



Preparing Projects



IBM Asset Transformation Workbench
v1.1



Preparing Projects

Note:

Before using this information and the product it supports, read the information in “Notices.”

First Edition (February 2005)

This edition applies to IBM Asset Transformation Workbench (product number 5724-L54) and to all subsequent releases and modifications until otherwise indicated in new editions.

For the latest information about this product, please refer to the Release Notes.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of IBM. Information in this document is subject to change without notice and is not guaranteed to be error-free.

You can order publications through your IBM representative or the IBM branch office serving your locality. 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.

Licensed Materials - Property of IBM.

Product Reference: IBM Asset Transformation Workbench v1.1

Document Reference: REL7.3.07.DOC02.A

© 2005 Copyright International Business Machines Corporation. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© 2004, 2005 Relativity Technologies, Inc. All rights reserved.

RescueWare is a registered trademark of Relativity Technologies, Inc. All other brands mentioned in this document are trademarks or registered trademarks of their respective holders.

Contents

Preface

<i>Audience</i>	ix
<i>Organization</i>	x
<i>Conventions</i>	xi
<i>Related Manuals</i>	xi
<i>Online Help</i>	xii

1 Overview

<i>Basic Concepts</i>	1-1
<i>Repositories, Entities, and Relationships</i>	1-1
<i>Workspaces and Projects</i>	1-2
<i>Registering Applications</i>	1-3
<i>Verifying Applications</i>	1-4
<i>Inventoring Applications</i>	1-5
<i>Reference Reports and Orphan Analysis</i>	1-5
<i>Decision and Resource Resolution</i>	1-5
<i>Relaxed Parsing</i>	1-6

	<i>Partitioning Applications</i>	1-6
	<i>What's Next?</i>	1-6
2	Setting Up a Workspace and Projects	
	<i>Creating a Workspace</i>	2-1
	<i>Opening a Workspace</i>	2-4
	<i>Registering Source Files in a Workspace</i>	2-6
	<i>Setting Registration Options</i>	2-6
	<i>Registering Source Files</i>	2-12
	<i>Refreshing Source Files</i>	2-13
	<i>Creating Projects</i>	2-14
	<i>Moving or Copying Files in Projects</i>	2-15
	<i>Including Objects in Projects</i>	2-17
	<i>What's Next?</i>	2-18
3	Setting Verification Options	
	<i>Setting Workspace Verification Options</i>	3-2
	<i>Specifying Options on the Legacy Dialects Tab</i>	3-2
	<i>Specifying Options on the Settings Tab</i>	3-5
	<i>Identifying System Programs</i>	3-14
	<i>Setting Project Verification Options</i>	3-16
	<i>Specifying the Processing Environment</i>	3-21
	<i>Optimizing Verification for Advanced Program Analysis</i>	3-22
	<i>What's Next?</i>	3-22
4	Verifying Files and Performing Post-Verification Tasks	
	<i>Verifying and Reverifying Source Files</i>	4-1
	<i>Performing Post-Verification Program Analysis</i>	4-2
	<i>Enabling IMS Port Analysis</i>	4-5
	<i>Generating Copybooks for AS/400 Applications</i>	4-6
	<i>Viewing Application Inventory Reports</i>	4-9
	<i>Viewing Inventory Reports</i>	4-9
	<i>Viewing Executive Reports</i>	4-10
	<i>What's Next?</i>	4-12

5	Identifying Missing or Unneeded Program Elements	
	<i>Using Reference Reports</i>	5-1
	<i>Generating Reference Reports</i>	5-2
	<i>Working with Reference Reports</i>	5-3
	<i>Detecting System Programs</i>	5-6
	<i>Using the Orphan Analysis Tool</i>	5-7
	<i>Generating Orphan Analysis Reports</i>	5-7
	<i>Exporting Reference and Orphan Analysis Reports</i>	5-12
	<i>What's Next?</i>	5-12
6	Resolving Decisions	
	<i>Understanding Decisions</i>	6-1
	<i>Resolving Decisions Manually</i>	6-3
	<i>Resolving Decisions Automatically</i>	6-6
	<i>What's Next?</i>	6-7
7	Restoring CICS Resources	
	<i>Understanding CICS Resource Views</i>	7-1
	<i>Using Resource Retriever</i>	7-3
	<i>Exporting a CICS Resource Report</i>	7-8
	<i>What's Next?</i>	7-8
8	Using the Relaxed Parsing Tools	
	<i>Using the Missing Copybooks Resolution Tool</i>	8-2
	<i>Generating a Missing Copybooks Report</i>	8-3
	<i>Working with a Missing Copybooks Report</i>	8-4
	<i>Setting Missing Copybook Resolution Options</i>	8-5
	<i>Restoring Missing Copybooks</i>	8-7
	<i>Using the Unknown Statements Resolution Tool</i>	8-10
	<i>Generating an Unknown Statements Report</i>	8-11
	<i>Working with an Unknown Statements Report</i>	8-12
	<i>Generating Statement Groups</i>	8-14
	<i>Applying Substitution Rules</i>	8-16

Exporting and Importing Substitution Rules8-18
Setting Unknown Statements Resolution Options8-19
Exporting Relaxed Parsing Tool Reports8-20
What's Next?8-20

9 Partitioning Applications

Understanding Application Partitioning9-1
Getting Started in Application Partitioner9-4
Using Application Partitioner9-10
Working with Partitioning Information9-10
Creating and Restoring Drivers and Utilities9-14
Editing Relationship Weights9-15
Executing the Partitioning Methods9-15
Viewing Diagrams and Graphs9-19
Saving and Restoring Sessions9-19
Committing or Rolling Back Your Work9-20
What's Next?9-20

A Using the Batch Refresh Feature

Understanding the Batch Refresh Feature A-1
How the Batch Refresh Feature Is Packaged A-3
Running the Batch Refresh Feature A-3

B Identifying Interfaces for Generic API Analysis

Glossary

Bibliography

Notices

Index

Preface

The IBM Asset Transformation Workbench (ATW) is a suite of PC-based software products for analyzing, re-architecting, and transforming legacy applications. The products are deployed in an integrated environment with access to a common repository of program objects. Repository models serve as the basis for a rich set of diagrams, reports, and other documentation.

The ATW suite consists of customizable modules that together address the needs of organizations at every stage of legacy application evolution — maintenance/enhancement, renovation, and modernization.

Audience

This guide assumes that you are a corporate Information Technology (IT) professional with a working knowledge of the legacy platforms you are using the product to analyze. If you are transforming a legacy application, you should also have a working knowledge of the target platform.

Organization

This guide contains the following chapters:

- Chapter 1, “Overview,” provides an overview of the preparation process and reviews basic ATW concepts.
- Chapter 2, “Setting Up a Workspace and Projects,” describes how to register applications in the repository, and set up workspaces and projects.
- Chapter 3, “Setting Verification Options,” describes how to set options that determine the legacy dialect the parser recognizes, whether to use staged or relaxed parsing, how to treat system programs, and other verification behavior.
- Chapter 4, “Verifying Files and Performing Post-Verification Tasks,” describes how to verify and reverify source files, and how to perform key post-verification tasks.
- Chapter 5, “Identifying Missing or Unneeded Program Elements,” describes how to use reference reports and orphan analysis to identify missing or unneeded application elements.
- Chapter 6, “Resolving Decisions,” describes how to identify and resolve dynamic calls and other relationships that the parser cannot resolve from static sources in Cobol, PL/I, and Natural programs.
- Chapter 7, “Restoring CICS Resources,” describes how to identify and restore missing CICS file connectors and transactions in file and program control tables for Cobol programs.
- Chapter 8, “Using the Relaxed Parsing Tools,” describes the tools ATW provides for the resolution of program elements in Cobol applications verified with the relaxed parser.
- Chapter 9, “Partitioning Applications,” describes how to use Application Partitioner to restructure a workspace into more meaningful projects.
- Appendix A, “Using the Batch Refresh Feature,” describes how to register and verify source files in batch mode, and how to use other utilities packaged with the batch refresh feature.

- Appendix B, “Identifying Interfaces for Generic API Analysis,” describes how to enable the generic API analysis feature.
- The Glossary defines the names, acronyms, and special terminology used in this guide.

Conventions

This guide uses the following typographic conventions:

- **Bold type** — Indicates a specific area within the graphical user interface, such as a button on a screen, a window name, or a command or function.
- *Italic type* — Indicates a new term. Also indicates a document title. Occasionally, italic type is used for emphasis.
- `Monospace type` — Indicates computer programming code.
- **Bold monospace type** — Indicates input you type on the computer keyboard.
- **1A/1B, 2A/2B** — In task descriptions, indicates mutually exclusive steps: perform step A or step B, but not both.

Related Manuals

This document is part of a complete set of ATW manuals. Together they provide all the information you need to get the most out of the system.

- *Getting Started* introduces ATW. This guide provides an overview of the workbench tools and discusses basic concepts. It describes how to install the product and how to manage licenses. It also describes how to use common product features.
- *Analyzing Projects* describes how to analyze applications at the project level. This guide describes how to create diagrams of applications and how to perform impact analysis across applications. It also describes how to estimate project complexity and effort, and how to create a project dictionary.

- *Analyzing Programs* describes how to analyze applications at the program level. This guide describes how to use HyperView tools to view programs interactively and perform program analysis in stages. It also describes how to analyze procedure and data flows, search the repository, and extract business rules with HyperView.
- *Profiling Projects* describes how to create and browse Web-generated views of the repositories in your organization.
- *Creating Components* describes how to extract program components from a legacy application.
- *Parser Reference Manual* describes legacy constructions supported by Application Analyzer in reference format.
- *Architecture Reference Manual* describes legacy constructions supported by Application Architect in reference format.

Online Help

In addition to the manuals provided with the system, you can learn about the product using the integrated online help. All GUI-based tools include a standard Windows **Help** menu.

You can display:

- The entire help system, with table of contents, index, and search tool, by selecting **Help: Help Topics**.
- Help about a particular ATW window by clicking the window and pressing the **F1** key.

Many ATW tools have *guides* that you can use to get started quickly in the tool. The guides are help-like systems with hyperlinks that you can use to access functions otherwise available only in menus and other program controls.

To open the guide for a tool, choose **Guide** from the **View** menu. Use the table of contents in the **Page** drop-down to navigate quickly to a topic.

Overview



Before you can analyze a legacy application in ATW, you need to *prepare* it. Preparing an application consists of loading, or *registering*, the application in the workbench, and then *verifying* that the entire application can be understood by the workbench parser. You use reports and other tools to ensure that your application can be parsed in its entirety.

Basic Concepts

Getting Started in the ATW documentation set provides an overview of the workbench tools and workbench process. This section reviews basic workbench concepts.

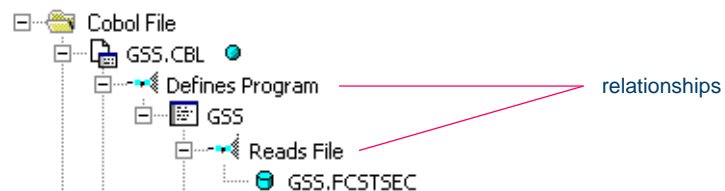
Repositories, Entities, and Relationships

ATW generates a *model* of your legacy application that serves as the basis for analyzing, re-architecting, or transforming the application. The

model is stored in a *repository* on your PC. A repository is a database of model objects, or *entities*.

The *relationships* between entities describe the ways in which the elements of your application interact. In Figure 1-1, the source file GSS.CBL file *defines* the GSS program. The program, in turn, *reads* the data file GSS.FCSTSEC.

Figure 1-1 *Entities and Relationships*

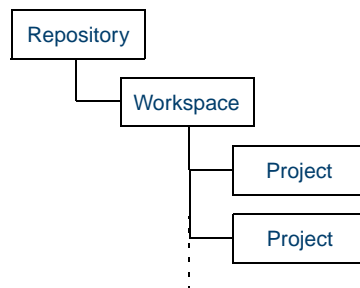


Workspaces and Projects

A *workspace* is a named container for an application or a portion of an application. You can divide a workspace up into *projects* that represent different parts of the application. You might have a project for the batch portion of your application and another project for the online portion, for example. You can also use a project to collect items for discrete tasks — all the source files affected by a change request, for example.

When you create a workspace in ATW, the system creates a repository for the workspace and a default project with the same name as the workspace. You can create new projects and move or copy entities between projects as needed.

Figure 1-2 *Repositories, Workspaces, and Projects*

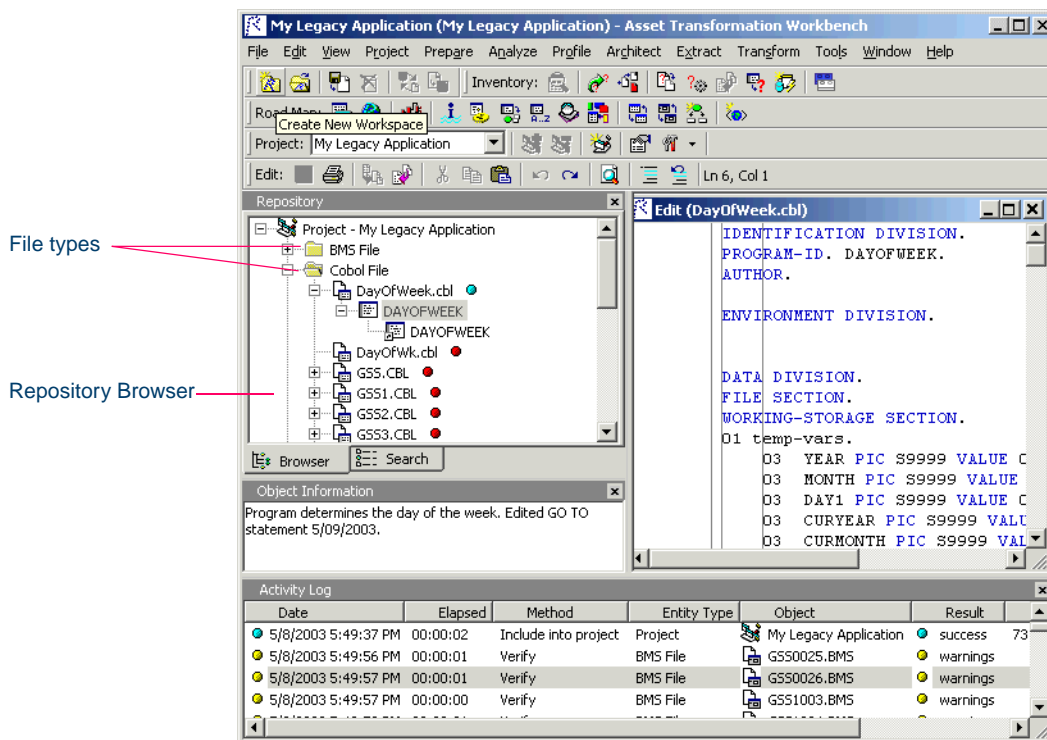


Registering Applications

When you register application files in a workspace, the workbench creates copies of the legacy files on your machine. These are the files you view and edit in the workbench tools. You can restore a file to its original state or update it to its current state as necessary. After registration, the workbench displays the contents of the current workspace in the workbench Repository Browser, organized by file type (Figure 1-3).

You can register files from any location visible to your PC, and in any number of stages. Make sure you have assigned appropriate file extensions to legacy files before you register them. You can view and add to the recognized extensions in the Extensions tab of the Workspace Registration options window (Figure 2-6 on page 2-7).

Figure 1-3 *ATW Repository Browser*



Verifying Applications

Verifying an application file ensures that its contents can be understood by the workbench parser. If you have not verified a file, the workbench displays the file in **bold** type. Since the parser has not generated an object model of the file's contents, only the file itself is displayed.

A verified file shows all the objects in the model generated for the file. Verification results are denoted as follows:




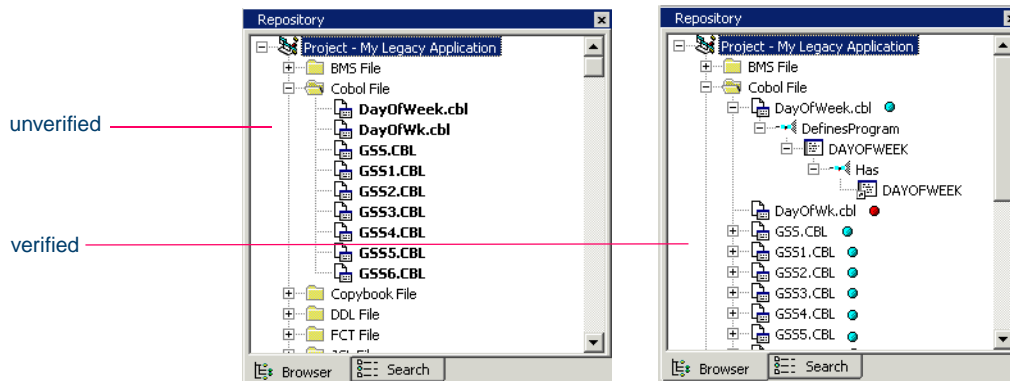
- A blue dot  means that the parser has verified the file successfully.
- A red dot  means that the parser has encountered an error in the file.
- A yellow dot  means that the parser has encountered an error in the file, but the relaxed parser has verified the file successfully. For more information on the relaxed parser, see [“Relaxed Parsing” on page 1-6](#).

Figure 1-4 compares unverified and verified workspaces. Notice in the verified workspace that the parser has built an object model for DayOfWeek.cbl, but not for DayOfWk.cbl, which contained an error. Generally, the parser creates an object model for as much of the file as it understands.

Figure 1-4 Unverified and Verified Workspaces



Inventorying Applications

Users often ask why the ATW parser encounters errors in working production systems. The reasons usually have to do with the source file delivery mechanism — incorrect versions or copybooks, corruption of special characters because of source code ambiguities, FTP errors, and so forth.

The workbench provides a robust set of tools for ensuring that all the parts of an application are available for analysis — for *inventorying applications*. This section takes a closer look at these tools.

Reference Reports and Orphan Analysis

ATW offers three related reports that you can use to identify missing or unneeded program elements in application source. The reports are based on the parser's analysis of references in verified source:

- An *unresolved report* identifies missing program elements.
- An *unreferred report* identifies unreferenced program elements.
- A *cross-reference report* identifies all application references.

The *Orphan Analysis* tool lets you analyze and resolve objects that do not exist in the reference tree for any top-level program object — *orphans*. Orphans can be removed from a system without altering its behavior.

Decision and Resource Resolution

ATW provides two tools for the resolution of program elements in verified source:

- *Decision Resolution* identifies and lets you resolve dynamic calls and other relationships that the parser cannot resolve from static sources in Cobol, PL/I, and Natural programs.
- *Resource Retriever* identifies and lets you restore missing CICS resources in Cobol and PL/I programs.

Relaxed Parsing

The *relaxed parsing* option lets you verify a source file despite errors. Ordinarily, the parser stops at a statement when it encounters an error. Relaxed parsing tells the parser to continue to the next statement.

The database the relaxed parser generates lets you resolve undefined variables in missing copybooks and incorrect or unsupported statements in program source. Once you have fixed these problems, you should be able to verify the file with the regular parser.

Tip: Relaxed parsing is also useful when you are performing less rigorous analyses that do not need every statement to be modeled — estimating the complexity of an application written in an unsupported dialect, for example.

ATW provides two tools for the resolution of program elements in Cobol applications verified with the relaxed parser:

- *Missing copybook resolution* resolves undefined variables in missing copybooks.
- *Unknown statement resolution* resolves incorrect or unsupported statements.

Partitioning Applications

Application Partitioner identifies legacy subsystems and partitions them into self-contained projects based on an analysis of repository contents. The projects can serve as the basis for coarse-grained components you create during the legacy transformation stage.

What's Next?

That's all you need to know before you prepare an ATW project. Now let's look at how you register applications in your repository, and set up workspaces and projects.

Setting Up a Workspace and Projects



Getting Started in the workbench document set described a streamlined procedure for creating a workspace and the options you can set for workbench startup. This chapter takes a closer look at how you set up workspaces and projects. For background on the relationship between repositories, workspaces, and projects, see Chapter 1, “Overview.”

Creating a Workspace

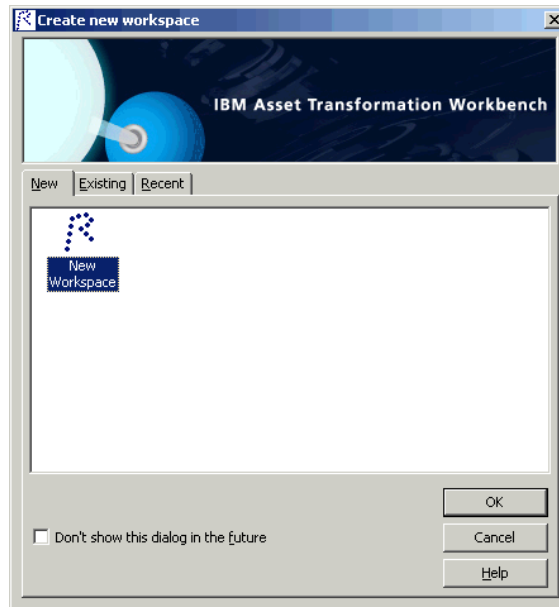
Follow the procedure below to create a new workspace. You can also export an existing project to a new workspace as described in [“To copy an entire project to a new workspace:” on page 2-16](#).

