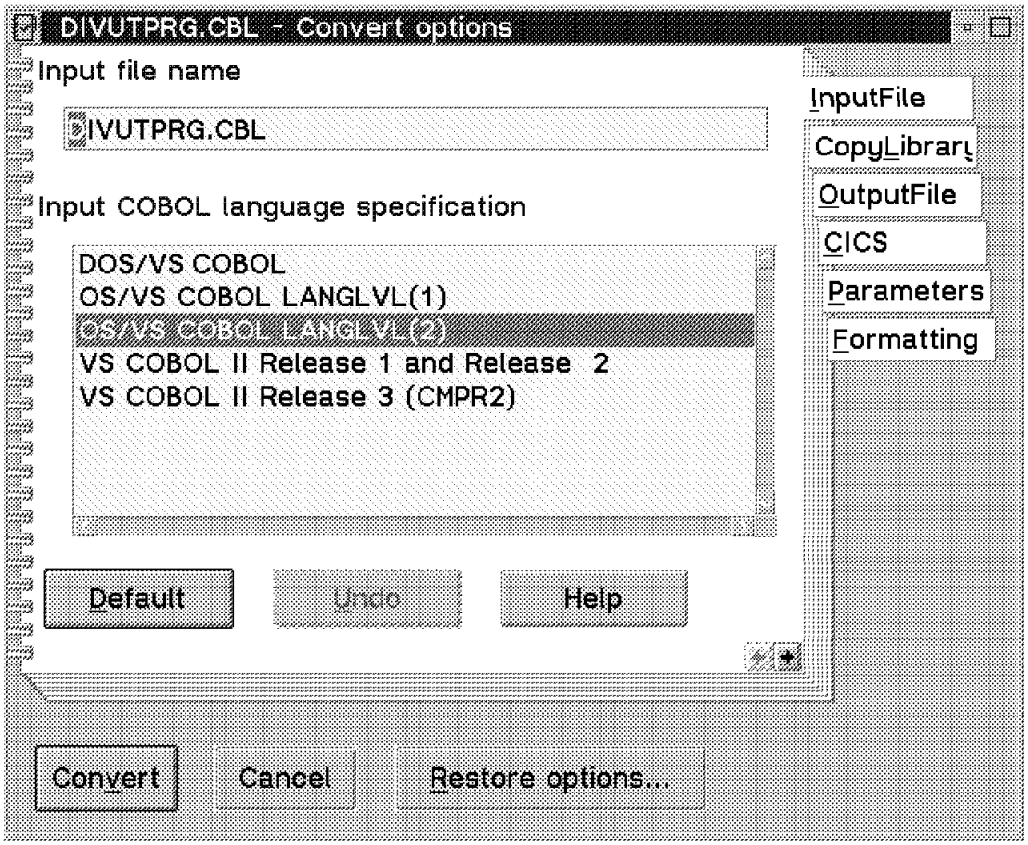


Figure 26. Convert options notebook

Analyzing Your Program

To initially analyze a COBOL program, you need to add a COBOL file to the PCS folder. You can add it as an input file. You can do this by selecting the **Folder** menu bar choice, then selecting **Add file...**. Type the required information into the entry fields and select **OK**. The COBOL program file appears as an icon in the PCS folder window.

To analyze a COBOL program:

1. Select a COBOL program in the opened PCS folder.
2. Select **Selected** from the menu bar, then select **Analyze options**.

   As depicted in Figure 27 on page 60, the Analyze/Structure options notebook opens.
3. After setting the options, select **Analyze** push button in the Analyze/Structure options notebook to analyze your program.

For more information about the Analyze function, see the online help.