To create a new workspace:

- 1 In the ATW **File** menu, choose **New Workspace**. The Create new workspace dialog opens (Figure 2-1 on page 2-2).

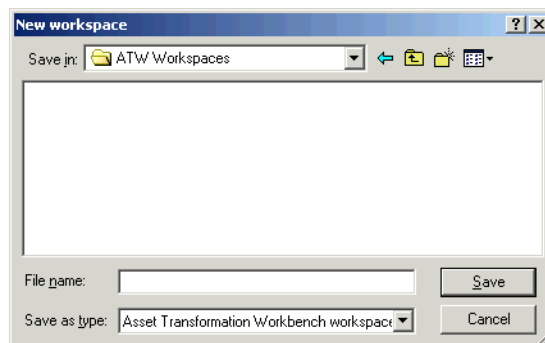
2-2 Setting Up a Workspace and Projects
Creating a Workspace

Figure 2-1 *Create New Workspace Dialog*



- 2 Double-click the **New** icon or select it and click **OK**. The New Workspace dialog opens (Figure 2-2).

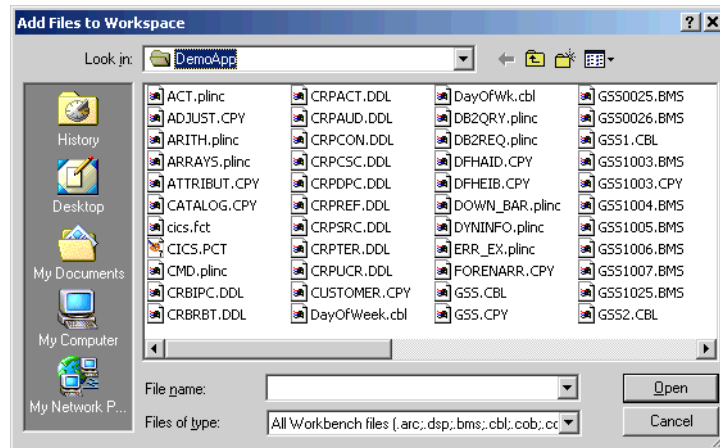
Figure 2-2 *New Workspace Dialog*



- 3 In the **Save in** drop-down, choose the folder to save the new workspace in. In the **File name** field, enter the name of the new workspace. Choose a name that describes the legacy application as

closely as possible. Click **Save**. The Add Files to Workspace dialog opens (Figure 2-3).

Figure 2-3 *Add Files to Workspace Dialog*



- 4 You can add files now or later. To add files now, select the files and click **Open**. To add files later, click **Cancel**. For more information, see [“Registering Source Files in a Workspace”](#) on page 2-6.

The new workspace is displayed in the ATW window (Figure 1-3 on page 1-3). If a workspace was already open in the window, the new workspace replaces it.

The system stores the workspace in the folder you selected in step 3. It also creates a subfolder with the same name as the workspace.

Tip: The system displays the Add Files to Workspace dialog only if the **Show Registration dialog** option is selected in the Environment tab of the User Preferences window. For information on how to set user preferences, see *Getting Started*.

Closing a Workspace

Close a workspace by choosing **Close Workspace** in the **File** menu. The system automatically saves your work.

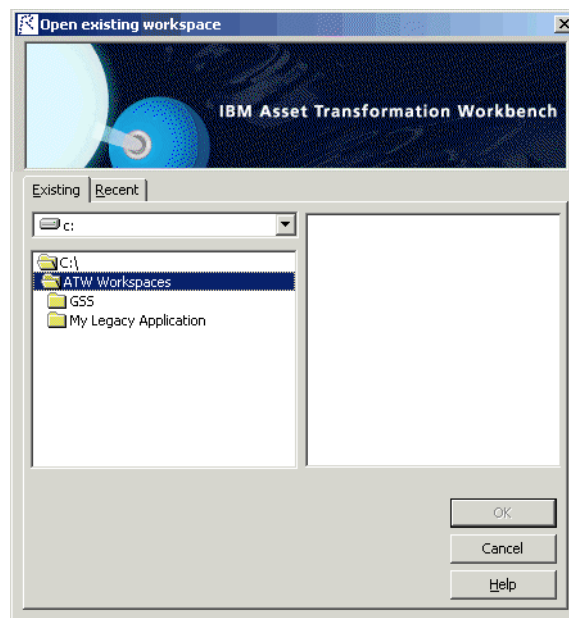
Opening a Workspace

Follow the procedure below to open an existing workspace. You can also open a workspace from the list of recently opened workspaces at the bottom of the **File** menu, or by double-clicking it in the folder you selected in [step 3 on page 2-2](#). You don't have to close a workspace before opening another workspace.

To open an existing workspace:

- 1 In the ATW **File** menu, choose **Open Workspace**. The Open existing workspace dialog opens. Click the Existing tab (Figure 2-4).

Figure 2-4 *Open Existing Workspace Dialog*

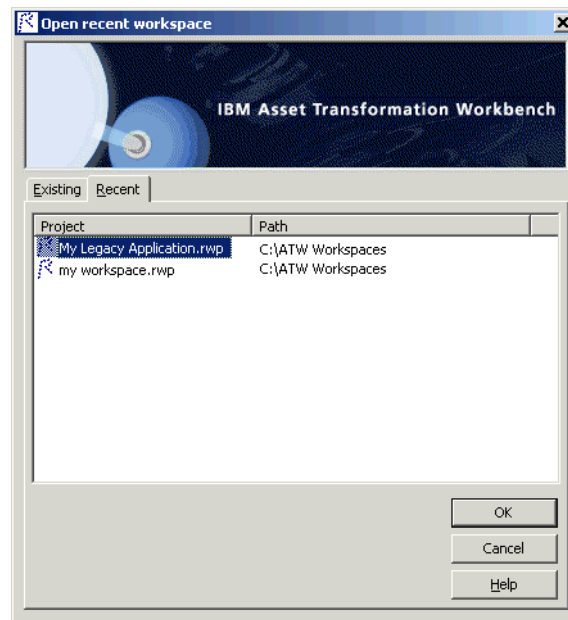


- 2A In the Existing tab, choose the drive and folder for the workspace you want to open. A list of workspaces in the selected folder is displayed in the righthand pane. Double-click the workspace you want to open, or select it and click **OK**.

- 2B Click the Recent tab to open a list of recently opened workspaces (Figure 2-5). Double-click the workspace you want to open, or select it and click **OK**.

Note: If you created the workspace in a previous release of ATW, the system prompts you to upgrade the workspace to the new release. Click **Yes**.

Figure 2-5 *Recent Tab*



- 3 The workspace is displayed in the ATW window (Figure 1-3 on page 1-3). If a workspace was already open in the window, the selected workspace replaces it.

Deleting a Workspace

Delete a workspace by choosing **Delete Workspace** in the **File** menu. You are prompted to confirm the deletion. Click **Yes**.

Registering Source Files in a Workspace

When you register files in a workspace, the system displays the files in the ATW Repository Browser, and stores copies of the files in the `\Workspace\Sources` subfolder on your machine. The files are organized in folders by file type.

You can register files from any location visible to your PC, and in any number of stages — you don't have to register the entire application at once. That's useful if you're not sure whether all of the file extensions in your legacy system are recognized by ATW, as described in [“File Extensions”](#) below.

Setting Registration Options

Workspace registration options determine the file extensions the system recognizes and whether the system converts source files to workstation encoding.

Note: Availability of options depends on your settings in the ATW configuration tool. For more information, see *Getting Started* in the ATW document set.

File Extensions

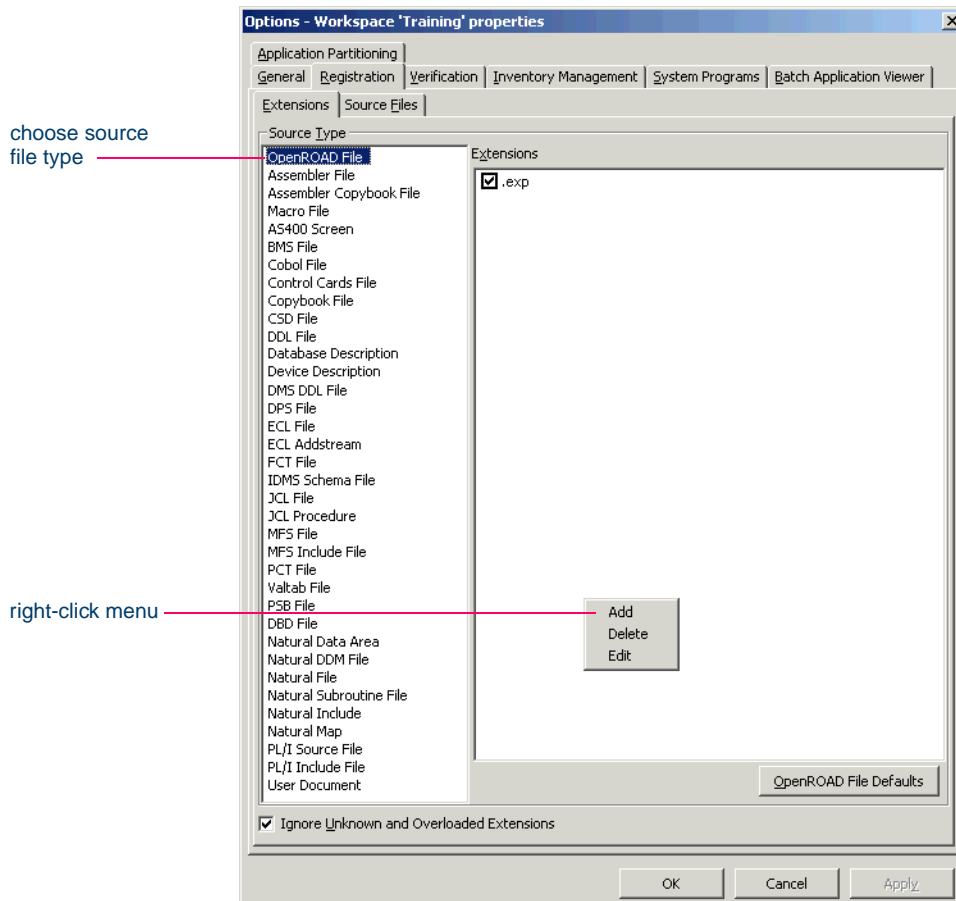
The registration process does not load source files with unknown file extensions. If you already know the extensions that will cause problems, you can make them known to the workbench before you register the files, as described in [step 2 on page 2-7](#).

If you don't know the problem file extensions, you can run the registration process anyway. As long as you set the appropriate option ([step 4 on page 2-8](#)), the workbench will generate messages indicating which extensions it did not recognize. You can then add those extensions to the list of recognized extensions and run the registration process again, this time only on the source files that failed to be registered earlier.

To set file extension options:

- 1 In the ATW **Tools** menu, choose **Workspace Options**. The Workspace Options window opens. Click the Registration tab, then the Extensions tab (Figure 2-6).

Figure 2-6 Extensions Tab



- 2 In the Source Type pane, select the source file type whose extensions you want to view. The recognized extensions for the file type are listed in the Extensions pane. Right-click in the Extensions pane and choose **Add** in the pop-up menu to add an extension to the list. The

system displays an empty text field next to a selected check box. Enter the name of the new file extension in the field and click outside the field. Make sure to enter the dot (!) Case is irrelevant.

You can edit an extension by right-clicking it and choosing **Edit** in the pop-up menu. You can delete an extension by right-clicking it and choosing **Delete** in the pop-up menu.

Deselect an extension if you do not want the system to recognize it. Recognized extensions control the filters displayed in the **Files of type** drop-down in the Add Files to Workspace Dialog (Figure 2-8 on page 2-12).

Note: If a source file does not specify an extension when it references an included file, the verification process assumes that the included file has one of the recognized extensions. If multiple included files have the same name but different extensions, the system registers the file with the first extension in the list.

- 3 For Cobol programs and copybooks, select the **Remove Sequence Numbers** check box if you want the system to replace preceding enumeration characters, or *sequence numbers*, in source lines with blanks. Sequence numbers are removed only from the source file versions maintained by the workbench.
- 4 Deselect **Ignore Unknown and Overloaded Extensions** if you want the registration process to issue warnings about unrecognized and overloaded extensions. An overloaded extension is one assigned to more than one file type.

Note: You can restore the default option settings by clicking **Source Type Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 5A Click **Apply** if you want to save your settings without dismissing the Workspace Options window.
- 5B Click **OK** if you want to save your settings and dismiss the User Workspace Options window.

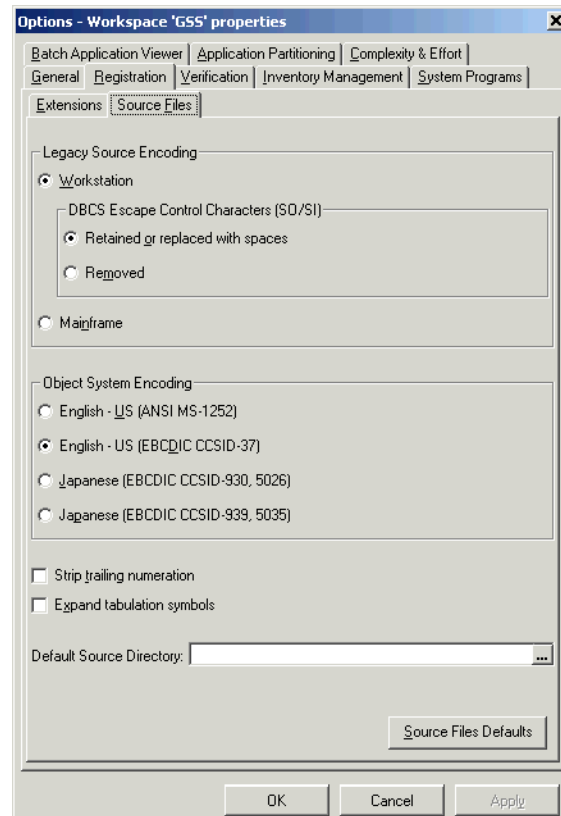
Encoding of Source Files

If the legacy application executes on a mainframe, it's usually best to convert the application source to workstation encoding. If that's not practical, you can have ATW convert it for you, as described in [step 2 on page 2-10](#).

To set source file encoding options:

- 1 In the ATW **Tools** menu, choose **Workspace Options**. The Workspace Options window opens. Click the Registration tab, then the Source Files tab (Figure 2-7).

Figure 2-7 *Source Files Tab*



2-10 Setting Up a Workspace and Projects
Registering Source Files in a Workspace

- 2** In the Legacy Source Encoding group box, choose:
 - **Workstation** if the legacy source is workstation-encoded. In the DBCS Escape Control Characters (SO/SI) group box, choose:
 - **Retained or replaced with spaces** if DBCS escape control characters were left as is or replaced with spaces.
 - **Removed** if the escape control characters were removed.

Note: Workstation-encoded Japanese source files must use Shift-JIS encoding.

- **Mainframe** if the legacy source is mainframe-encoded. When this option is selected, the registration process automatically converts legacy files to workstation-encoding. Only the source file versions maintained by the workbench are converted.
- 3** In the Object System Encoding group box, choose one of the following:
 - **English - US (ANSI MS-1252)** if the original legacy source was U.S. English ANSI-encoded (Unisys 2200 and HP3000 Cobol).
 - **English - US (EBCDIC-CCSID-37)** if the original legacy source was U.S. English EBCDIC-encoded (IBM Cobol).
 - **Japanese (EBCDIC-CCSID-930, 5026)** if the original legacy source was Japanese EBCDIC-encoded, CCSID-930, 5026.
 - **Japanese (EBCDIC-CCSID-939, 5035)** if the original legacy source was Japanese EBCDIC-encoded, CCSID-939, 5035.

During analysis and transformation, hexadecimal literals in Cobol programs and BMS files are translated into character literals according to this setting.

Important: Do not change these settings after source files are registered in a workspace.

- 4 Select the **Strip Trailing Enumeration** check box if you want the system to strip trailing enumeration characters (columns 73 through 80) from source lines. Trailing enumeration characters are removed only from the source file versions maintained by the workbench.
- 5 Select the **Expand tabulation symbols** check box if you want the system to replace tabulation symbols with a corresponding number of spaces. Tabulation symbols are replaced only in the source file versions maintained by the workbench.
- 6 In the **Default Source Directory** field, enter the root folder on your PC from which the system should refresh unresolved Cobol Copybooks, PL/I Includes, JCL Procedures, ECL Addstreams, and Natural Includes. You can type over the path in the text box or click the button to the right of the text box to browse for a new location. For more information, see [“Refreshing Source Files” on page 2-13](#).

Tip: You can restore the default option settings by clicking **Source Files Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 7A Click **Apply** if you want to save your settings without dismissing the Workspace Options window.
- 7B Click **OK** if you want to save your settings and dismiss the User Workspace Options window.

Registering Source Files

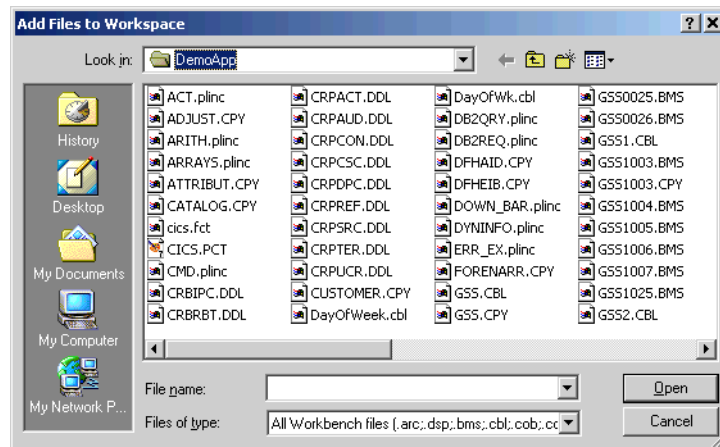
You can register source files in compressed formats (ZIP or RAR), as well as uncompressed formats. ATW automatically unpacks the compressed file and registers its contents.

Tip: If you are registering compressed files in newer archive formats that are not supported by ATW, you can specify the command line syntax and extract file for the format in the Archivers tab of the User Preferences window.

To register source files:

- 1 If you have more than one project, select the project for the source files in the Repository Browser. Drag-and-drop the source files to the selected project, or choose **Add Files to Workspace** in the **File** menu. The Add Files to Workspace dialog opens (Figure 2-8).

Figure 2-8 *Add Files to Workspace Dialog*



- 2 In the **Look in** drop-down, choose the folder that contains the legacy application files to add to the workspace. In the list of files in the center of the dialog, choose the files you want to add to the workspace. You can restrict the list of files to a certain type by choosing the type in the **Files of type** drop-down.

Tip: To add a range of listed files to the workspace, hold down the Shift key, click the first item in the range, then click the last item in the range. To add selected files, hold down the Control key, then click each file you want to add.

3 Click **Open** to add the files. If a file already exists, you are prompted to confirm that you want to replace it. Click **Yes** or **No** as appropriate.

Note: Settings in the User Preferences window determine whether you are notified that you have registered the files successfully and whether you are prompted to verify the files. For more information, see *Getting Started*.

Creating New Source Files

To create a new source file, choose **New** in the **File** menu. A dialog box opens, where you can specify the file name (with extension!) and type. To create a new source file with the same content as an existing file, select the file and choose **Save As** in the **File** menu. The system automatically registers the created files.

Refreshing Source Files

Use the ATW *refresh* feature to update source files to their current state. You can refresh all of the objects in a project or folder, or only selected objects.

The refresh looks for updated legacy source in the original location of the file or, for unresolved source, the location you specified in [step 6 on page 2-11](#). Once it finds the source, it overwrites the version of the source file maintained by the system. You do not need to re-register the refreshed source file, but you do need to re-verify it.

Note: If you are licensed to use the batch refresh feature, you can perform the refresh in batch mode, as described in [Appendix A, “Using the Batch Refresh Feature.”](#)

To refresh source files:

1 In the Repository Browser, select the project, folder, or file you want to refresh and choose **Refresh Sources from Disk** in the **File** menu.

- 2 You are prompted to confirm that you want to refresh the selected files. Click **Yes**. The system overwrites the workspace source files.

Exporting Source Files from a Workspace

Export the workspace source for a project or file to a new location by selecting the project or file and clicking **Export Sources** in the **File** menu. The source is copied to the location you specify.

Creating Projects

When you create a workspace, the system creates a default project with the same name as the workspace. You can create projects in addition to the default project when you need to analyze subsystems separately or collect items in more manageable units.

To create a project:

- 1 In the ATW **Project** menu, choose **New Project**. The Create Project dialog opens (Figure 2-9).

Figure 2-9 Create Project Dialog



- 2 Enter the name of the new project and click **OK**. The new project is displayed in the ATW window (Figure 1-3 on page 1-3). The project is selected by default.

Deleting a Project or Its Contents

Delete a project by selecting it and choosing **Delete Project** in the **Project** menu. Delete the contents of a project by selecting it and choosing **Empty Project Contents** in the **Project** menu. In either case, you are prompted to confirm the deletion. Click **Yes**.

Moving or Copying Files in Projects

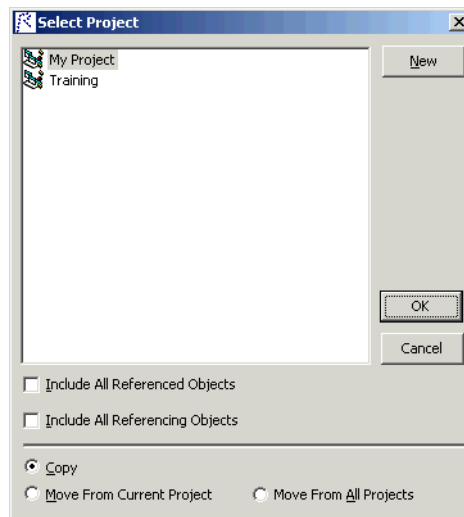
You add source files to a project as described in [“Registering Source Files” on page 2-12](#). You can also move or copy source files to different projects, or out of a project entirely to the workspace.

Tip: You can copy a source file or the contents of a project or folder to a different project by selecting it and dragging and dropping the selection, or by using the **Edit** menu choices to cut and paste the selection.

To move or copy objects between projects:

- 1 In the Repository Browser, select the project, folder, or file you want to move or copy. In the **Project** menu, choose **Copy Project Contents** (if you selected a project) or **Include into Project** (if you selected a folder or file). The Select Project window opens (Figure 2-10).

Figure 2-10 *Select Project Window*



- 2 In the Select Project window, select the project you want to move or copy the selection into. Click **New** if you want to create a new project.

- 3 Select the **Include All Referenced Objects** check box if you want to move or copy the objects in the workspace referenced by the selected object — the Cobol copybooks included in a Cobol program file, for example. Select the **Include All Referencing Objects** check box if you want to move or copy the objects in the workspace that reference the selected object.

Note: This feature is available only for verified files.

- 4 Choose one of the following:
 - **Copy** to copy the selection to the specified project.
 - **Move From Current Project** to move the selection to the specified project.
 - **Move From All Projects** to move the selection from all projects to the selected project.
- 5 Click **OK** to move or copy the selection.

To move source files from a project to the workspace:

- 1 In the Repository Browser, select the project you want to move to the workspace. In the **Project** menu, choose **Empty Project Contents** (if you selected a project) or **Exclude from Project** (if you selected a folder or file). You are prompted to confirm the removal. Click **Yes**.

To copy an entire project to a new workspace:

- 1 In the Repository Browser, select the project you want to copy to a new workspace. In the **Project** menu, choose **Export Project to New Workspace**. The New Workspace dialog opens (Figure 2-2 on page 2-2).
- 2 In the **Save in** drop-down, choose the folder to save the new workspace in. In the **File name** field, enter the name of the new workspace. Choose a name that describes the workspace as closely as possible. Click **Save**. The project is copied to the new workspace.

Including Objects in Projects

After verification, you can include referenced or referencing objects in a project to ensure a closed system. You can include all referencing objects or only “directly referencing” objects: If program A calls program B, and program B calls program C, A is said to directly reference B and indirectly reference C. You can also remove unused support objects.

To include referenced objects in a project:

- 1 To include in a project every object referenced by the objects in the project (including indirectly referenced objects), select the project in the Repository Browser and choose **Include All Referenced Objects** in the **Project** menu.

To include all referencing objects in a project:

- 1 To include in a project every object that references the objects in the project (including indirectly referencing objects), select the project in the Repository Browser and choose **Include All Referencing Objects** in the **Project** menu.

To include directly referencing objects in a project:

- 1 To include in a project every object that directly references the objects in the project, select the project in the Repository Browser and choose **Include Directly Referencing Objects** in the **Project** menu.

To move unused support objects from a project to the workspace:

- 1 To move unused support objects — Cobol copybooks, JCL procedures, PL/I include files, and so forth — from a project to the workspace, select the project in the Repository Browser and choose **Compact Project** in the **Project** menu.

Deleting Objects from a Workspace

Delete an object or the contents of a project or folder by selecting it and clicking the appropriate **Delete** choice in the **File** menu. You are prompted to confirm the deletion. Click **Yes**.

2-18 Setting Up a Workspace and Projects
What's Next?

What's Next?

Now that you have learned how to set up workspaces and projects and register legacy source files, you are ready to start verifying source files. That's the subject of the next chapter.

Setting Verification Options



Verification options determine the legacy dialect the parser recognizes, whether to use staged or relaxed parsing, how to treat system programs, and other verification behavior. *Workspace verification* options control verification behavior for the current workspace. *Project verification* options control verification behavior for the current project.

It's a good idea to become familiar with the options before verifying applications. While your workbench configuration will determine appropriate defaults (see the topic box below), not all the defaults will be suited to your needs. The staged verification feature especially may help you save time by performing only as much of the verification as you need.

How Configuration Manager Settings Affect the Options

Both workspace and project verification options may be affected by your settings in the Configuration Manager, as described in *Getting Started* in the ATW document set. If you do not configure the workbench for Unisys Cobol, for example, you will not see options related to verifying Unisys Cobol source files.

Setting Workspace Verification Options

Workspace verification options control verification behavior for the current workspace — recognized dialects, file verification options, and how the parser treats system programs.

Specifying Options on the Legacy Dialects Tab

Use the Legacy Dialects tab on the Workspace Options Verification tab to identify the dialect the application is written in. For information on supported dialects and versions, see the *ATW Release Notes*.

To specify options on the Legacy Dialects tab:

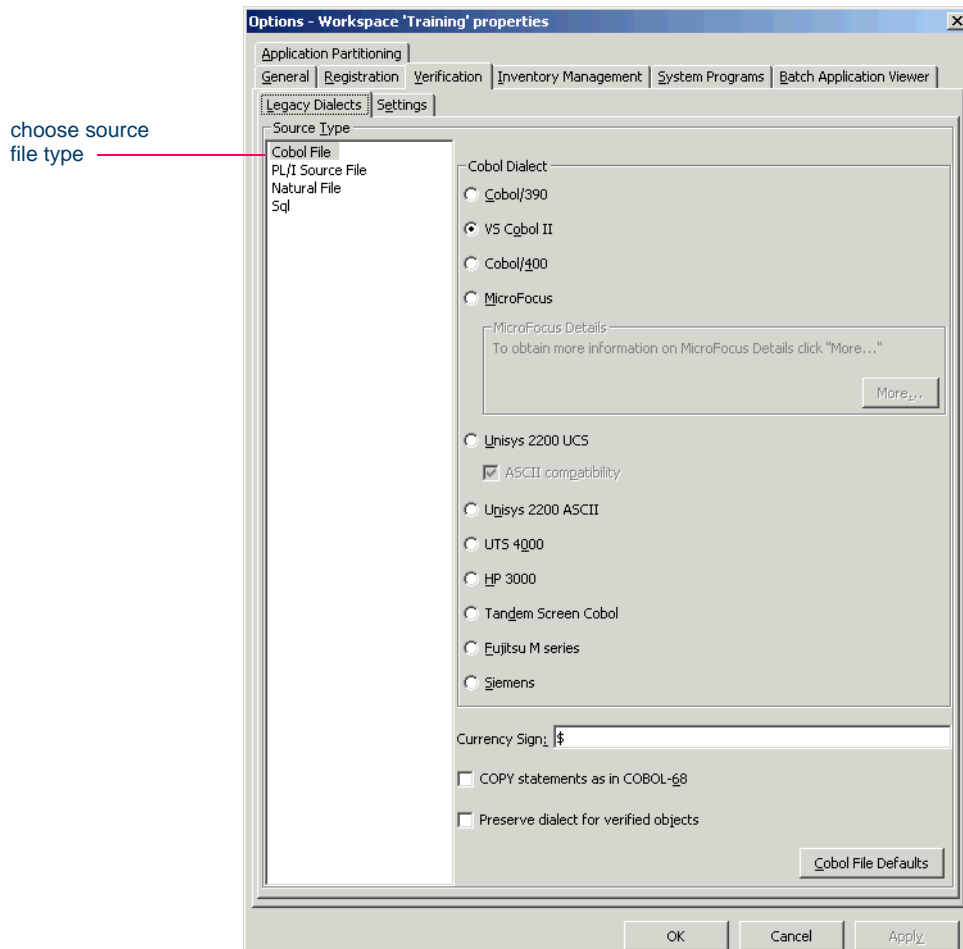
- 1 In the ATW **Tools** menu, choose **Workspace Options**. The Workspace Options window opens. Click the Verification tab, then the Legacy Dialects tab (Figure 3-1 on page 3-3).
- 2 In the Source Type pane, select the source file type whose dialects you want to view.
 - For Cobol files, go to [step 3](#).
 - For PL/I files, go to [step 7 on page 3-4](#).
 - For Natural files, go to [step 10 on page 3-4](#).
 - For applications that use SQL, go to [step 12 on page 3-4](#).
- 3 In the Cobol Dialect pane, choose the Cobol dialect used by the application.
 - For Unisys 2200 UCS Cobol, select **ASCII compatibility** if you need to ensure consistency with the ASCII version of Unisys Cobol (emulate behavior of compiler COMPAT option).
 - For MicroFocus Cobol, click **More** to open a dialog where you can set further options. In the **Binary Storage Mode** drop-down, choose Word (2, 4, or 8 bytes) or Byte (1 to 8 bytes). Select **Enable MF comments** if the application contains comments in the first position. In the PERFORM behavior group box, choose:
 - **MicroFocus** if the application was compiled with the PERFORM-TYPE option set to MF.

- **IBM family** if the application was compiled with the PERFORM-TYPE option set to OSVS.

Note: The PERFORM-TYPE option specifies the behavior of return jumps from nested PERFORM statements. For details, see the MicroFocus compiler documentation.

- 4 In the **Currency Sign** field, enter the currency symbol used by the application.

Figure 3-1 Legacy Dialects Tab (Cobol File)



3-4 Setting Verification Options
Setting Workspace Verification Options

- 5** Select **COPY statements as in COBOL-68** if the application was compiled on the mainframe with the OLDCOPY option set.
- 6** Select **Preserve dialect for verified objects** to ensure that the parser re-verifies successfully verified Cobol files with the same dialect it used when the files were first successfully verified. Go to [step 13A](#) or [step 13B](#).
- 7** In the PL/I Dialect pane, choose the PL/I dialect used by the application.
- 8** In the In margins pane, specify the current margins for PL/I source files. In the Out margins pane, specify the margins for the PL/I components to be created with the ATW Component Maker tool.
- 9** In the Special symbols pane, add or delete special symbols used in PL/I files. Go to [step 13A](#) or [step 13B](#).
- 10** In the Natural Dialect pane, choose the Natural dialect used by the application.
- 11** In the Line Number Step pane, select the line-numbering increment you want the parser to use if you choose to restore line numbers in Natural source files (see “[Restoring Line Numbers in Natural Source](#)” on page 3-5). Choose:
 - **Auto detect** if you want the parser to use a line-numbering increment based on line number references in the source code.
 - **User defined** if you want the parser to use the line-numbering increment you specify. Enter the increment in the **Value** field.Go to [step 13A](#) or [step 13B](#).
- 12** In the SQL Dialect pane, select the SQL dialect used by the application.

Note: You can restore the default option settings by clicking **Source Type Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 13A** Click **Apply** if you want to save your settings without dismissing the Workspace Options window.
- 13B** Click **OK** if you want to save your settings and dismiss the User Workspace Options window.

Restoring Line Numbers in Natural Source

Your source file delivery mechanism may have stripped line numbers from Natural source. You can restore stripped line numbers by selecting the Natural source files in the Repository Browser and choosing **Restore Line Numbers in Natural Source** in the **Edit** menu. The parser uses the increment settings you specified in [step 11 on page 3-4](#) when it restores the line numbers.

Specifying Options on the Settings Tab

Use the Settings tab on the Workspace Options Verification tab to enable staged parsing, relaxed parsing, sort card analysis for batch applications, Natural library support, and the like. Set workspace verification options for each type of file listed in the Settings tab — BMS, Cobol, DDL, and so forth.

To specify options on the Settings tab:

- 1** In the ATW **Tools** menu, choose **Workspace Options**. The Workspace Options window opens. Click the Verification tab, then the Settings tab (Figure 3-2 on page 3-6).
- 2** In the Source Type pane, select the source file type whose verification options you want to set, then select the options. Table 3-1 on page 3-7 shows the options available for each type of file.
- 3** Select **Ignore Duplicate Entry Points** to allow programs to contain duplicate entry points. The parser creates an entry point object for the first program in which the entry point was encountered and issues a warning for the second program.

Note: You can restore the default option settings by clicking **Source Type Defaults**, then choosing **Restore Defaults** in the drop-

3-6 Setting Verification Options
Setting Workspace Verification Options

down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 4A** Click **Apply** if you want to save your settings without dismissing the Workspace Options window.
- 4B** Click **OK** if you want to save your settings and dismiss the Workspace Options window.

Figure 3-2 *Settings Tab (Cobol File)*

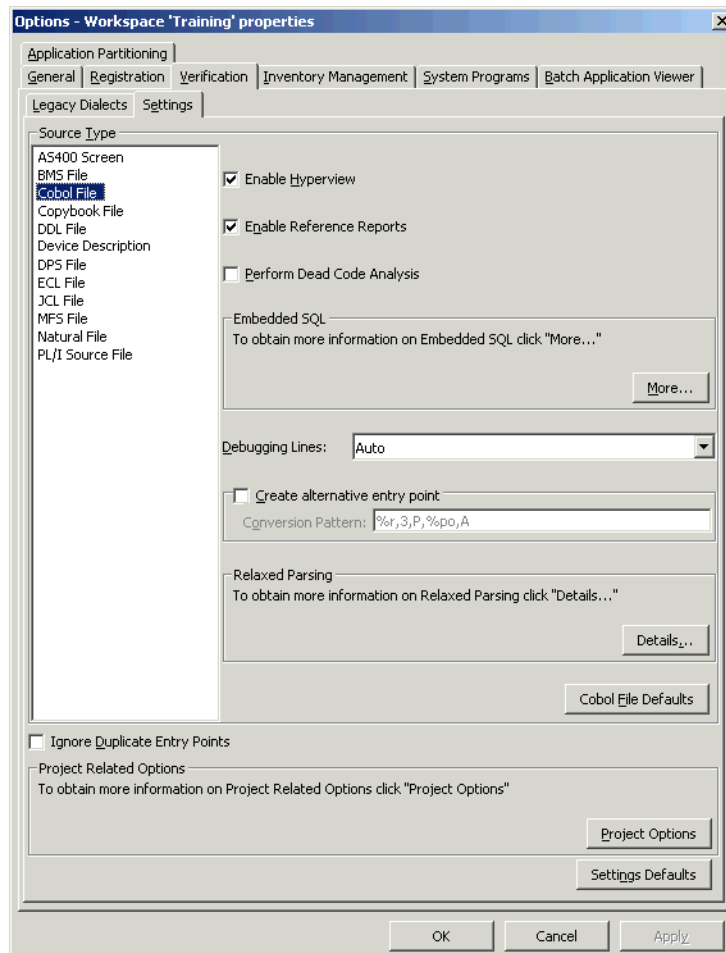


Table 3-1 *Verification Options for Source File Types*

Option	File Types	Description
Allow Implicit Instream Data	JCL	Inserts a DD * statement before implicit instream data if the statement was omitted from JCL.
Allow Keywords to Be Used as Identifiers	Cobol, Copybook	Allows Cobol keywords to be used as identifiers.
Create Alternative Entry Point	Cobol	Creates an additional entry point with a name based on the specified conversion pattern. Supports systems in which load module names differ from program IDs. For assistance, contact support services.
Debugging Lines	Cobol	Controls parsing of debugging lines. Off means parse lines as comments. On means parse lines as normal statements. Auto means parse lines based on the program debugging mode.
Perform Dead Code Analysis	Cobol	Enables collection of dead code statistics. See “Enabling Staged Parsing” on page 3-10.
Enable HyperView	Cobol, Natural, PL/I	Enables generation of HyperView information. See “Enabling Staged Parsing” on page 3-10.
Enable Reference Reports	Cobol, ECL, JCL, Natural, PL/I	Enables generation of relationships between extracted objects. See “Enabling Staged Parsing” on page 3-10.
Enable Quoted SQL Identifiers	Cobol, DDL	Allows quoted SQL identifiers.

3-8 Setting Verification Options
Setting Workspace Verification Options

Table 3-1 *Verification Options for Source File Types (continued)*

Option	File Types	Description
Ignore Text After Column 72	DDL	Allows trailing enumeration characters (columns 73 through 80) in source lines.
Libraries Support	Natural	Enables Natural library support. Choose a recognized library in the Libraries pane. Right-click in the Libraries pane and choose Add in the pop-up menu to add a library to the list. See “Enabling Natural Library Support” on page 3-13.
Perform DSN Calling Chains Analysis	JCL	Enables analysis of dataset calling chains in JCL. See “Enabling Advanced Data Flow Analysis for JCL Files” on page 3-12.
Perform System Calls Analysis	JCL	Enables analysis of system program input data to determine the application program started in a job step.
Relaxed Parsing	AS400 Screen, BMS, Cobol, Copybook, DDL, Device Description, DPS, ECL, MFS, Natural	Enables relaxed parsing. See “Enabling Relaxed Parsing” on page 3-12.
Relaxed Parsing for Embedded Statements	Cobol	Enables relaxed parsing for embedded SQL, CICS, or DLI statements.

Table 3-1 *Verification Options for Source File Types (continued)*

Option	File Types	Description
Sort Program Aliases	JCL	Enables batch sort card analysis. Choose a recognized sort utility in the Sort Program Aliases pane. Right-click in the Sort Program Aliases pane and choose Add in the pop-up menu to add a sort utility to the list. See “Enabling Sort Card Analysis for Batch Applications” on page 3-13.
SQL Statements Processor	Cobol	Specifies whether the SQL Preprocessor or Coprocessor was used to process embedded SQL statements.
Treat as IMS System Definition	PCT	Specifies whether to extract information about IMS “root programs” and corresponding PSB files. For more information, see “Enabling IMS Port Analysis” on page 4-5.
Truncate Names of Absolute Elements	ECL	Allows the parser to truncate suffixes in the names of Cobol programs called by ECL. Specify a suffix in the adjoining text box. See “Truncating Names of Absolute Elements” on page 3-13.
Use Database Schema	Cobol, PL/I	Specifies whether to associate a program with a database schema. When this option is selected, the parser collects detailed information about SQL ports that cannot be determined from program text (SELECT *). If the schema does not contain the items the SQL statement refers to, an error is generated.

Enabling Staged Parsing

File verification generates repository information in four stages, as described below. You can control which stage the workbench parser performs by setting the *staged parsing* options on the Settings tab for workspace verification options. That may save you time verifying very large applications.

Rather than verify the application completely, you can verify it one or two stages at a time, generating only as much information as you need right away. When you are ready to work with a full repository, you can perform the entire verification at once, repeating the stages you've already performed and adding the stages you haven't.

Tip: You can also improve verification performance by postponing program analysis until after verification, as described in [“Performing Post-Verification Program Analysis” on page 4-2](#)

Basic Repository Information To generate basic repository information, deselect the staged parsing options in the Settings tab. The parser:

- Generates relationships between source files — Cobol source files and copybooks, for example.
- Generates basic logical, or *extracted*, objects — programs and jobs (but not entry points or screens, for example).
- Generates Defines relationships between source files and extracted objects.
- Calculates program complexity.
- Identifies missing support files — Cobol copybooks, JCL procedures, PL/I include files, and so forth.

Note: If you generate only basic repository information when you verify an application, advanced program analysis information is not collected, regardless of your settings in the Project Options Verification tab (see steps [10-19](#) on pages 3-18-3-21).

Full Extracted Objects Information To generate complete repository information for extracted objects, select **Enable Reference Reports** in

the Settings tab. Set this option to generate reference and orphan analysis reports for extracted objects, and to enable the workbench analysis tools.

Note: If you select this option, verify all legacy objects in the workspace synchronously to ensure complete repository information.

HyperView Information To generate a HyperView *parse tree*, select **Enable HyperView** in the Setting tabs. A HyperView parse tree defines the relationships among the constructs that comprise the file being verified — its sections, paragraphs, statements, conditions, variables, and so forth.