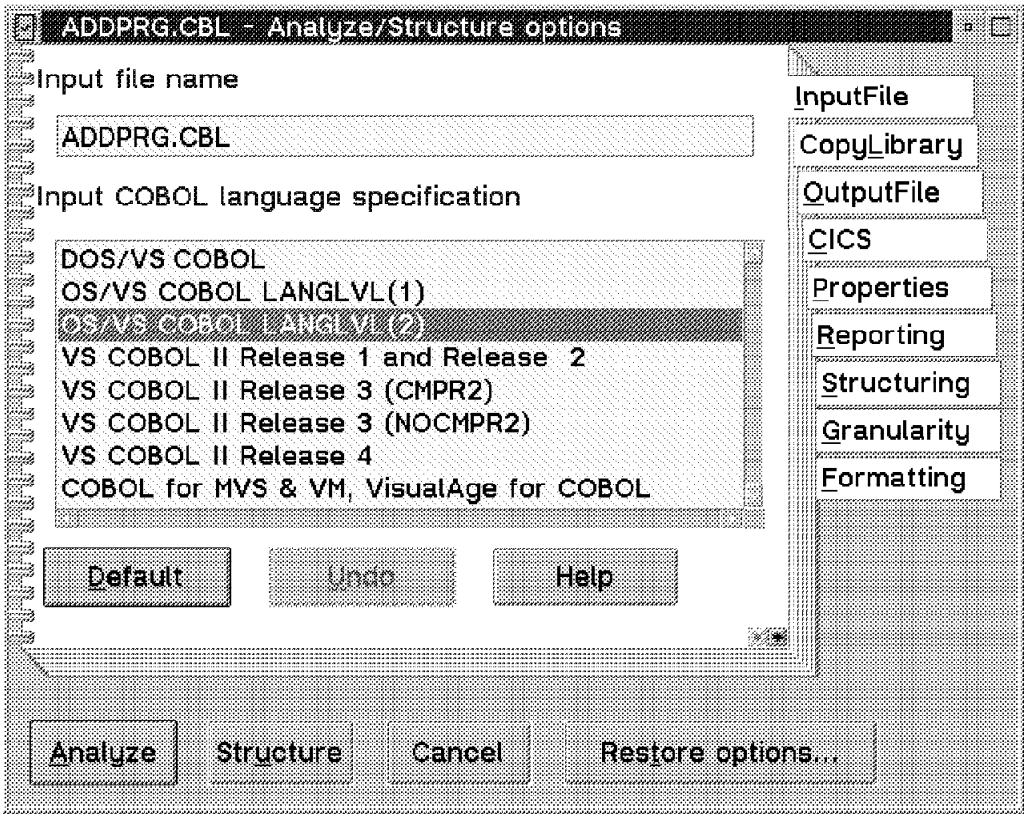


Figure 27. Analyze/Structure options notebook

The reports generated after the analysis are displayed as icon objects in the PCS folder. You can view the information in the reports by double-clicking on the report icon objects.

Preparing Your Program to Improve Structuring

In about 25% of typical COBOL programs, some manual preparation of the input program before structuring will help you obtain the maximum benefit from the structuring function. After analyzing the input program, you can use the information from the analysis step to improve the input program and decrease the number of complex PERFORMs.

Program Conversion and Structuring has an Expert Advice feature that tells you how to do the program preparation. This advice is provided for programs with complex performed procedures (Group I) or procedures that have multiple entries and/or exits (Group II). You can use the Expert Advice feature after you have analyzed or struc-

tured a program by opening the Reengineering report and selecting **Expert Advice** from the menu bar, then selecting **Open as Expert-Advice...**.

Structuring Your Program

To initially analyze a COBOL program, you need to add a COBOL file to the PCS folder. You can add it as an input file. You can do this by selecting the **Folder** menu bar choice, then selecting **Add file...**. Type the required information into the entry fields and select **OK**. The icon file name appears in the PCS folder window.

To structure a COBOL program:

1. Select a program in the opened PCS folder.
2. Select **Selected** from the menu bar, then select **Structure options**.

   As depicted in Figure 27 on page 60, the Analyze/Structure options notebook opens.
3. After setting the options, select **Structure** push button in the Analyze/Structure options notebook to structure your program.

For more information about the Structure function, see the online help.

The reports generated after structuring are displayed as objects in the PCS main folder. You can view the information in the reports by double-clicking on the report objects.

Modularizing Your Program

After reviewing the Reengineering Report, you can use Program Conversion and Structuring's modularization advice to find out which procedures represent the best candidates for modularization. This will help you reduce the size of the program and its overall complexity.

To include modularization advice in your Reengineering Report, set the Analyze/Structure options as follows:

1. Highlight the input program object, select the **Selected** menu bar choice, then select either the **Analyze options** or **Structure options** choice.

   The Analyze/Structure Options notebook appears.
2. Select the **Reporting** tab.
3. Make sure the box to the left of **Perform modularization analysis** is checked.
4. Click in the **Minimum** setting entry field under **Modularization statement** to set the minimum number of lines a PERFORM should contain to be considered for modularization in the report.
5. Click in the **Maximum** entry field to set the maximum number of lines a PERFORM should contain to be considered for modularization in the report.
6. After setting the options, select the **Structure** push button in the Analyze/Structure options notebook to structure your program and generate the Reengineering report.

Testing Your Program

You can then look at the Reenginering Report to see the modularization advice:

1. Double-click on the report object in the project.

   The report opens.
2. Click on the **Mod Advice** tab to open the modularization advice page.
3. To see modularization statistics, select the **Statistics table** push button.