Note: If you do not generate HyperView information when you verify an application, impact analysis, data flow, and execution flow information is not collected, regardless of your settings in the Project Options Verification tab (see steps [10-19](#) on pages 3-18-3-21).

Dead Code Statistics To generate *dead code* statistics, and to set the Dead attribute to True for dead constructs in HyperView, select **Perform Dead Code Analysis** in the Setting tabs. The statistics comprise:

- Number of *dead statements* in the source file and referenced copybooks. A dead statement is a procedural statement that can never be reached during program execution.
- Number of *dead data elements* in the source file and referenced copybooks. Dead data elements are unused level-1 structures and their members. A level-1 structure is regarded as unused if it and all its members are unused.
- Number of *dead lines* in the source file and referenced copybooks. Dead lines are source lines containing dead statements or dead data elements.

You can view the statistics in the Complexity Metrics tool, as described in *Analyzing Projects* in the ATW document set.

Enabling Relaxed Parsing

The *relaxed parsing* option lets you verify a source file despite errors. Ordinarily, the parser stops at a statement when it encounters an error. Relaxed parsing tells the parser to continue to the next statement.

The database the relaxed parser generates lets you resolve undefined variables in missing copybooks and incorrect or unsupported statements in program source, as described in [Chapter 8](#). Once you have fixed these problems, you should be able to verify the file with the regular parser.

Relaxed parsing is also useful when you are performing less rigorous analyses that do not need every statement to be modeled — estimating the complexity of an application written in an unsupported dialect, for example.

Note: Relaxed parsing may affect the behavior of other tools. You cannot generate code from legacy application source verified with the relaxed parser.

Enabling Advanced Data Flow Analysis for JCL Files

Ordinarily, ATW data flow analysis tools let you trace the flow of data into or out of a dataset only up to the program actually referenced in the JCL, whether or not that program writes to or reads from the dataset. If you need to trace the flow of data through the entire “calling chain” — that is, not only the referenced program, but also any programs that program calls, and any programs *they* call in turn:

- Select **Perform DSN Calling Chains Analysis** in the Workspace Options Settings tab (Figure 3-2 on page 3-6).
- Verify JCL files *after* you verify the source files for the programs they use. If you reverify the source file for a program, you must also reverify the JCL file that uses it.

Tip: If you verify an entire project, the workbench parses the files in appropriate order, taking account of the dependencies between JCL and program file types.

Enabling Sort Card Analysis for Batch Applications

If you use sort utilities in JCL files, you can enable sort card analysis by specifying the names of the sort utilities to the parser in the Settings tab for JCL files. The parser creates an artificial program entity that defines the inputs and outputs for each sort utility invocation. The program has a name of the form *JCLFileName.JobName.StepName.SequenceNumber*, where *SequenceNumber* identifies the order of the step in the job.

Enabling Natural Library Support

If you load Natural programs to a workspace from multiple libraries, and need to prevent library name collisions or want to maintain a list of libraries, specify the library names to the parser in the Settings tab for Natural source files. If you use this feature, the source files themselves must have names of the form *library.program.extension*.

Note: For assistance renaming Natural source files, contact support services.

Truncating Names of Absolute Elements

If you are verifying ECL files for an application in which absolute element names differ from program IDs, you can tell the parser to truncate suffixes in the names of Cobol programs called by ECL. If a Cobol program named CAP13MS.cob, for example, defines the entry point CAP13M, and an ECL program named CAP13M.ecl executes an absolute element called CAP13MA, then setting this option causes the parser to create a reference to the entry point CAP13M rather than CAP13MA.

Propagating Option Settings to Other Users

You can ensure uniform results across your team by propagating verification option settings to team members. Use the **Save To** choice in the drop-down menu below the **Defaults** push button to save your option settings to a file. You can then send the file to other users working on the same application. They use the **Load From** choice in the same menu to restore the option settings from the file.

Identifying System Programs

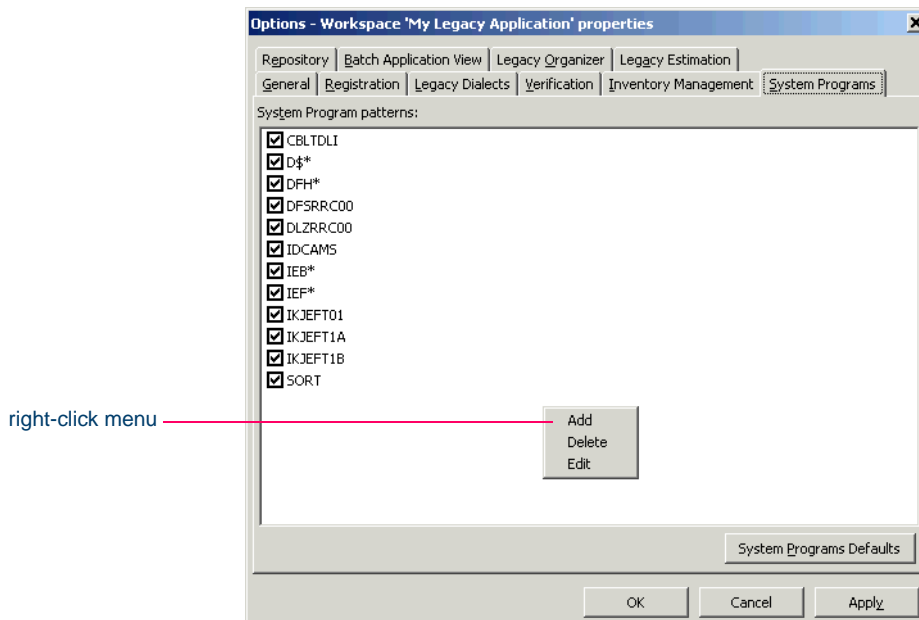
A *system program* is a generic program — a mainframe sort utility, for example — provided by the underlying system and used in unmodified form in the legacy application. You need to identify system programs to the parser so that it can distinguish them from application programs and create relationships for them with their referencing files. Use the System Programs tab in the Workspace Options window to identify system programs.

Note: Contact support services to learn how to identify system programs as drivers, and how to specify simple data flows between input/output datasets of system programs.

To identify system programs:

- 1 In the ATW **Tools** menu, choose **Workspace Options**. The Workspace Options window opens. Click the System Programs tab (Figure 3-3).

Figure 3-3 *System Programs Tab*



- 2** In the System Program Patterns pane, select the patterns that match the names of the system programs your application uses. Recognized patterns are listed in the pane.

Right-click in the pane and choose **Add** from the pop-up menu to add a pattern to the list. The system displays an empty text field next to a selected check box. Enter the pattern in the field and click outside the field.

You can edit a pattern by right-clicking it and choosing **Edit** from the pop-up menu. You can delete a pattern by right-clicking it and choosing **Delete** from the pop-up menu. Deselect a pattern if you do not want the system to recognize it.

Note: You can restore the default option settings by clicking **System Program Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 3A** Click **Apply** if you want to save your settings without dismissing the Workspace Options window.
- 3B** Click **OK** if you want to save your settings and dismiss the User Workspace Options window.

How to Detect the System Programs Your Application Uses

The most convenient way to detect the system programs an application uses is to run an *unresolved report* after verification, as described in [Chapter 5](#). Once you learn from the report which system programs are referenced, you can identify them in the System Programs tab and reverify any *one* of their referencing source files.

The reference report tool lets you bring up the System Programs tab while you are in the tool itself. Use the **System Programs** choice in the reference report **View** menu to display the tab, then follow the instructions in [step 2](#) above to identify system programs to the parser.

Setting Project Verification Options

Project verification options control verification behavior for the selected project — the transaction-processing environment, whether to resolve decisions automatically after verification, and the like. You set project verification options for Cobol, PL/I, or Natural files.

To set project verification options:

- 1 In the ATW **Tools** menu, choose **Project Options**. The Project Options window opens. Click the Verification tab. The Verification tab opens (Figure 3-4 on page 3-17).
- 2 In the Source Type pane, select the source file type whose project verification options you want to view.
 - For Cobol files, go to [step 3](#).
 - For PL/I files, go to [step 13 on page 3-20](#).
 - For Natural files, go to [step 19 on page 3-21](#).

Note: For background on the autodetection methods described in steps [3-8](#) below, see [“Specifying the Processing Environment” on page 3-21](#).

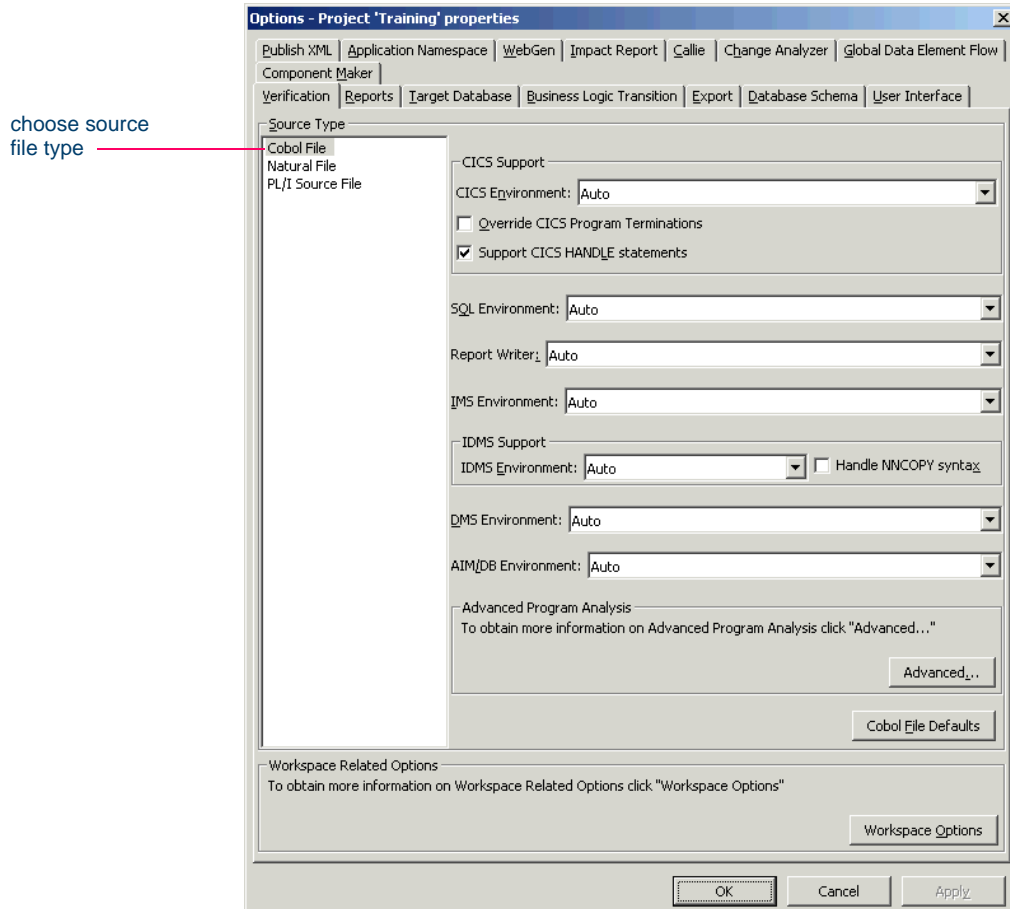
- 3 In the **CICS Environment** drop-down, specify how you want the parser to interpret CICS-related code in Cobol files.

Select **Override CICS Program Terminations** if you want the parser to interpret CICS RETURN, XCTL, and ABEND commands as not terminating program execution. When this option is selected, error-handling code after these statements is either analyzed or treated as dead code.

Select **Support CICS HANDLE statements** if you want the parser to detect dependencies between CICS statements and related error-handling statements.

- 4 In the **SQL Environment** drop-down, specify how you want the parser to interpret SQL-related code in Cobol files. In the **Report Writer Environment** drop-down, specify how you want the parser to interpret Report Writer-related code.

Figure 3-4 Verification Tab (Cobol File)

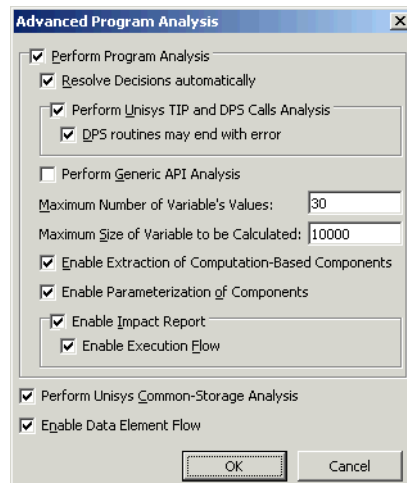


- 5 In the **IMS Environment** drop-down, specify how you want the parser to interpret IMS-related code in Cobol files.
- 6 In the **IDMS Environment** drop-down, specify how you want the parser to interpret IDMS-related code in Cobol files. Select **Handle NNCOPY syntax** if you want the parser to recognize NNCOPY statements in Cobol files.
- 7 In the **DMS Environment** drop-down, specify how you want the parser to interpret Unisys CDML statements in Cobol files.

3-18 Setting Verification Options
Setting Project Verification Options

- 8 In the **AIM/DB Environment** drop-down, specify how you want the parser to interpret Fujitsu AIM/DB-related code in Cobol files.
- 9 Click the **Advanced** button. The Advanced Program Analysis window opens (Figure 3-5).

Figure 3-5 *Advanced Program Analysis Window (Cobol File)*



- 10 Select **Perform Program Analysis** to enable program analysis and component extraction features for Cobol files in the project. If you do not want to enable each feature, you can disable individual features as necessary.

Tip: You can improve verification performance by postponing program analysis until after verification, as described in [“Performing Post-Verification Program Analysis”](#) on page 4-2

Select:

- **Resolve Decisions Automatically** if you want the parser to autoresolve decisions after successfully verifying files. For more information, see [Chapter 6, “Resolving Decisions.”](#)
- **Perform Unisys TIP and DPS Calls Analysis** if you want the parser to perform TIP and DPS calls analysis for Unisys 2200 Cobol files.

- **DPS routines may end with error** if you want the parser to perform call analysis of DPS routines that end in an error. When this option is selected, error-handling code for these routines is either analyzed or treated as dead code.
- **Perform Generic API Analysis** if you want the parser to define relationships with objects passed as parameters in calls to program interfaces, in addition to relationships with the called programs themselves. For information on how to identify the programs and parameters to the workbench, see [Appendix B, “Identifying Interfaces for Generic API Analysis.”](#)

Tip: You may be able to improve verification performance and avoid out-of-memory problems by manipulating the **Maximum Number of Variable’s Values** and **Maximum Size of Variable to Be Calculated** fields. For more information, see [“Optimizing Verification for Advanced Program Analysis” on page 3-22.](#)

- **Enable Extraction of Computation-Based Components** if you want to enable computation-based componentization.
 - **Enable Parameterization of Components** if you want to enable parameterized structure- and computation-based componentization.
 - **Enable Impact Report** if you want to enable the HyperView Impact Report tool and Impact pane. Select **Enable Execution Flow** if you want to enable the HyperView Execution Path pane.
- 11** Select **Perform Unisys Common-Storage Analysis** if you want the parser to include in the analysis for Unisys Cobol files variables that are not explicitly declared in CALL statements but participate in interprogram communications.

Note: You must set this option to include Unisys Cobol common storage variables in impact traces and global data flow diagrams.

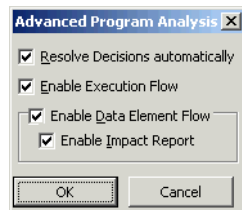
- 12** Select **Enable Data Element Flow** if you want to enable the Global Data Flow and Change Analyzer tools. Go to [step 20A](#) or [step 20B](#).

- 13 In the Transaction Environment pane, specify how you want the parser to interpret CICS-related code in PL/I files. For more information, see [“Specifying the Processing Environment” on page 3-21](#).

Note: The Auto setting is not available for PL/I.

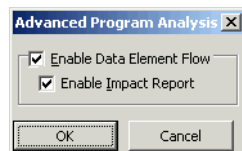
- 14 Click the **Advanced** button. The Advanced Program Analysis window opens (Figure 3-6 on page 3-20).

Figure 3-6 *Advanced Program Analysis Window (PL/I File)*



- 15 Select **Resolve Decisions Automatically** if you want the parser to autoresolve decisions after successfully verifying files. For more information, see [Chapter 6, “Resolving Decisions.”](#)
- 16 Select **Enable Execution Flow** if you want to enable the Hyper-View Execution Path pane for PL/I programs.
- 17 Select **Enable Data Element Flow** if you want to enable the Global Data Flow and Change Analyzer tools for PL/I programs. Select **Enable Impact Report** if you want to enable the Impact Report tool for PL/I programs. Go to [step 20A](#) or [step 20B](#).
- 18 Click the **Advanced** button. The Advanced Program Analysis window opens (Figure 3-7).

Figure 3-7 *Advanced Program Analysis Window (Natural File)*



- 19** Select **Enable Data Element Flow** if you want to enable the Global Data Flow and Change Analyzer tools for Natural programs. Select **Enable Impact Analysis** if you want to enable the HyperView Impact Report tool for Natural programs. Go to [step 20A](#) or [step 20B](#).

Note: You can restore the default option settings by clicking **Source Type Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

- 20A** Click **Apply** if you want to save your settings without dismissing the Project Options window.
- 20B** Click **OK** if you want to save your settings and dismiss the Project Options window.

Specifying the Processing Environment

The ATW parser autodetects the *environment* a file is intended to execute in based on the environment-related code it finds in the file. To ensure correct data flow, it sets up the internal parse tree for the file in a way that emulates the environment on the mainframe.

For Cobol CICS, for example, the parser treats an EXEC CICS statement or DFHCOMMAREA variable as CICS-related and, if necessary, adds the standard CICS copybook DFHEIB to the workspace; declares DFHCOMMAREA in the internal parse tree; and adds the phrase `Procedure Division using DFHEIBLK, DFHCOMMAREA` to the internal parse tree.

Autodetection is not always appropriate, of course — you may want the parser to treat a file as a transaction-processing program even in the absence of CICS- or IMS-related code. For each autodetected environment, select:

- Auto, if you want the parser to autodetect the environment for the file.
- Yes, if you want to force the parser to treat the file as environment-related even in the absence of environment-related code.

- No, if you want to force the parser to treat the file as unrelated to the environment even in the presence of environment-related code. The parser classifies environment-related code as a syntax error.

Optimizing Verification for Advanced Program Analysis

When you set the advanced program analysis options described in [step 10 on page 3-18](#), the parser calculates constant values for variables at every node in the HyperView parse tree. That's one reason why very large Cobol applications may encounter performance or memory problems during verification.

You may be able to improve verification performance and avoid out-of-memory problems by manipulating advanced program analysis options:

- In the **Maximum Number of Variable's Values** field, enter the maximum number of values to be calculated for each variable during verification for advanced program analysis. Limit is 100.
- In the **Maximum Size of Variable to Be Calculated** field, enter the maximum size in bytes for each variable value to be calculated during verification for advanced program analysis.

The lower the maximums, the better performance and memory usage you can expect. For each setting, you are warned during verification about variables for which the specified maximum is exceeded. It's usually best to increase the overflowed maximum and reverify the application.

What's Next?

That completes our survey of ATW verification options. Now let's look at how you use the workbench to verify applications and view inventory reports.

Verifying Files and Performing Post-Verification Tasks



Verifying a source file ensures that its contents can be understood by the workbench parser. For each successfully verified file, the parser builds an object model that serves as the basis for diagrams, reports, and other documentation. For an unsuccessfully verified file, the parser builds an object model for as much of the file as it understands. The workbench stops parsing the file at the statement at which it encountered an error.

Note: For background on the verification process, see [“Verifying Applications” on page 1-4](#). For Verification options, see [Chapter 3, “Setting Verification Options.”](#)

Verifying and Reverifying Source Files

You can verify a single file, a group of files, all the files in a folder, or all the files in a project. If you verify an entire project, the workbench parses the files in appropriate order, taking account of likely dependencies between file types.

To verify source files:

- 1 In the Repository Browser, select the project, folder, or files you want to verify and choose **Verify** in the **Prepare** menu.
- 2 The parser builds an object model for each successfully verified file. For an unsuccessfully verified file, the parser builds an object model for as much of the file as it understands. Verification results are displayed in the Activity Log.

Note: If your application uses AS/400-style copy DDS statements, you need to generate copybooks for the application *before* you verify Cobol files, as described in [“Generating Copybooks for AS/400 Applications”](#) on page 4-6.

How the System Refreshes the Repository

When you edit a source file in ATW, the system recursively checks every repository object that may be affected by the edit — *refreshes* the repository. If the edit *invalidates* the object, you need to reverify the source file that contains it. The file with the invalidated object is displayed in **bold** type in the Repository Browser.

Invalidating Objects before Reverification

You can save time reverifying very large applications by invalidating some or all of the source files in them before you reverify. You can invalidate a single file, a group of files, all the files in a folder, or all the files in a project. In the Repository Browser, select the project, folder, or files you want to invalidate and choose **Invalidate Selected Objects** in the **File** menu. Invalidated files are displayed in **bold** type in the Repository Browser.

Performing Post-Verification Program Analysis

Much of the performance cost of program verification is incurred by the program analysis features described in steps [10-19](#) in [Chapter 3](#). These features enable impact analysis, data flow analysis, and similar tasks.

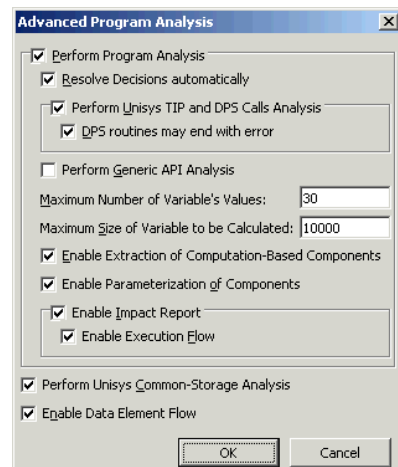
You can improve verification performance by postponing some or all of advanced program analysis until after verification. Use the *post-verifica-*

tion program analysis feature to collect the remaining program analysis information without having to reverify your entire legacy program.

To perform post-verification program analysis, select the project verification options for each program analysis feature you want to enable (Figure 4-1). In the Repository Browser, select the programs you want to analyze (or the entire project) and choose **Analyze Program** in the **Prepare** menu.

Note: Source files must have been verified with the **Enable Reference Reports** and **Enable HyperView** options selected in the Settings tab for workspace verification options, as described in [“Enabling Staged Parsing”](#) on page 3-10.

Figure 4-1 *Advanced Program Analysis Project Options (Cobol File)*



The system collects the required information for each analysis feature you select. And it does so incrementally — if you verify a Cobol source file with the **Enable Data Element Flow** option selected, and then perform post-verification analysis with both that option and the **Enable Impact Analysis** option selected, only impact analysis information will be collected.

The same is true for information collected in a previous post-verification analysis. In fact, if all advanced analysis information has been collected

for a program, the post-verification analysis feature simply will not start. In that case, you can only generate the analysis information again by re-verifying the program.

Restrictions

There are a few cases in which analysis information is not collected incrementally:

- For PL/I programs, selecting **Resolve Decisions Automatically** causes information for **Enable Data Element Flow** also to be collected, whether or not it already has been collected. Select these options together when you perform program analysis.
- For Cobol programs, selecting *any* of the options dependent on the **Perform Program Analysis** option, whether during a previous verification or a previous program analysis, results in *none* of the information for those options being collected in a subsequent post-verification program analysis, with two exceptions: information *is* collected for **Enable Impact Report** and **Enable Execution Flow**, as long as you have not selected the options previously.

So if you verify a program with the **Resolve Decisions Automatically** option selected, then perform a subsequent program analysis with the **Perform Generic API Analysis** option selected, API analysis information is not collected. Whereas if you perform the subsequent program analysis with the **Enable Impact Report** option selected, impact analysis information *is* collected.

Similarly, if you perform program analysis with the **Enable Impact Report** option selected, then perform a subsequent program analysis with the **Enable Parameterization of Components** option selected, no parameterization information is collected. Whereas if you perform the subsequent program analysis with the **Enable Execution Flow** option selected, execution flow information is collected.

What this suggests is that, with the exception of **Enable Impact Report** and **Enable Execution Flow**, you should select all of the **Perform Program Analysis** options you are going to need for program analysis the *first* time you collect analysis information, whether during verification or subsequent post-verification analysis.

Enabling IMS Port Analysis

It is virtually impossible to determine from program code the database segments or screens an IMS program operates on. Only an application-wide analysis can trace PSB usage through the entire application call sequence.

To determine the types of database operation — insert, read, update, or delete — an IMS program performs, and to list in the browser each of the database segments or screens the operation may be performed on, select a project and choose **IMS Analysis** in the **Prepare** menu.


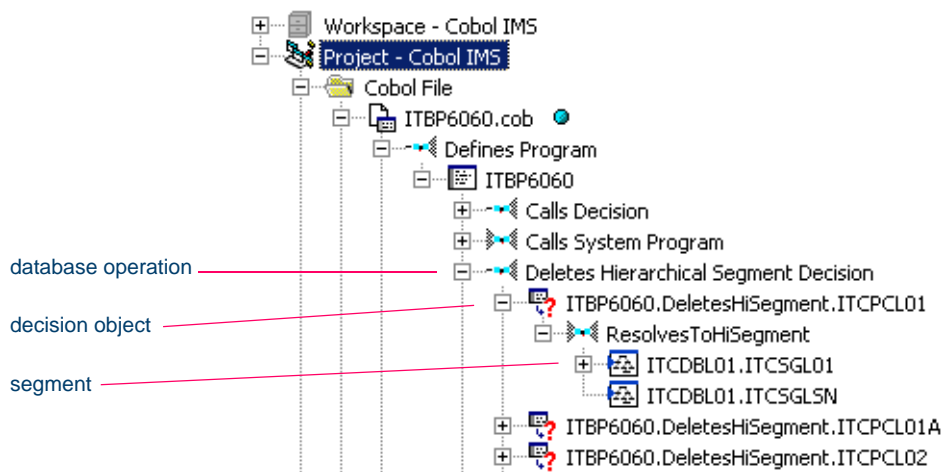
Figure 4-2 shows typical results. The objects marked with the  icon are abstract *decision objects*, indicating that the database operation — in this case, Deletes — has been resolved to multiple segments.

Figure 4-2 *IMS Port Analysis Results*



Setup

The IMS port analysis feature assumes that your application identifies IMS “root programs” and corresponding PSBs in a JCL file for a batch application or a PCT file for an online application. A root program is directly invoked by IMS with a list of PCBs as parameters. It can pass these PCBs as parameters in calls to other programs.

Verify source files in the following order:

Tip: If you verify an entire project, the workbench parses the files in appropriate order, taking account of the dependencies between file types.

- DBD files (assuming you need full cross-reference information)
- MFS files
- PSB files
- Cobol or PL/I files

Note: For Cobol files, set the **Perform Program Analysis** and **Enable Data Element Flow** project verification options. For PL/I files, set the **Enable Data Element Flow** project verification option

- JCL or PCT files

Note: For PCT files, set the **Treat as IMS System Definition** option in the Workspace Verification options Settings tab.

Generating Copybooks for AS/400 Applications

Cobol programs that execute in the AS/400 environment often use copy statements that reference Data Description Specification (DDS) files or Display and Printer Specification (DSP and PRT) files rather than copybooks. If your application uses copy DDS statements to reference these types of files, you need to generate copybooks for the application before you verify Cobol source.

To generate copybooks for AS/400 applications:

- 1 Register DDS, DSP, and PRT files as described in [Chapter 2, “Setting Up a Workspace and Projects.”](#)
- 2 Verify DDS and DSP, and PRT files as described in [“Verifying and Reverifying Source Files” on page 4-1.](#)

- For each DDS file, the system creates a *database file* object with the same name as the DDS file.
 - For each DSP or PRT file, the system creates a *device file* object with the same name as the DSP or PRT file.
- 3 Resolve any missing references in DDS, DSP, and PRT files as described in [Chapter 5, “Identifying Missing or Unneeded Program Elements.”](#)
 - 4 In the Repository Browser, select the database file or device file objects and choose **Generate Copybooks** in the **Prepare** menu. The system creates a *target copybook* object for each database and device file object with a name of the form *database file.DBCOPYBOOK* or *device file.DVCOPYBOOK* in the Target Copybooks folder.
- Tip:** Select **Generate and Convert Target Copybooks in One Step** in the Generate Copybooks tab of the Project Options window (Figure 4-3 on page 4-8) if you want to generate target copybooks and convert them to physical copybooks in the same step. For more information, see [step 5](#).
- 5 Before converting target copybooks to physical copybooks, set project options that control conversion behavior. In the **Tools** menu, choose **Project Options**. The Project Options window opens. Click the Generate Copybooks tab (Figure 4-3 on page 4-8).

In the Generate Copybooks tab:

- Select **Generate and Convert Target Copybooks in One Step** if you want to generate target copybooks and convert them to physical copybooks in the same step (see [step 4](#)).
- In the Target Copybooks Conversion pane, select **Assign Converted Files to the Current Project** if you want the system to create physical copybooks in the current project.
- In the Conversion Conflicts pane, choose:
 - **Keep Old Legacy Objects** if you want the system not to overwrite existing physical copybooks.
 - **Replace Old Legacy Objects** if you want the system to overwrite existing physical copybooks.

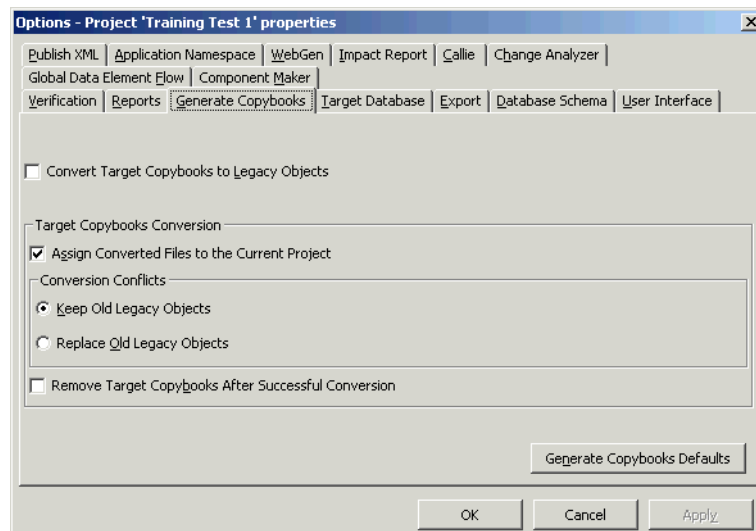
- Select **Remove Target Copybooks after Successful Generation** if you want the system to remove target copybooks from the current project after physical copybooks are generated.

Click **Apply** if you want to save your settings without dismissing the Project Options window. Click **OK** if you want to save your settings and dismiss the Project Options window.

- 6 If you have not automatically converted target copybooks to physical copybooks, select the target copybooks in the Repository Browser and choose **Convert to Legacy** in the **Prepare** menu. The system converts the target copybooks to physical copybooks with names of the form `DD_OF_database.file.CPY` or `DV_OF_device.file.CPY`.

Note: More than one physical copybook may be created for each target copybook.

Figure 4-3 *Generate Copybooks Tab*



Viewing Application Inventory Reports

ATW offers a high-level summary of workspace inventories and a detailed HTML report that managers can use to drill down into project inventories.

Viewing Inventory Reports

The Inventory Report gives high-level statistics for each source file type in the current workspace — size in bytes, number of lines of code, whether verified, and the like.

To generate an Inventory Report:

- 1 In the Repository Browser, select a project and choose **Inventory Report** in the **Prepare** menu. The Inventory Report window opens (Figure 4-4).

Figure 4-4 *Inventory Report*

The screenshot shows a window titled "Inventory Report - Training (Training)" with a menu bar (File, View, Help) and a table of file statistics. The table has columns for Type, Size, LOC, Quantity, and Unverified. It is organized into three sections: Workspace, All Projects, and Project, each with a sub-total row and a list of file types.

	Type	Size	LOC	Quantity	Unverified
Workspace	Training	339,424	7,144	45	18
	BMS File	57,725	885	4	
	Cobol File	200,248	4,533	17	
	Copybook File	76,398	1,636	18	18
	FCT File	793	13	1	
	JCL File	3,677	60	3	
	JCL Procedure	379	12	1	
	PCT File	204	5	1	
All Projects	Total	339,424	7,144	45	18
	BMS File	57,725	885	4	
	Cobol File	200,248	4,533	17	
	Copybook File	76,398	1,636	18	18
	FCT File	793	13	1	
	JCL File	3,677	60	3	
	JCL Procedure	379	12	1	
	PCT File	204	5	1	
Project	Training	339,424	7,144	45	18
	BMS File	57,725	885	4	
	Cobol File	200,248	4,533	17	
	Copybook File	76,398	1,636	18	18
	FCT File	793	13	1	
	JCL File	3,677	60	3	
	JCL Procedure	379	12	1	
	PCT File	204	5	1	

- 2 Choose **Save As** in the **File** menu to export the report to HTML, Excel, RTF, Word, or formatted text.

Note: Excel is usually the best option if you need to manipulate the report or perform computations. For more information on exporting reports, see *Getting Started* in the ATW document set.

Viewing Executive Reports

The Executive Report offers HTML views of application inventories that a manager can use to assess the risks and costs of supporting the application:

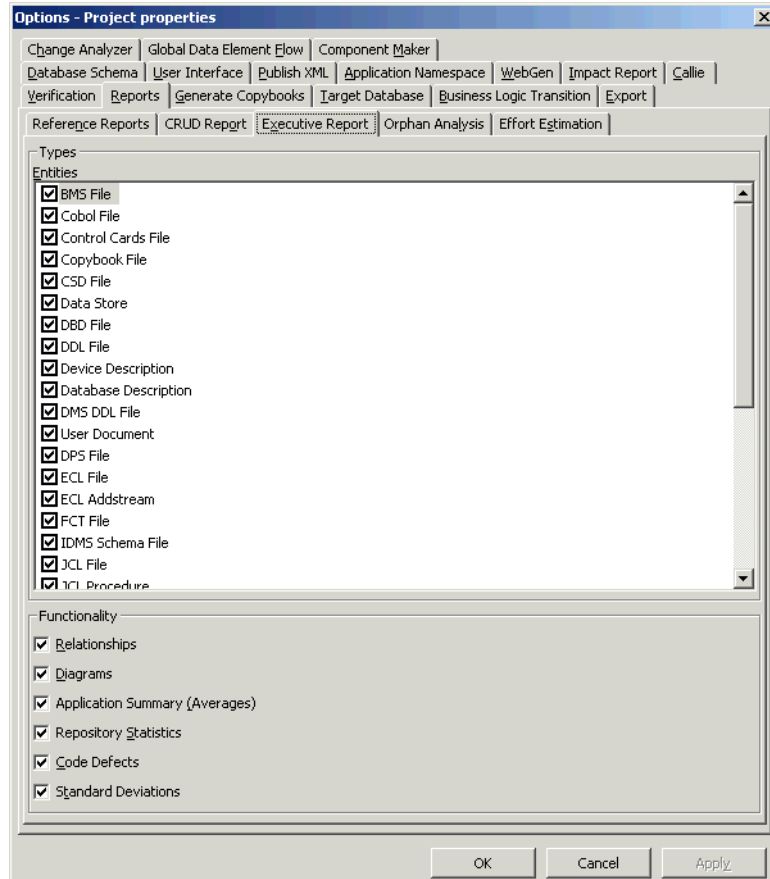
- The Application Summary view gives statistics for industry-standard metrics such as program volume, maintainability, cyclomatic complexity, and number of defects.
- The Code Defects view gives statistics for common defects — deeply nested IFs, GOTO non-exits, range overlaps, and the like — that mark programs as candidates for re-engineering.
- The Repository Statistics view gives statistics for ATW verification results and unresolved or unreferenced application elements.
- The Standard Deviations view displays graphs that plot the deviation of the programs in the application from the means for six key industry-standard metrics.

The top page in each view displays the available statistics and graphs. Click the links to view the detail for each type of statistic or graph. In the statistic or graph detail page, click the link for a program to view the detail for that program.

To generate an Executive Report:

- 1 In the Repository Browser, select a project and choose **Project Options** in the **Tools** menu. The Project Options window opens. Click the **Reports** tab, then the **Executive Report** tab (Figure 4-5 on page 4-11).

Figure 4-5 *Project Options Executive Report Tab*



2 In the Executive Report tab:

- Place a check mark next to each type of entity you want to include in the report.
- Place a check mark next to each type of function you want the report to perform.

Generally, the fewer entities and functions you choose, the better performance you can expect. You should especially consider not reporting on code defects for large Cobol applications, data stores, and, if you do not need cross-reference information, relationships.

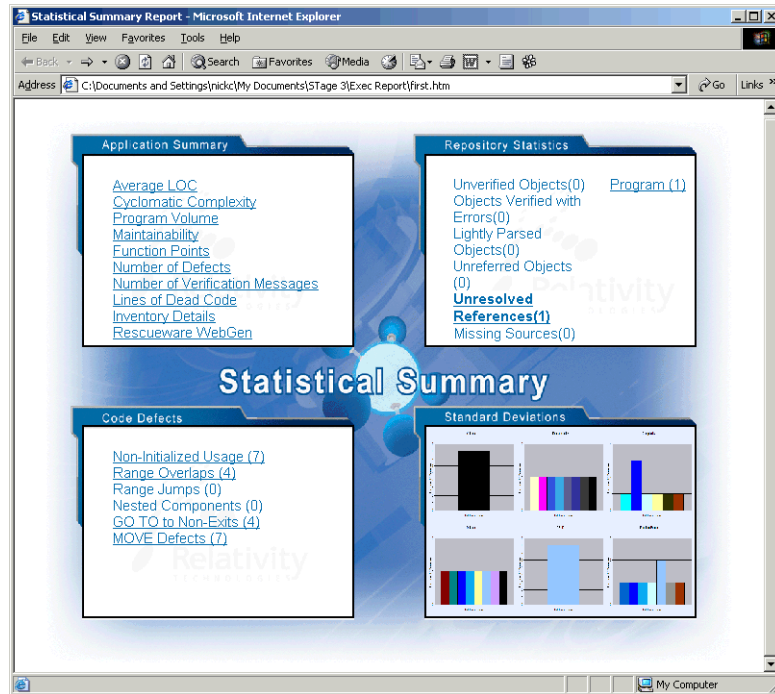
4-12 Verifying Files and Performing Post-Verification Tasks

What's Next?

Click **Apply** if you want to save your settings without dismissing the Project Options window. Click **OK** if you want to save your settings and dismiss the Project Options window.

- 3 In the Repository Browser, choose **Executive Report** in the **Prepare** menu. You are prompted to display the report now. Click **Yes** if you want to view the report immediately. The Executive Report opens in a browser (Figure 4-6). The report is stored in `\Workspace\Report\Project\root.htm`.

Figure 4-6 *Executive Report*



What's Next?

You need to correct any errors in your legacy application before the parser can generate a complete application model. You can correct coding errors in the system editor, as described in *Getting Started*. Use the tools described in the next chapter to identify missing or unneeded program elements.

Identifying Missing or Unneeded Program Elements



Reference reports identify missing or unneeded files or objects in legacy applications. The reports are based on the parser's analysis of references in verified source:


- An *unresolved report* identifies missing application elements.
- An *unreferred report* identifies unreferenced application elements.
- A *cross-reference report* identifies all application references.

The *Orphan Analysis* tool lets you analyze and resolve objects that do not exist in the reference tree for any top-level object — *orphans*. Orphans can be removed from a system without altering its behavior.

Using Reference Reports

When you verify a legacy application, the parser generates a model of the application that describes the objects in the application and how they interact. If a Cobol source file contains a COPY statement, for example, the system creates a relationship between the file and the Cobol copy-

5-2 Identifying Missing or Unneeded Program Elements
Using Reference Reports

book referenced by the statement. If the copybook doesn't exist in the repository, the system flags it as missing by listing it with a  symbol in the tree view of the Repository Browser. Reference reports let you track these kinds of referential dependencies in verified source.

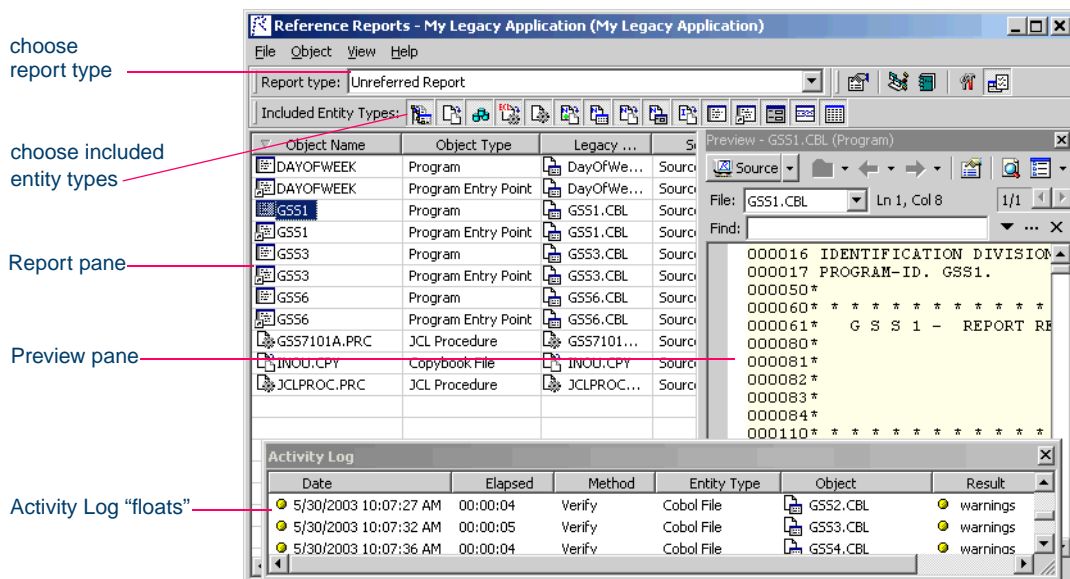
Generating Reference Reports

You generate reference reports for the current project. You can restrict the report to that project, or include references from other projects.