   For more information about this report, click on the **Help** push button.

You can use the modularization advice to manually modularize your program. Modularization advice is not provided for programs containing CICS HANDLE commands.

Testing Your Program

The final step is to verify that your structured program is equivalent in function to the input program.

Appendix A. Updating the CONFIG.SYS File

This appendix contains a list of CONFIG.SYS file updates made by the Redeveloper installation process, where **c:\IBMCOBOL** is the drive and directory containing Redeveloper.

```
LIBPATH=c:\IBMCOBOL\DLL;
SET PATH=c:\IBMCOBOL\HELP;c:\IBMCOBOL\BIN;
SET DPATH=c:\IBMCOBOL\TMP;c:\IBMCOBOL\SAMPLES;c:\SAMPLES\CSTUT;
c:\IBMCOBOL\MVS;c:\IBMCOBOL\MESSAGES;c:\IBMCOBOL\ICON;c:\IBMCOBOL\HELP;
SET HELP=c:\IBMCOBOL\HELP;
SET BOOKSHELF=c:\IBMCOBOL\HELP;
SET CMWPATH=c:\IBMCOBOL;
```

The following CONFIG.SYS file updates apply to Program Conversion and Structuring:

```
SET ECFFILE=c:\IBMCOBOL\BIN;
SET ECFWORK=c:\IBMCOBOL\TMP;
SET ECFCCCA='1',
```

© Copyright IBM Corp. 1996

Glossary

A.

analyze. In Program Conversion and Structuring, a function that helps you determine which programs require structuring. Program Conversion and Structuring analyzes an input program's structural properties. It produces a Reengineering Report and appends to the Complexity Metrics Log.

In Program Understanding, a function that produces files which helps you understand program logic, control flow, and data flow for selected programs.

application. A set of software components used to perform specific types of work on a computer, such as inventory control or payroll.

attribute. (1) Used to describe Application Understanding components. (2) In DB2/2, a characteristic of data contained in a column or row, such as length, data type, or data format.

C.

CICS HANDLE commands. CICS, or Customer Information Control System, is an IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

code. One or more statements that are to be executed by the computer. Instruction(s).

Comments and other language processor statements that appear in or before a block of code. Statements are also called code. All statements are either code or data.

Complex Performed Procedures. A procedure has complex usage if the combination of its invocations (through PERFORM, GO TO, or fall-through), together with its internal control flow require Program Conversion and Structuring to integrate the procedure into the places from which it is invoked in order to structure it. A procedure that has complex usage is also called a Group I procedure.

Complexity Metrics Log. In Program Conversion and Structuring, a file that contains a summary of the complexity metrics of all programs analyzed or structured. This report provides a useful management tool to help evaluate the maintainability of entire libraries (or systems) of COBOL programs. During each Analysis or Structure run, the complexity metrics are appended to this summary.

component. Parts of Redeveloper Application Understanding, Program Understanding, and Program Conversion and Structuring are component parts of Redeveloper

One of ten types of information about MVS JCL that you can use Application Understanding to add, modify, or query. Components are classes of individual entities that can be found in the database. Components are shown in the Application Understanding main window.

convert. A function in Program Conversion and Structuring to convert the source code from DOS VS COBOL, OS/VS COBOL LANGLVL(1), OS/VS COBOL LANGLVL(2), VS COBOL II Release 1 and Release 2, or VS COBOL II Release 3 (CMPR2) to VS COBOL II Release 3 or COBOL for MVS & VM, VisualAge for COBOL.

Convert Log. An object in the Main Folder window of Program Conversion and Structuring (PCS) that contains a history of all workstation programs that have been converted by PCS.

copylib. An abbreviated term for COPY Library.

Cross Reference Browser. A window for cross-referencing between the Input COBOL Report and the Output COBOL Report after Program Conversion and Structuring finishes structuring your input program.

cycle. The frequency in which a job runs.

D.

data. A statement classified as not intended to be executed. See code.

data set. In MVS, a named collection of related data records that is stored and retrieved by an assigned name.

data structure. A compound data element.

Definition notebook. A type of window in Application Understanding. The Definition notebook contains attri-

© Copyright IBM Corp. 1996

bute information about components and relationships between components. You can use the window to add, modify, or delete attribute information and create or delete relationships between components.

I.

Input COBOL Report. In Program Conversion and Structuring, the Input COBOL Report lists the input COBOL program. Line numbers are attached to each statement. You can use the line numbers as you investigate the control flow properties of the input program documented in the Reengineering Report.

inventory window. A type of window in Application Understanding. The inventory window contains the set of all components in the database represented by the selected component icon.