To generate reference reports:

- 1 In the Repository Browser, select the project for the reference report and choose **Reference Reports** in the **Prepare** menu. An empty Reference Reports window opens on top of the ATW main window.
- 2 In the **Report type** drop-down, choose the reference report type. Figure 5-1 shows an unreferred report window. The windows for the other reference reports are similar.

Figure 5-1 Unreferred Report Window



Working with Reference Reports

The Reference Report window consists of a Report pane, Preview pane, and Activity Log. The Report pane is always displayed. Select the appropriate choice in the **View** menu to show/hide the Preview pane and Activity Log.

Report Pane

The Report pane displays the objects in the reference report and their relationships. Figure 5-2 shows the Report pane for an unresolved report window. Table 5-1 on page 5-4 describes the columns in the Report pane.

Figure 5-2 *Report Pane for Unresolved Report Window*

Object Name	Object Type	Referred by	Referring Object Type	Relationship	Obj
G550025	Screen	G551	Program	Receives	
G550025	Screen	G551	Program	Sends	
G550026	Screen	G552	Program	Receives	
G550026	Screen	G552	Program	Sends	
G551003	Screen	G55	Program	Receives	
G551003	Screen	G55	Program	Sends	
G551004	Screen	G554	Program	Receives	
G551004	Screen	G554	Program	Sends	
G551005	Screen	G555	Program	Receives	
G551005	Screen	G555	Program	Sends	
G551006	Screen	G553	Program	Receives	
G551006	Screen	G553	Program	Sends	
G551007	Screen	G556	Program	Receives	
G551007	Screen	G556	Program	Sends	
G551025	Screen	G551	Program	Receives	
G551025	Screen	G551	Program	Sends	
G552025	Screen	G551	Program	Receives	
G552025	Screen	G551	Program	Sends	

Sorting Entries Click a column heading in the Report pane to sort the report entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

5-4 Identifying Missing or Unneeded Program Elements
Using Reference Reports

Viewing Properties Select an object in the Report pane and choose **Properties** in the **Object** menu to display a set of tabs with object properties. For usage information, see *Getting Started* in the ATW document set.

Restricting the Types of Objects Included in the Report Choose **Options** in the **View** menu to open the Reference Reports Options window, where you can select the types of entities included in the report.

Tip: You can also click the icons on the Included Entity Types tool bar to select and deselect included types. You can hide the tool bar by deselecting **Selection Toolbar** in the **View** menu.

Restricting References to the Current Project Choose **Restrict References to Project** in the **View** menu to limit the report to references in the current project.

Table 5-1 *Reference Report Columns*

Column Name	Report Types	Description
Object Name	All	The name of the unresolved, unreferenced, or cross-referenced object.
Object Type	All	The entity type of the unresolved, unreferenced, or cross-referenced object.
Legacy Object	Unreferred Report, Cross-Reference Report	The source file that contains the unreferenced or cross-referenced object.
Source	Unreferred Report, Cross-Reference Report	The location in the workspace folder of the source file that contains the unreferenced or cross-referenced object.
Referred by	Unresolved Report, Cross-Reference Report	The name of the referring object.

Table 5-1 *Reference Report Columns* (continued)

Column Name	Report Types	Description
Referring Object Type	Unresolved Report, Cross-Reference Report	The entity type of the referring object.
Relationship	Unresolved Report, Cross-Reference Report	The relationship between the unresolved or cross-referenced object and the referring object.
Object Description	All	The description of the unresolved, unreferenced, or cross-referenced object entered by the user on the Description tab of the Object Properties window.

Preview Pane

The Preview pane lets you browse information about the object selected in the lefthand column of the Report pane. The information available depends on the type of object selected — you see only source code for a copybook, for example, but full HyperCode for a program.

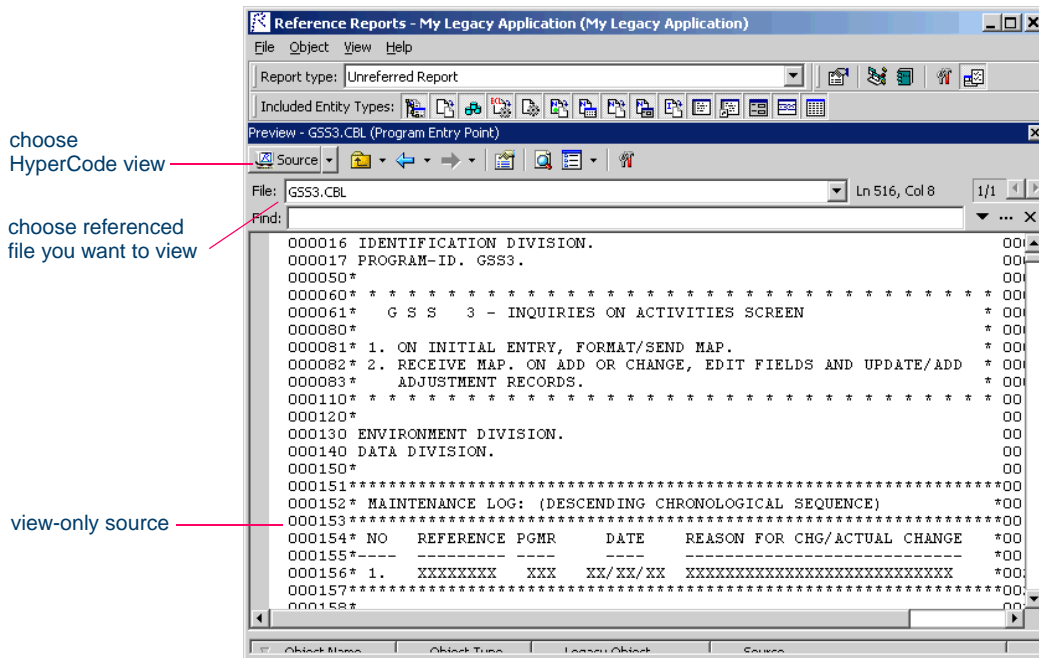
Note: “HyperCode” is shorthand for the information displayed in the ATW HyperView tool. For HyperView usage information, see *Analyzing Programs* in the ATW document set.

Choose **Preview** in the **View** menu to open the Preview pane (Figure 5-3 on page 5-6). Select an object in the Object Name column of the Report pane to view it in the Preview pane. Choose the information you want to view for the object from the **Source** drop-down. Use the search facilities to navigate to the source you want to view.

Note: The Preview pane shows a view-only copy of application source.

5-6 Identifying Missing or Unneeded Program Elements
Using Reference Reports

Figure 5-3 Preview Pane for Program Object



Detecting System Programs

A *system program* is a generic program — a mainframe sort utility, for example — provided by the underlying system and used in unmodified form in the legacy application. You need to identify system programs to the parser so that it can create relationships for them with their referencing files.

The most convenient way to detect the system programs an application uses is to run an unresolved report after verification. Once you learn from the report which system programs are referenced, you can identify them to the parser in the System Programs tab (Figure 3-3 on page 3-14) and reverify any *one* of their referencing source files.

The reference report tool lets you bring up the System Programs tab while you are in the tool itself. Use the **System Programs** choice in the **View** menu to display the tab, then follow the instructions in [step 2 on page 3-15](#) to identify system programs to the parser.

Using the Orphan Analysis Tool

Use the *Orphan Analysis* tool to determine whether an object exists in the reference tree for a top-level program object. An object that does not exist in the reference tree for *any* top-level object is called an *orphan*. Orphans can be removed from a system without altering its behavior.

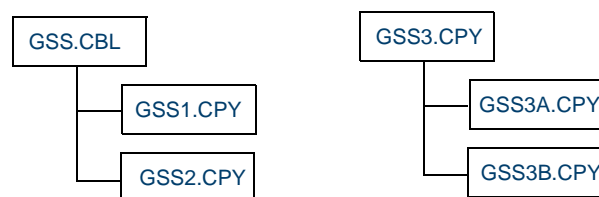
What's the difference between an orphan and an unreferenced object?

- All unreferenced objects are orphans.
- Not every orphan is unreferenced.

Consider the example shown in Figure 5-4 below. An unreferred report shows that the copybook GSS3.CPY is not referenced by any object in the project. Meanwhile, a cross-reference report shows that GSS3.CPY references GSS3A.CPY and GSS3B.CPY — these copybooks do *not* appear in the unreferred report because they *are* referenced by GSS3.CPY.

Only orphan analysis will show that the two copybooks are not in the reference tree for the GSS program — and, therefore, can be safely removed from the project.

Figure 5-4 *Unreferenced and Orphan Objects*



GSS3.CPY is unreferenced, while GSS3A.CPY and GSS3B.CPY are referenced. But all three copybooks are orphans!

Generating Orphan Analysis Reports

Orphan analysis reports show whether an object exists in the reference tree for a top-level object — whether it can be *reached* in the tree for one or more *startup* objects. You generate orphan analysis reports for the cur-

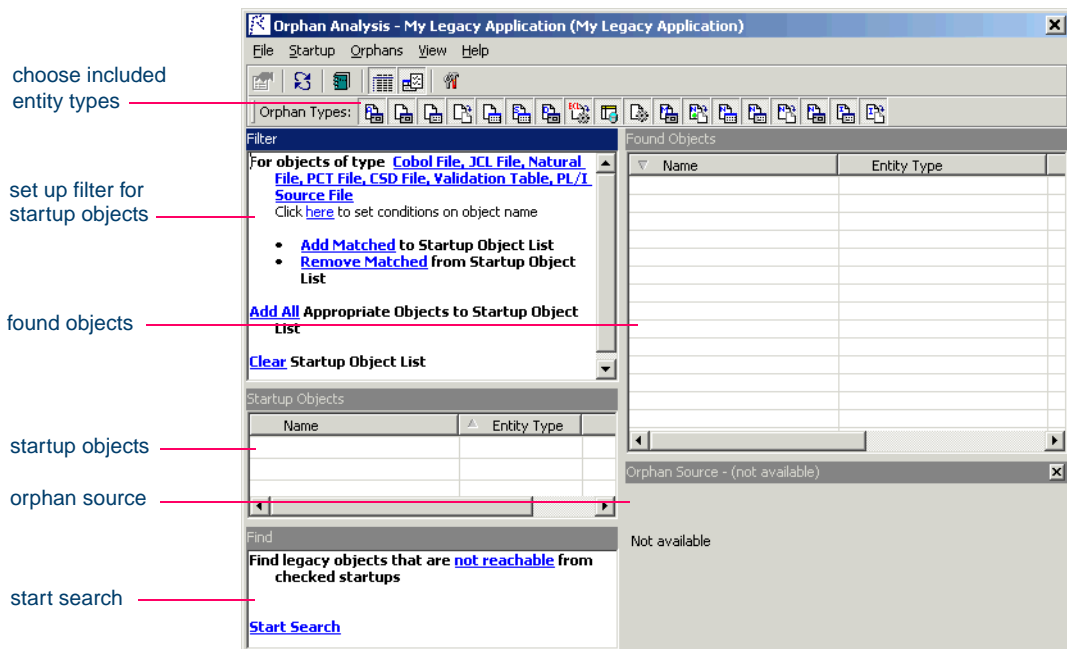
5-8 Identifying Missing or Unneeded Program Elements
Using the Orphan Analysis Tool

rent project. Unlike reference reports, the report cannot include references from other projects.

To generate orphan analysis reports:

- 1 In the Repository Browser, select the project for orphan analysis and choose **Orphan Analysis** in the **Prepare** menu. An empty Orphan Analysis window opens on top of the ATW main window (Figure 5-5).

Figure 5-5 Orphan Analysis Window (Empty)

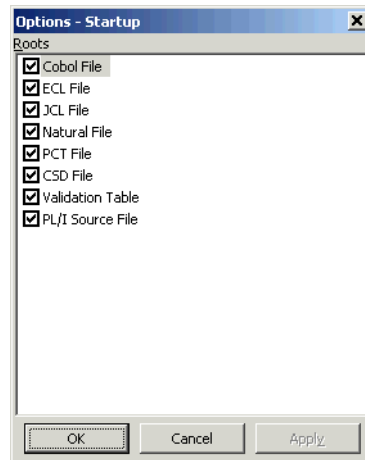


- 2 In the **Orphan Types** tool bar, choose the types of entities to include in the report by clicking the appropriate icon. Place your cursor over an icon for a moment to display a tool tip that describes the icon.

Tip: You can also set up the analysis in the Options window. Choose **Options** in the **View** menu to open the Options window.

- 3 In the Filter pane, set up a search filter for the startup objects in the analysis. You can filter on entity type, entity name, or both:
 - To filter on entity type, click on the link for an entity type in the **For objects of type** field, or if no entities are displayed, click on the **[Entity Types]** link. The Startup Options window opens (Figure 5-6). Select the types of entities you want to filter on and click **OK**.

Figure 5-6 *Startup Options Window*



- To filter on entity name, click the **here** link. The Name Options window opens (Figure 5-7 on page 5-10). The recognized name matching patterns for orphan analysis are listed in the **Like** and **Unlike** fields.

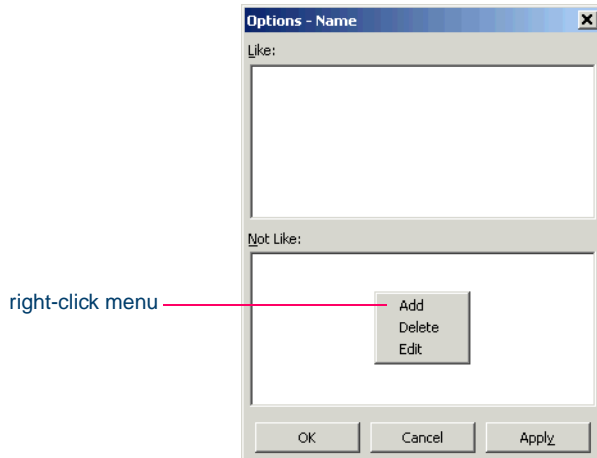
Right-click in the **Like** or **Unlike** field and choose **Add** from the pop-up menu to add a pattern to the list. The system displays an empty text field next to a selected check box. Enter the text for the pattern and click outside the field. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

You can edit a pattern by right-clicking it and choosing **Edit** from the pop-up menu. You can delete a pattern by right-clicking it and choosing **Delete** from the pop-up menu. Deselect a pattern if you

5-10 Identifying Missing or Unneeded Program Elements
Using the Orphan Analysis Tool

do not want the system to recognize it. Click **OK** when you are done.

Figure 5-7 *Name Options Window*



- 4 Click the **Add Matched** link to apply the filter. The matched objects are displayed in the **Startup Objects** pane. Place a check mark next to each startup object you want to use in the analysis.

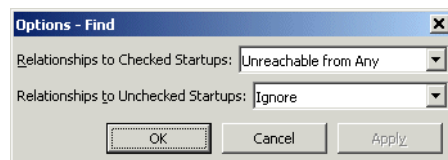
Tip: To place a check mark next to all the matched objects, choose **Select All** in the **Startup** menu, then **Mark as Startup** in the **Startup** menu. To uncheck all the matched objects, choose **Unmark** in the **Startup** menu.

To select (but not mark as startup) all objects of a given type, choose **Select:Object Type** in the **Startup** menu. You can remove a matched object from the list by selecting it and choosing **Remove from List** in the **Startup** menu.

To add all the objects in the project that would be appropriate startup objects to the list, click the **Add All** link in the Filter pane. You can filter this list by clicking the **Remove Matched** link in the Filter pane. To clear the list, click **Clear** in the Filter pane.

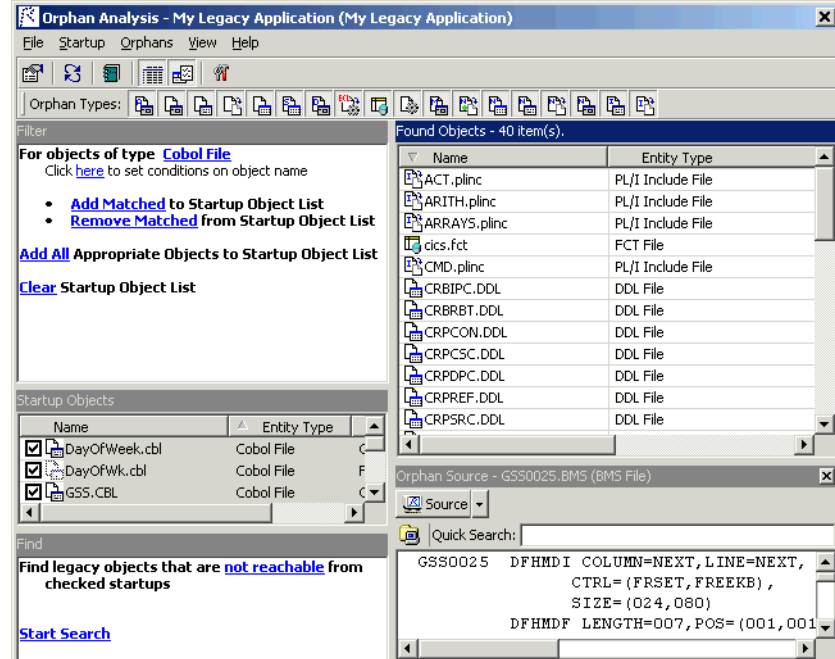
- Click the link in the **Find legacy objects...** text at the top of the pane. The Find Options dialog opens (Figure 5-8). Define the terms of the orphan search by selecting the appropriate choice in the **Relationships to Checked Startups** drop-down, the **Relationships to Unchecked Startups** drop-down, or both. Click **OK** when you are done.

Figure 5-8 *Find Options Dialog*



- Click the **Start Search** link in the Startup pane to start the search. The system shows the results of the search in the Found Objects pane (Figure 5-9).

Figure 5-9 *Orphan Analysis Window (Populated)*



5-12 Identifying Missing or Unneeded Program Elements
Exporting Reference and Orphan Analysis Reports

- 7** Choose **Report View** in the **View** menu to show the entity type and source file of the orphan in the Found Objects pane. Deselect **Report View** to show the list of orphans only.
- 8** Select an orphan to show its source code in the Orphan Source pane. The Orphan Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).

Tip: Right-click a startup object or orphan and choose **Properties** in the pop-up menu to display a set of tabs with object properties. For usage information, see *Getting Started* in the ATW document set.

- 9** Select an orphan in the Found Objects pane and choose **Exclude from Project** to remove the orphan from the project but not the workspace. Choose **Delete from Workspace** to remove the orphan from the workspace.

Tip: You can hide the Orphan Source pane and Orphan Types tool bar by selecting the appropriate choice in the **View** menu. You can display the Activity Log for the ATW session by selecting **Activity Log** in the **View** menu.

Exporting Reference and Orphan Analysis Reports

Choose **Save Report As** in the **File** menu to export a reference or orphan analysis report to HTML, Excel, RTF, Word, or formatted text. You can export a reference report to the ATW WebGen tool by clicking **Create WebGen Page** in the **File** menu. All three types of reference report are exported to the page. The page is stored in the file `\Workspace\Projects\Project\Webgen\Xref\Main.htm`.

What's Next?

Now that you have identified missing program elements in your project and removed any orphans, you can drill down deeper into the application

to locate potential problems. The tool described in the next chapter lets you identify and resolve dynamic calls and other relationships that the parser cannot resolve from static sources in Cobol, PL/I, and Natural programs.

5-14 Identifying Missing or Unneeded Program Elements

What's Next?

Resolving Decisions



You need to have a complete picture of the control and data flows in a legacy application before you can diagram and analyze the application. The parser models the control and data transfers it can resolve from static sources. Some transfers, however, are not resolved until run time. *Decision resolution* lets you identify and resolve dynamic calls and other relationships that the parser cannot resolve from static sources in Cobol, PL/I, and Natural programs.

Understanding Decisions


A *decision* is a reference to another object — a program or screen, for example — that is not resolved until run time. Consider a Cobol program that contains the following statement:

```
CALL 'NEXTPROG' .
```

The ATW parser models the transfer of control to program NEXTPROG by creating a Calls relationship between the original program and NEXTPROG.

But what if the statement read this way instead (Figure 6-1):

```
CALL NEXT.
```

where NEXT is a field whose value is only determined at run time. In this case, the parser creates a Calls relationship between the program and an abstract *decision object* called *PROG.CALL.NEXT*, and lists the decision object with a  icon in the tree view of the Repository Browser.

Manually Resolving Decisions The Decision Resolution tool creates a list of such decisions and helps you navigate to the program source code that indicates how the decision should be resolved. You may learn from a declaration or MOVE statement, for example, that the NEXT field takes either the value NEXTPROG or ENDPROG at run time. In that case, you would resolve the decision manually by telling the system to create *resolves to* relationships between the decision and the programs these literals reference.

Automatically Resolving Decisions Of course, where there are hundreds or even thousands of such decisions in an application, it may not be practical to resolve each decision manually. In these situations, you can use the *autoresolve* feature to resolve decisions automatically.

The Decision Resolution tool analyzes declarations and MOVE statements, and any other means of populating a decision point, to determine the target of the control or data transfer. The tool may not be able to auto-resolve every decision, or even every decision completely, but it should get you to a point where you can complete decision resolution manually.

Figure 6-1 *Static and Dynamic Calls*

```
CALL 'NEXTPROG' .           IF A > 0
                             THEN
                               MOVE 'NEXTPROG' TO NEXT
                             ELSE
                               MOVE 'ENDPROG' TO NEXT.
                             CALL NEXT.
```

*The call on the right can only be resolved dynamically at run time.
Use decision resolution to resolve dynamic calls.*

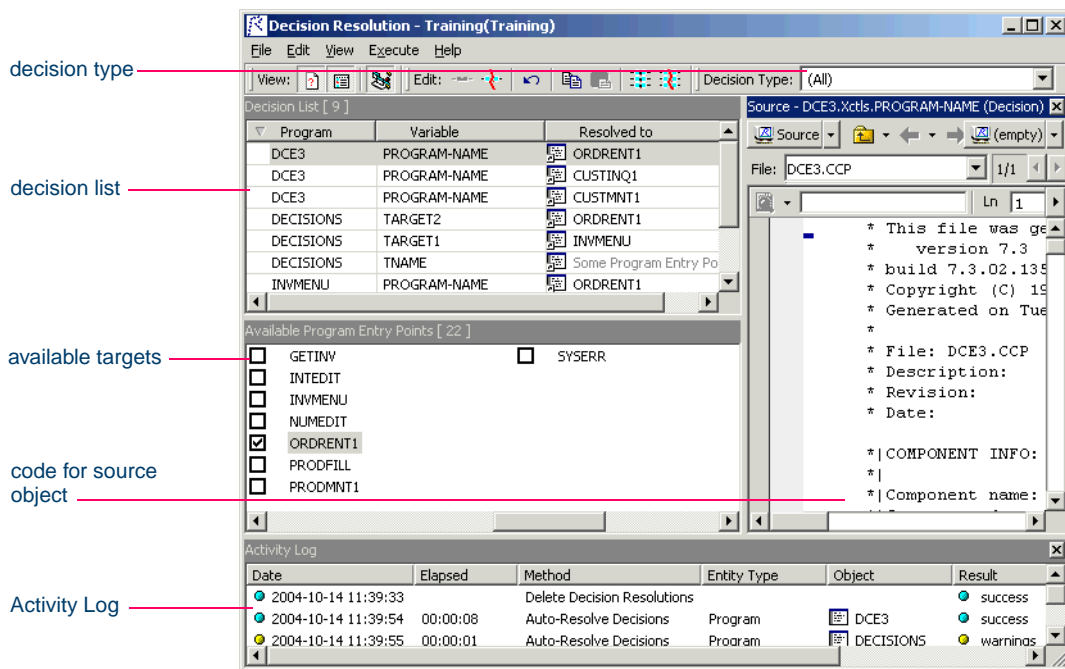
Resolving Decisions Manually

The Decision Resolution tool generates decisions for programs in the current project. You can limit the targets for decision resolution to objects in the current project, or include objects in other projects as potential targets.

To resolve decisions manually:

- 1 In the Repository Browser, select the project for which you want to resolve decisions and choose **Resolve Decisions** in the **Prepare** menu. The Decision Resolution tool window opens on top of the ATW main window (Figure 6-2).

Figure 6-2 *Decision Resolution Tool Window*



- 2 In the **Decision Type** drop-down, choose the type of decision you are interested in. A list of decisions of that type is displayed in the Decision List pane:

- The Program column lists the names of programs that contain decisions.
- The Variable column lists the program variables that require decisions.
- The Completed column lists the decision autoresolve status: True if the decision has been autoresolved, False otherwise.
- The Unreachable column lists decisions in dead code.
- The Resolved to column lists the target objects each variable resolves to — one or more entry points, for example. An unresolved decision contains the grayed-out text *Some Object*.

Tip: Click a column heading in the Decision List pane to sort the report entries by that column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

- 3 The Available *Targets* pane lists the targets in the workspace for the selected decision type. In the **View** menu, choose **Restrict to Current Project** to limit the targets to objects in the current project.
- 4 Select an entry in the Decision List pane to navigate to the decision in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).
- 5A To resolve decisions to available targets, select one or more entries in the Decision List pane and place a check mark next to one or more target objects in the Available *Targets* pane.

Tip: To select a range of entries, hold down the Shift key, click the first item in the range, then click the last item in the range. To select entries that are not in a range, hold down the Control key, then click each entry you want to select.

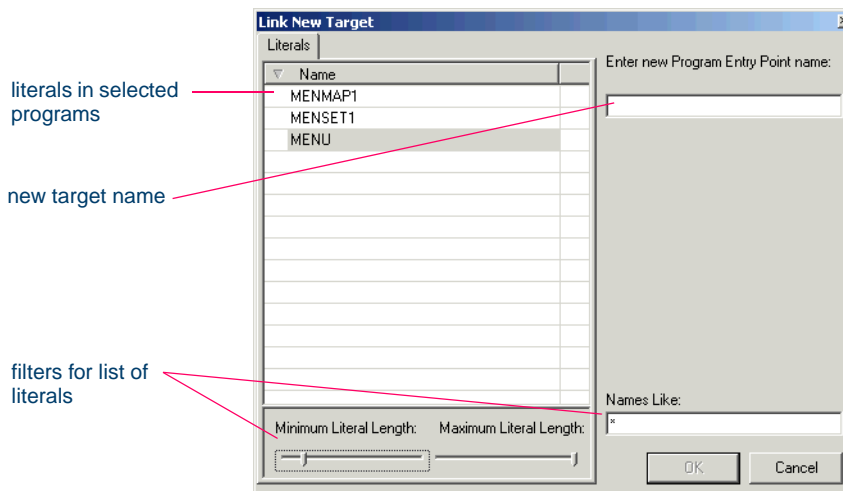
If you link an entry to multiple targets, the Decision Resolution tool creates as many entries as there are targets.

Tip: If you are linking an entry to multiple targets, you can save time by selecting the targets and choosing **Link Selected**

Targets in the **Edit** menu. Use the **Copy** choice in the **Edit** menu to copy selected targets to the clipboard, then the **Paste** choice to link the targets to an entry.

- 5B** To resolve decisions to targets not in the workspace, select one or more entries in the Decision List pane and choose **Link New Target** in the **Edit** menu. The Link New Target window opens (Figure 6-3).

Figure 6-3 *Link New Target Window*



In the Link New Target window, enter the name of the new target in the field on the righthand side of the window, or populate the field by double-clicking a literal in the list of program literals on the Literals tab. You can filter the list by using:

- The **Minimum Literal Length** slider to specify the minimum number of characters the literal can contain.
- The **Maximum Literal Length** slider to specify the maximum number of characters the literal can contain.
- The **Names Like** field to enter a matching pattern for the literal. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

When you are satisfied with your entry, click **OK**.

- 6 To delete a decision resolution, remove the check mark next to the target object. You can delete all decision resolutions for a decision type by selecting the type in the **Decision Type** drop-down and choosing **Delete All Resolutions** in the **Edit** menu. Before saving, you can undo your last change by choosing **Undo all changes** in the **Edit** menu.
- 7 In the **File** menu, choose **Save** to save the decision resolutions in the repository.

Tip: You can hide the Source pane or Activity Log window by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the Source pane or Activity Log window again.

Resolving Decisions Automatically

You can autoresolve decisions during verification by setting the **Resolve decisions automatically** option in the Project Options Verification tab, as described in [Chapter 3, “Setting Verification Options.”](#) You can also autoresolve decisions after verification, as described below.

To resolve decisions automatically:

- 1 Follow steps [1-3](#) to open the Decision Resolution tool and set the decision type and target source.
- 2A Choose **AutoResolve All Decisions** in the **Execute** menu if you want to autoresolve every decision in the current project.
- 2B Choose **AutoResolve Unresolved Decisions** in the **Execute** menu if you want to autoresolve only unresolved decisions in the current project.
- 2C Select one or more entries in the Decision List pane and choose **AutoResolve Decisions in Selected Programs** in the **Execute** menu if you want to autoresolve only decisions in the selected programs.

The Decision Resolution tool analyzes declarations, MOVE statements, and other means of populating a decision point to determine

the target of the control transfer. It places check marks next to target objects in the Available *Targets* pane.

Notes: The tool cannot autoresolve every decision. The target name may be read from a data file, for example.

The Autoresolve feature is not available for Natural programs.

- 3 Follow [step 6](#) to delete a decision resolution. Follow [step 7](#) to save decision resolutions to the repository.

What's Next?

Now that you have a complete picture of the control flow in your legacy application, you can narrow your focus to look at more specialized program elements. The tool described in the next chapter lets you identify and restore missing CICS resources in Cobol and PL/I programs.

6-8 Resolving Decisions
What's Next?

Restoring CICS Resources



The *Resource Retriever* tool lets you identify and restore missing CICS file connectors and transactions in file and program control tables for Cobol and PL/I programs. It lets you create tables for the restored objects, and add new objects to the repository as necessary. It automatically registers new objects in your project when you save the table files.

Understanding CICS Resource Views

Resource Retriever offers two views of the project:

- The *File to Data Store* view shows the relationships between File Control Table (FCT) logical files referenced by a program and existing physical data stores in the project. Use this view to define file connectors and create the File Control Table for the new file connectors.
- The *Transaction to Program Entry* view shows the relationships between Program Control Table (PCT) transactions referenced by

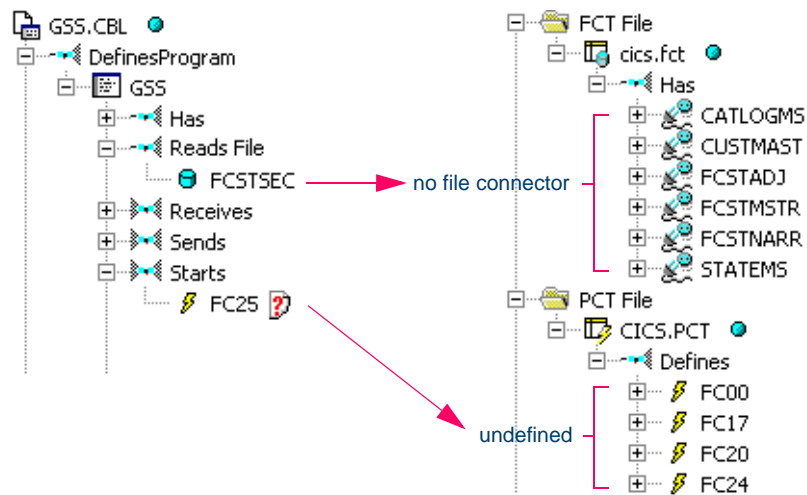
7-2 Restoring CICS Resources
Understanding CICS Resource Views

a program and existing program entry points in the project. Use this view to define transactions and create the Program Control Table for the new transactions.

Consider the example shown in Figure 7-1. The GSS.CBL program references an FCT logical file named FCSTSEC, but no file connector for the file exists in the cics.fct File Control Table. Similarly, GSS.CBL references a transaction named FC25, but the transaction is not defined in the CICS.PCT Program Control Table.

Resource Retriever not only identifies these problems, but lets you create the FCT and PCT files that resolve them!

Figure 7-1 *Restoring CICS Resources*



*FCSTSEC has no file connector. FC25 is undefined.
Use Resource Retriever to resolve these program elements.*

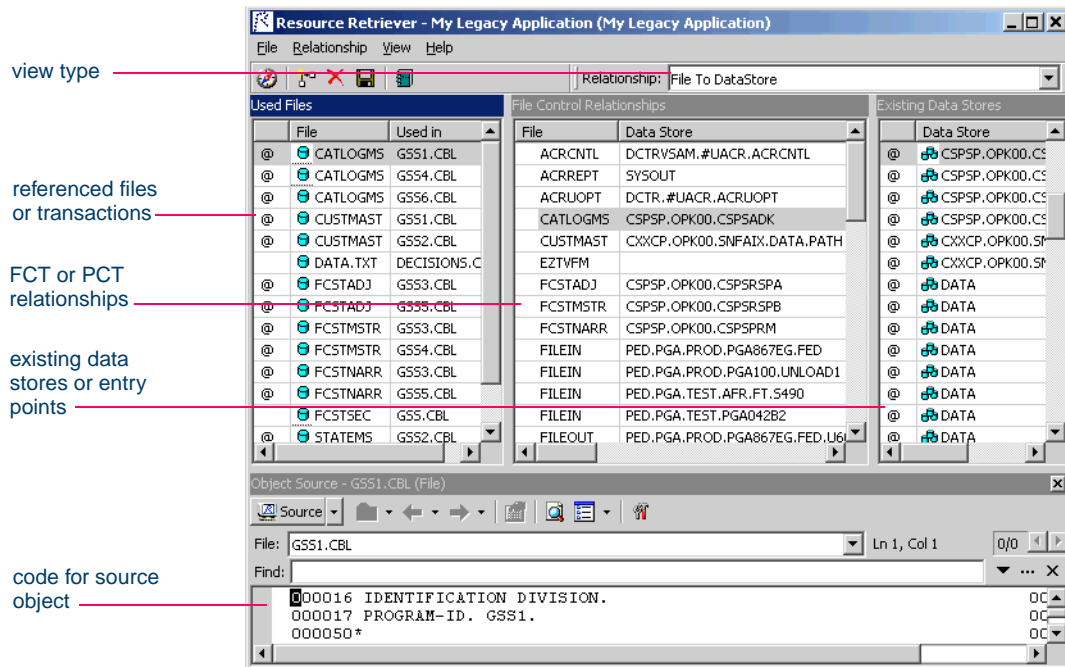
Using Resource Retriever

You generate CICS resource views for the current project. The views do not include referenced objects in other projects.

To define file connectors:

- 1 In the Repository Browser, select the project for the resource report and choose **Retrieve Missing Resources** in the **Prepare** menu. The Resource Retriever tool window opens on top of the ATW main window (Figure 7-2).

Figure 7-2 Resource Retriever Window



- 2 Choose File To Data Store in the **Relationship** drop-down. A list of relationships of that type is displayed in the File Control Relationships pane:
 - The File column lists the names of logical files defined in the project.

- The Data Store column lists the physical data stores the logical files are related to.
- The Defined in column lists the source objects that define the relationship.

Select an entry in the File Control Relationships pane to view the File Control Table in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).

Tip: Selecting an entry in the File Control Relationships pane highlights *each* of the entries for the logical files it describes in the Used Files pane. That’s useful when a logical file is referenced in multiple source files.

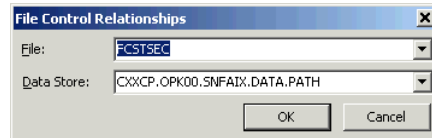
- 3 The Used Files pane displays the logical files referenced in the project and the source files that contain the references. An @ symbol next to an entry indicates that the logical file is defined in a File Control Table.


Select an entry in the Used Files pane to view the source file in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).

Tip: Click a column heading in the File Control Relationships or Used Files pane to sort the report entries by that column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

- 4 The Existing Data Stores pane displays the existing data stores in the project. An @ symbol next to an entry indicates that the data store is defined in a File Control Table. Select a data store in the Existing Data Stores pane to view its properties in the Source pane.
- 5 To create relationships between logical files and data stores, select an undefined logical file or data store and choose **Add** in the **Relationship** menu. The File Control Relationships dialog opens (Figure 7-3 on page 7-5).

Figure 7-3 *File Control Relationships Dialog*

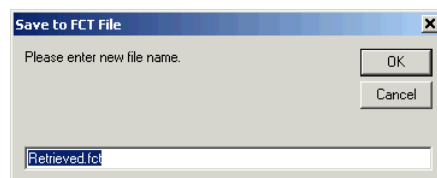


- 6 The **File** combo box displays the logical file you want to define. In the **Data Store** combo box, choose the data store you want to relate the logical file to. Click **OK**. The new relationship is added to the File Control Relationships pane. The  symbol next to the relationship indicates that it has not been assigned to a File Control Table.

Tip: You can also use this dialog to add logical file or data store objects to the repository. Enter the name of the new object in the appropriate combo box, then create the relationship for it as you would for an existing object.

- 7 To delete newly created relationships, select the relationships in the File Control Relationships pane and choose **Delete** in the **Relationship** menu.
- 8 To create the File Control Table for new relationships, select the relationships and choose **Save** in the **Relationship** menu. The Save to FCT File dialog opens (Figure 7-4).

Figure 7-4 *Save to FCT File Dialog*



- 9 Enter the name of the new File Control Table in the text box and click **OK**. The new table is registered in the repository and appears in the Defined in column for the relationship in the File Control Relationships pane.

Tip: Select the appropriate choice in the **View** menu to show/hide the Source pane and Activity Log window.

To define transactions:

- 1 In the Repository Browser, select the project for the resource report and choose **Retrieve Missing Resources** in the **Prepare** menu. The Resource Retriever window opens on top of the ATW main window (Figure 7-2 on page 7-3).
- 2 Choose Transaction To Program Entry in the **Relationship** drop-down. A list of relationships of that type is displayed in the Program Control Relationships pane:
 - The Transaction column lists the names of transactions defined in the Program Control Tables for the project.
 - The Program Entry Point column lists the program entry points the transactions are related to in the Program Control Tables.
 - The Defined in column lists the Program Control Table that defines the relationship.

Select an entry in the Program Control Relationships pane to view the Program Control Table in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).

Tip: Selecting an entry in the Program Control Relationships pane highlights *each* of the entries for the transactions it describes in the Started Transactions pane. That’s useful when a transaction is referenced in multiple source files.

- 3 The Started Transactions pane displays the transactions referenced in the project and the source files that contain the references. An @ symbol next to an entry indicates that the transaction is defined in a Program Control Table.

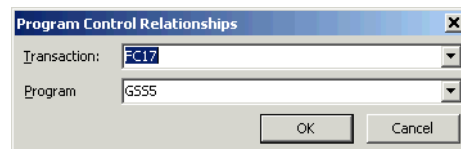
Select an entry in the Started Transactions pane to view the source file in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).


Tip: Click a column heading in the Program Control Relationships or Started Transactions pane to sort the report entries by that

column. Grab-and-drag the border of a column heading to increase or decrease the width of the column.

- 4 The Existing Program Entry Points pane displays the existing program entry points in the project. An @ symbol next to an entry indicates that the entry point is referenced in a Program Control Table. Select an entry point in the Existing Program Entry Points pane to view its source in the Source pane. The Source pane is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).
- 5 To create relationships between transactions and program entry points, select an undefined transaction or entry point and choose **Add** in the **Relationship** menu. The Program Control Relationships dialog opens (Figure 7-3).

Figure 7-5 *Program Control Relationships Dialog*



- 6 The **Transaction** combo box displays the transaction you want to define. In the **Program** combo box, choose the entry point you want to relate the transaction to. Click **OK**. The new relationship is added to the Program Control Relationships pane. The  symbol next to the relationship indicates that it has not been assigned to a Program Control Table.

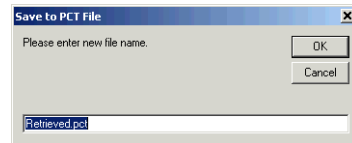
Tip: You can also use this dialog to add transaction or entry point objects to the repository. Enter the name of the new object in the appropriate combo box, then create the relationship for it as you would for an existing object.

- 7 To delete newly created relationships, select the relationships in the Program Control Relationships pane and choose **Delete** in the **Relationship** menu.

7-8 Restoring CICS Resources
Exporting a CICS Resource Report

- 8 To create the Program Control Table for new relationships, select the relationships and choose **Save** in the **Relationship** menu. The Save to PCT File dialog opens (Figure 7-6).

Figure 7-6 *Save to PCT File Dialog*



- 9 Enter the name of the new Program Control Table in the text box and click **OK**. The new table is registered in the repository and appears in the Defined in column for the relationship in the Program Control Relationships pane.

Tip: Select the appropriate choice in the **View** menu to show/hide the Source pane and Activity Log window.

Exporting a CICS Resource Report

Choose **Save Report As** in the **File** menu to export a CICS resource report to HTML, Excel, RTF, Word, or formatted text.

What's Next?

Now that you have a complete model of your legacy application in the ATW repository, you can skip to [Chapter 9](#), where you'll learn how to use Application Partitioner to restructure your workspace into more meaningful projects.