J.

job. A collection of related job steps. A job is identified in JCL by a JOB statement.

job control language (JCL). In MVS, a sequence of commands used to identify a job to the operating system and to describe the job's requirements.

job step. A job step of a load module or procedure consists of job control statements that request and control execution of a load module and request the resources needed to run the load module. A job step is identified in the JCL by an EXEC statement.

L.

library. A repository containing a file or related set of files; for example, in stock control, a set of inventory control files.

M.

modularizing. In Program Conversion and Structuring, the breaking up of a large program into a main program and one or more separately compilable subprograms.

multiple entries. Two different kinds of statements can be reported as multiple entries of the procedure:

- GO TO statements outside the procedure whose targets lie strictly within the procedure.
- ENTRY statements within the procedure.

metrics reports. For Program Conversion and Structuring, the Reengineering Report and the Complexity Metrics Log contain counts of statements and structures in the input program.

multiple exits. Five different kinds of statements can be reported as multiple exits of the procedure:

- GO TO statements within the procedure whose targets lie outside of the procedure
- GOBACK statements within the procedure
- STOP RUN statements within the procedure
- EXIT PROGRAM statements within the procedure
- CALL statements whose call-literal is named in the list of non-returning calls.

N.

nodes. In Program Understanding, a graphical representation of one or more blocks. See block.

O.

object icons. A graphic symbol, displayed on a screen, that represents objects where a particular function or software application can be selected.

object module. Output of a compiler, input to a link-editor.

Output COBOL Report. In Program Conversion and Structuring, a file that lists the output COBOL program from the structuring function.

P.

procedure. In MVS, a named collection of job steps that can be included in a job.

program unit. In MVS, the name of a compilable block of code that gains control when the compiled program is run.

project environment. In a project environment, all the files you might use to create an application (source code, analysis reports, and graphics, for example) are organized in a single project window.

The benefit of this approach is that you can easily launch any available VisualAge for COBOL or Redevel-

oper tool simply by selecting it from the project window menu.

Q.

qualifier. In DB2/2, a short identifier used to filter data from the database. All objects such as tables or queries in DB2/2 are prefixed with a unique qualifier that allows them to be grouped, named, and displayed under a particular category or userid, for example (USERID.QUERYNAME). The object can be viewed only if the correct qualifier is specified.

R

Reengineering Report. In Program Conversion and Structuring, the results of an analysis of the PERFORM logic and control flow, and input program complexity metrics. This report can help you evaluate the complexity and maintainability of the input program. From this, you can determine whether the program requires structuring.

routine. A largest collection of nodes or blocks identified by a callable entry node.

S.

scan output. The results of a request for Application Understanding to analyze MVS JCL libraries for information such as:

- Jobs currently in production
- Jobsteps within jobs
- Procedures called by job steps
- DD statements within job steps
- Executables used in job steps

There are two outputs created by the scan:

- a file in CDIF format suitable for loading into the DB2/2 database
- a log of activities that occurred during the scan

Application Understanding places the scan output in MVS data set whose name you specify in the Scan Request window. You can then download the CDIF format scan output to an OS/2 file and load it into the DB2/2 database.

scanner. The portion of Application Understanding residing on the MVS host that extracts data from JCL libraries in order to provide information about production applications and the components that comprise them.

set window. A type of window in Application Understanding. The Set Window contains the set of all components in the database represented by the selected component icon. There are nine set windows, one for each type of component that exists in Application Understanding:

- Application set window
- Cycles set window
- Jobs set window
- Procedures set window
- DD statements set window
- Executable set window
- Program units set window
- Source modules set window
- Libraries set window

source module. The input to a compiler, written in a source language.

standalone environment. In a standalone environment, you organize your files into separate folders available from the Main Folder.

Statistics Table. In Program Structuring and Conversion, the Statistics Table provides detailed statistics for any procedures listed in Modularization Advice, among others. You bring up this table by selecting the **Statistics table...** push button on the Mod Advive page of the Reengineering Report (Structure) notebook.

structure. In Program Conversion and Structuring, a function that creates a structured program. The structure function analyzes input program structural properties, generates the structured COBOL program, produces an output file, a Reengineering report, and a Cross Reference Browser. It also appends metric statistics to the Complexity Metrics Log.

symbol. An item defined by the language processor that represents something understood by the language processor: a general register, a constant, a routine name, a record field, or a statement to branch to.

Index

A

actions used in the project 27
adding
  components 13
adding files to the project 24
Adding input programs to PCS 58
Advice 38, 60
Analysis reports 6
Analyze/Structure options notebook 60
analyzing your program 6, 38, 59
Application Understanding
  binding to database 18
  defining database 18
  find in inventory project action 32
  load database action 31
  manually installing host files 16
  messages 19
  populating database with sample data 18
  retrieve JCL project action 30
  scan JCL project action 29
Application Understanding inventory database
  viewing contents 48
Application Understanding tool
  starting from folder 44
Applications Understanding
  how to use Redeveloper project actions 28

C

CCCA and COBOL/SF technology 5
choosing what to install 12
complex PERFORMs 38, 60
Complexity Metrics log 38
components
CONFIG.SYS file
  updating automatically 12
conversion reports 6
convert log 6, 37
Convert options notebook 58
Convert Results file 6
Convert Results Report 37
converting from COBOL input languages 5
converting to COBOL output languages 5
converting your program 6, 58
creating the maintenance project 23
Cross Reference Browser 6

D

deleting components 13

G

getting help for installation 10

H

help
  installation 10

I

Input COBOL Report 38
Installation
  message help 13
  problems 13
installation fails 14

L

Loading Application Understanding database
  with JCL Scan output 47

M

modiyfing Application Understanding inventory
  database 50
modularizing your program 40

O

online documentation
  viewing x
Output COBOL Report 39

P

PERFORMs, complex 38, 60
populating Application Understanding database 45
preparing your program 38

© Copyright IBM Corp. 1996

Program Conversion and Structuring
how to use Redeveloper project actions 36
Program Conversion and Structuring tool
starting from folder 56
Program Conversion and Structuring, analyzing your program 38
Program Conversion and Structuring, modularizing your program 40
Program Conversion and Structuring, structuring your program 39
Program Understanding
analyze flow project action 34
create and show flow graph project action 32
create flow graph 34
how to use Redeveloper project actions 32
show compiled source 34
show control flow graph action 34
show flow graph 34
Program Understanding tool
analyze flow function 53
create and show flow graph function 56
create flow graph function 53
show control flow graph function 55
show expanded source function 54
show flow graph function 53
starting from folder 51
project application types 21
purpose of tools 2

R

README file 11
Redeveloper
accessing tools individually 42
components 11
getting started using Redeveloper Integration 21
other components required 9
overview 1
purpose 2
who uses it 1
Redeveloper integrated project
actions 27
adding files 24
adding files originating from a OS/2 workstation 24
adding files originating from an MVS host 24
adding OS/2 files 26
creation of 23
installation requirements 21
role of 21
setup verification 22
Redeveloper tools folder, opening 42
Reengineering Report 38, 60
reinstalling components 13
requirements 9
retrieving JCL Scan Output 46
role of project 21

S

scanning JCL libraries 45
selecting components to install 12
setting
target directory 10
stopping installation 14
Structuring reports 7
structuring your program 7, 39, 61

T

target drive and directory
setting 10
testing your program 40, 62

U

understanding Application Understanding tool 4
understanding Program Structuring and Conversion tool 5
understanding Program Understanding tool 4
understanding Redeveloper tools 43
updating CONFIG.SYS 12
using Application Understanding actions 28
using Program Conversion and Structuring actions 36
using Program Understanding actions 32
using Program Understanding subproject files 34

V

viewing online documentation x

W

working with Redeveloper project application types 21

We'd Like to Hear from You

VisualAge for COBOL for OS/2
Introducing Redeveloper
Installing and Getting Started
Version 1 Release 2

Publication No. GC26-8749-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - IBMMail: USIB2VVG at IBMMAIL
  - IBMLink: COBPUBS at STLVM27
  - Internet: COMMENTS@VNET.IBM.COM

  Be sure to include the following with your comments:
  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

VisualAge for COBOL for OS/2
Introducing Redeveloper
Installing and Getting Started
Version 1 Release 2

Publication No. GC26-8749-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

Name

Address

Company or Organization

Phone No.

Readers' Comments
GC26-8749-00

IBM®

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department W92/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

IBM®

Part Number: 83H9122

Printed in U.S.A.

IBM VisualAge for COBOL for OS/2

GC26-8421	Getting Started
SC26-8419	Programming Guide
SC26-4769	Language Reference
SC26-8789	Building VisualAge for COBOL for Fun and Profit
GC26-8663	Fact Sheet
GC26-8483	Technology Brochure
GC26-8749	Introducing Redeveloper

IBM's VisualAge products and services are
not associated with or sponsored by Visual Edge
Software, Ltd.

83H9122

GC26-8749-00