Using the Relaxed Parsing Tools



The *relaxed parsing* option lets you verify a source file despite errors. Ordinarily, the parser stops at a statement when it encounters an error. Relaxed parsing tells the parser to continue to the next statement.


The database the relaxed parser generates lets you resolve undefined variables in missing copybooks and incorrect or unsupported statements in program source. Once you have fixed these problems, you should be able to verify the file with the regular parser.

ATW provides two tools for the resolution of program elements in Cobol applications verified with the relaxed parser:

- *Missing copybook resolution* resolves undefined variables in missing copybooks.
- *Unknown statement resolution* resolves incorrect or unsupported statements.

You set the relaxed parsing option when you verify your application. For more information, see [Chapter 4, “Verifying Files and Performing Post-Verification Tasks.”](#)

Using the Missing Copybooks Resolution Tool

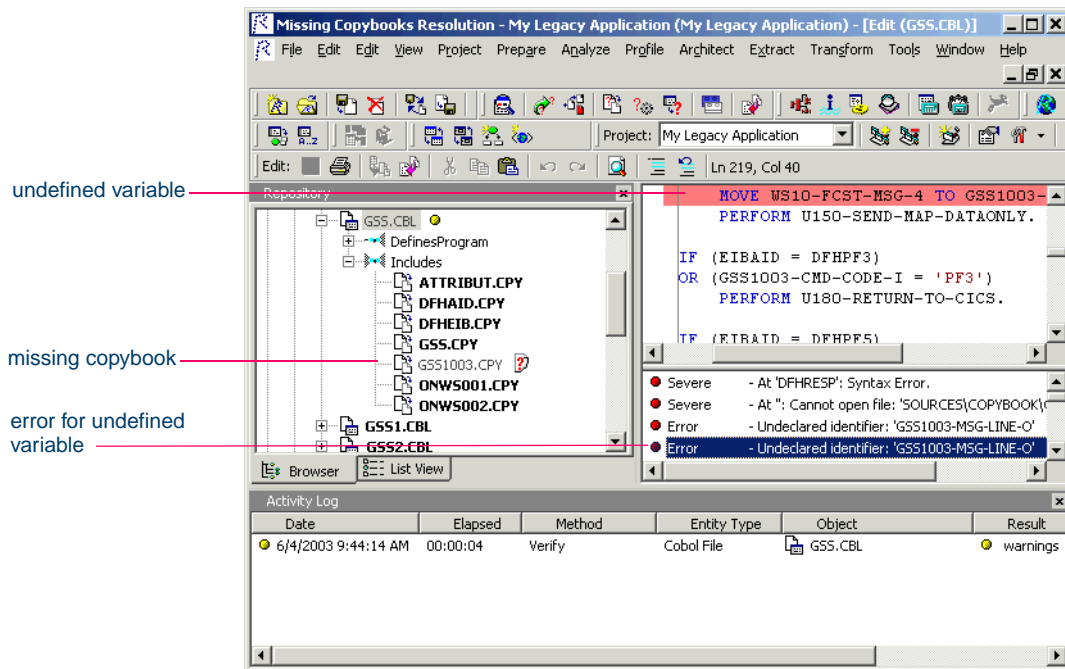
If a copybook for a Cobol application is missing from your workspace, the parser identifies the missing copybook with a  symbol in the tree view of the Repository Browser, and generates errors for undefined variables in the application program (Figure 8-1).

The Missing Copybooks Resolution tool lets you reconstruct the contents of lost copybooks. You can restore:

- Copybooks explicitly referenced in the application.
- *System copybooks* that the underlying system makes available to all applications.

You can restore explicitly referenced copybooks automatically or manually. You must restore system copybooks manually.

Figure 8-1 Missing Copybook (Relaxed Parsing Results)



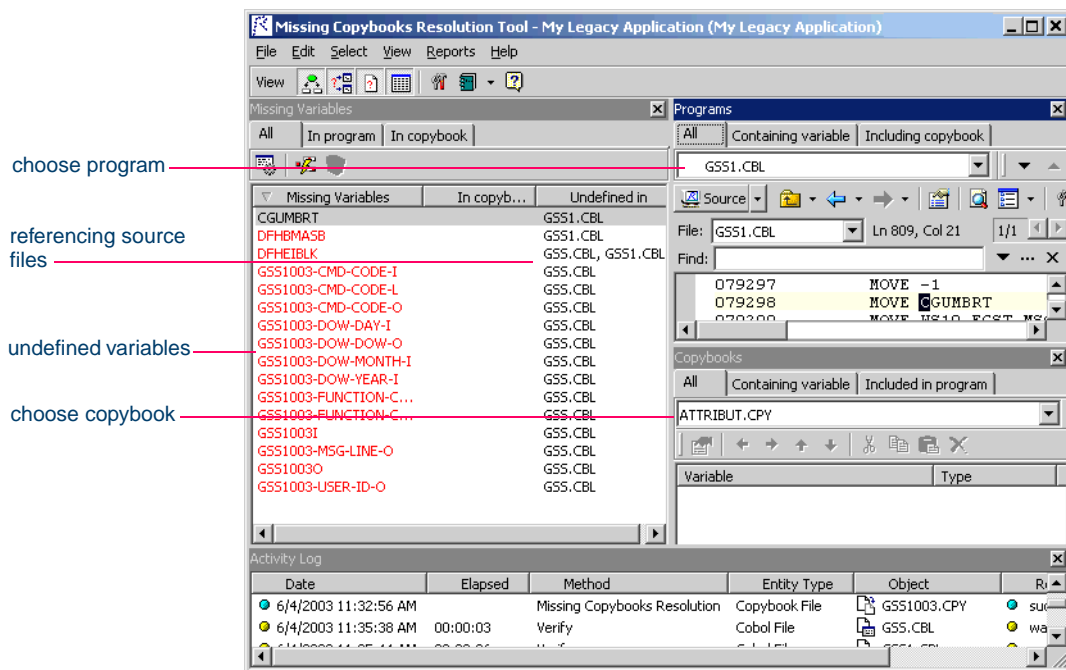
Generating a Missing Copybooks Report

You typically use a missing copybooks report to restore explicitly referenced or system copybooks. But you can also use the report simply to identify undefined variables. Once you know which variables are undefined, you can define them in existing copybooks by hand.

To generate a missing copybooks report:

- 1 In the Repository Browser, select the project for the missing copybooks report and choose **Resolve Missing Copybooks** in the **Prepare** menu. The Missing Copybooks Resolution tool window opens on top of the ATW main window (Figure 8-2).

Figure 8-2 *Missing Copybooks Report*



Working with a Missing Copybooks Report

The missing copybooks report window consists of a Missing Variables pane, Programs pane, Copybooks pane, and Activity Log. You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Missing Variables Pane

The Missing Variables pane contains three columns:

- The Missing Variables column lists the undefined variables in the workspace. Undefined variables are displayed in red. Defined variables are displayed in black. A variable defined in one source file but not in the others that use it is displayed in red.
- The In copybook column lists the copybooks that contain defined variables.
- The Undefined in column lists the source files that reference undefined variables.

Click a tab to limit the scope of the variables displayed in the pane:

- Click All to view all the unresolved variables in the workspace.
- Click In program to view only the unresolved variables in the program selected in the Programs pane.
- Click In copybook to view only the resolved variables in the copybook selected in the Copybooks pane.

Sorting Entries Click a column heading in the Missing Variables pane to sort the report entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Viewing Source Select a variable to view it in the source file displayed in the Programs pane.

Programs Pane

The Programs pane lists the programs that use unresolved variables. Select a program in the drop-down to view its source in the Source display. The Source display is similar to the Reference Report window Preview pane. For more information, see [“Preview Pane” on page 5-5](#).

Click a tab to limit the scope of the programs displayed in the pane:

- Click All to view all the programs in the project that use undefined variables.
- Click Containing variable to view only the programs that use the variable selected in the Missing Variables pane.
- Click Including copybook to view only the programs that include the copybook selected in the Copybooks pane.

Copybooks Pane

The Copybooks pane lists the missing copybooks referenced by the programs in the workspace. After copybooks are restored, it lists the variables defined in the selected copybook. Click a tab to limit the scope of the copybooks displayed in the pane:

- Click All to view all the missing copybooks in the workspace.
- Click Containing variable to view only the copybooks that contain the resolved variable selected in the Missing Variables pane.
- Click Included in program to view only the copybooks included in the program selected in the Programs pane.

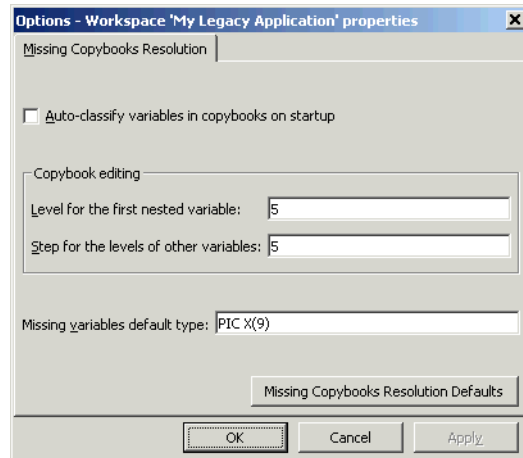
Setting Missing Copybook Resolution Options

Options let you specify whether you want to automatically generate missing copybooks when you start the Missing Copybooks Resolution tool, the nesting level of variables, and their default Cobol type.

To set Missing Copybook Resolution options:

- 1 In the **View** menu, choose **Options**. The Missing Copybooks Resolution Options window opens (Figure 8-3 on page 8-6).

Figure 8-3 *Missing Copybooks Resolution Options Window*



- 2 Select **Auto-classify variables in copybooks on startup** if you want to automatically distribute variables by copybooks when you start the Missing Copybooks Resolution tool. This option takes effect the next time you start the tool.
- 3 In the **Level for the first nested variable** field, enter the default nesting level of the first nested variable in restored copybooks. This level immediately follows the topmost level, which is always 01.
- 4 In the **Step for the levels of other variables** field, enter the nesting increment for subsequently nested variables in restored copybooks.
- 5 In the **Missing variables default type** field, enter the default Cobol type for variables defined in restored copybooks.

Tip: You can restore the default option settings by clicking **Missing Copybooks Resolution Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

Restoring Missing Copybooks

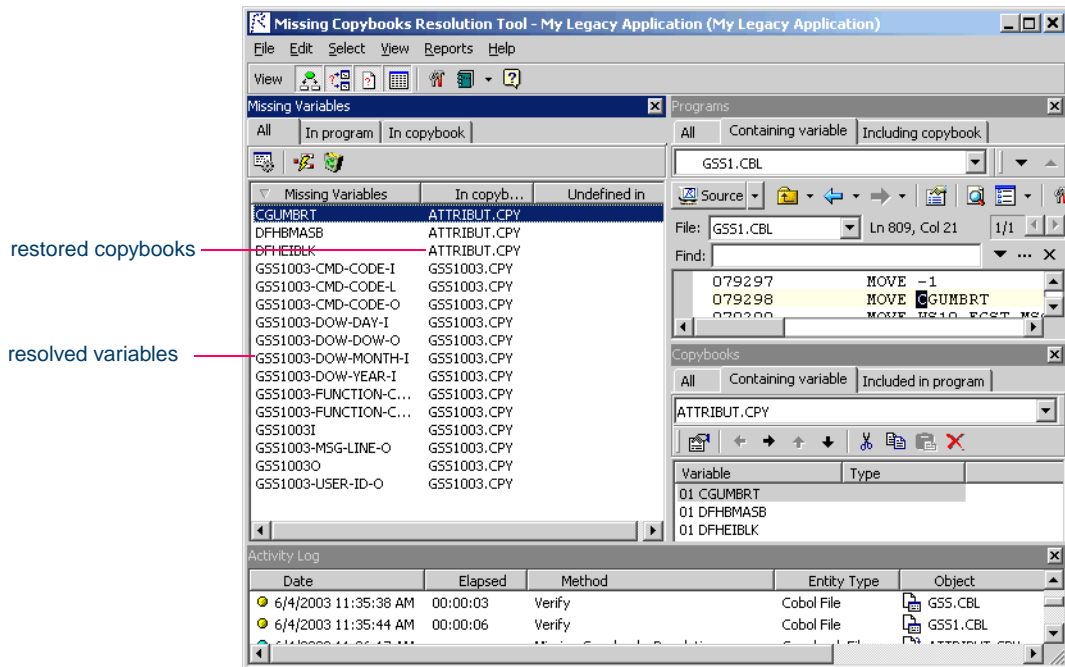
You can restore explicitly referenced copybooks automatically or manually. You must restore system copybooks manually.

To restore missing copybooks:

- 1A To restore explicitly referenced copybooks automatically, choose **Variables:Classify in Copybooks** in the **File** menu. You are prompted to confirm that you want to restore copybooks automatically. Click **Yes**. Figure 8-4 shows the Missing Copybooks report with the results for automatically restored copybooks.

Note: If a variable is used in more than one program, the tool chooses the least invasive alternative — restoring the copybook that is included in the *fewest* number of programs in the project that do *not* use the variable.

Figure 8-4 Missing Copybooks Report with Restored Copybooks



- 1B To restore an explicitly referenced copybook manually, select the unresolved variables you want to define in the copybook and choose **Put Into Copybook:Copybook** in the right-click menu.
- 1C To restore a system copybook, choose **Create System Copybook** in the **File** menu. The System Copybook dialog opens. Enter the name of the system copybook in the text field and click **OK**. Select the unresolved variables you want to define in the copybook and choose **Put Into Copybook:System Copybook** in the right-click menu.

Note: The parser will not recognize the system copybook unless you do either of the following: add a COPY statement for it to the source files that use it; or, if that is not practical, identify the system copybook in the Legacy.xml file. For assistance with the Legacy.xml file, contact support services.

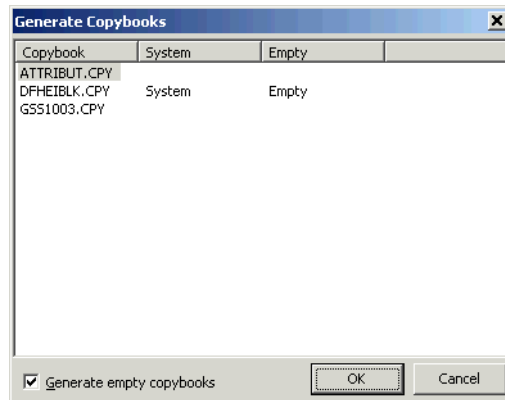
- 2 In the Copybooks pane, select the restored copybook in the drop-down list. Use the left and right arrows to nest the variables in the copybook. Use the up and down arrows to change the order of the variables in the copybook.
- 3 Specify the Cobol type of a variable by right-clicking it in the Copybooks pane and choosing **Properties** in the pop-up menu. The Variable Properties dialog opens. Enter the Cobol type in the **Variable Properties** field and click **OK**. The type is displayed in the Type column.

Tip: Use the other choices in the right-click menu to copy or move a variable to another copybook and delete a variable. To delete the definitions for all variables, choose **Variables:Undefine All** in the **File** menu. You are prompted to confirm that you want to delete all the variable definitions. Click **Yes**.

- 4 In the **File** menu, choose **Generate Copybooks**. The Generate Copybooks window opens (Figure 8-5 on page 8-9).

Tip: If you plan to export a missing copybooks report, make sure to do so *before* you generate copybooks. The report is empty after copybooks are generated.

Figure 8-5 *Generate Copybooks Window*



- 5 To generate empty copybooks as well as populated copybooks, select **Generate empty copybooks**. You might generate empty copybooks if you plan to edit copybooks manually.
- 6 Click **OK**. The copybooks are saved in the repository and registered in the project.

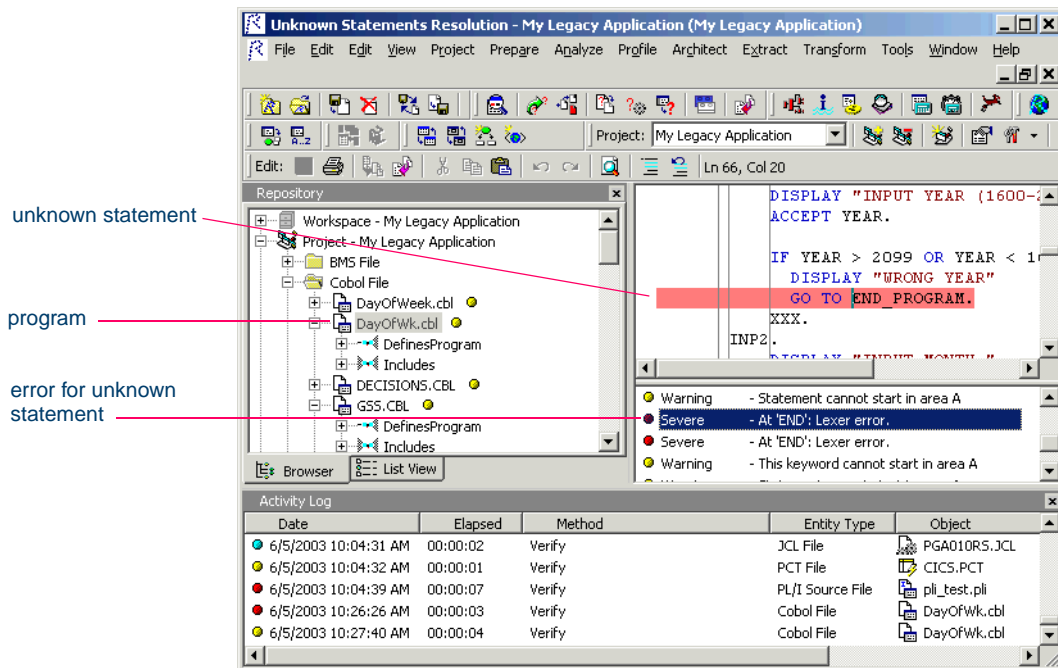
Note: If you don't complete your work in the Missing Copybooks Resolution tool, you should re-verify with the relaxed parsing option the source files that use restored copybooks. Otherwise, you will lose your work. Once you have restored *all* missing copybooks, re-verify your source files with the regular parser.

Using the Unknown Statements Resolution Tool

The parser generates errors for Cobol statements it does not recognize — *unknown statements* (Figure 8-1). If there are only a few unknown statements in your application, you can correct them by hand in the Editor.

If you have a great many unknown statements, however, you will want to re-verify your application with the relaxed parser, so that you can use the Unknown Statements Resolution tool to resolve the statements. This tool lets you group unknown statements by type, and then apply a *substitution rule* to the group to resolve the statements in a single step.

Figure 8-6 *Unknown Statement (Relaxed Parsing Results)*



Generating an Unknown Statements Report

You typically use an unknown statements report to resolve incorrect or unsupported statements. You can also use the report simply to summarize the unknown statements in the application by type.

To generate an unknown statements report:

- 1 In the Repository Browser, select the project for the unknown statements report and choose **Resolve Unknown Statements** in the **Prepare** menu. The Unknown Statements Resolution tool window opens on top of the ATW main window.

Figure 8-7 shows the report after groups have been created and the substitution rule applied.

Figure 8-7 *Unknown Statements Report (after Substitution)*

The screenshot shows the 'Unknown Statements Resolution - My Legacy Application (My Legacy Application)' window. It features a menu bar (File, View, Reports, Help) and a toolbar. The main area is divided into several panes:

- Rules:** A table with columns 'Group of statements', 'Count', and 'Action'.

Group of statements	Count	Action
(Default)	<Empty>	comment out
DFHRESP	10	substitution
IF	2	substitution
- Original Statements:** A list of statements, including 'IF MONTH > 12 OR MONTH < 1 THEN DISPLAY "WRONG MO...' and 'IF YEAR > 2099 OR YEAR < 1600 THEN DISPLAY "WRONG Y...'. A red line points from the label 'statements in selected group' to this pane.
- Modified Statement:** A pane showing the resolved code, such as 'IF YEAR > 2099 OR YEAR < 1600 THEN DISPLAY "WRONG YEAR" GO TO END-PROGRAM.'. A red line points from the label 'modified statement' to this pane.
- Activity Log:** A table at the bottom showing the process details.

Date	Elapsed	Method	Entity Type	Object
6/5/2003 10:04:31 AM	00:00:02	Verify	JCL File	PGA010R5.JCL

Red lines from the left side of the image point to the 'Rules' table (labeled 'statement groups'), the 'Original Statements' pane (labeled 'statements in selected group'), and the 'Modified Statement' pane (labeled 'modified statement').

Working with an Unknown Statements Report

The unknown statements report window consists of a Rules pane, Original Statements pane, Source pane, Modified Statement pane, and Activity Log. You can hide a pane by clicking the close box in the upper righthand corner. Select the appropriate choice in the **View** menu to show the pane again.

Rules Pane

The Rules pane lists the statement groups you have created, the number of statements in each group, and the substitution rules they use. The Default group is always displayed. You cannot delete, rename, or copy the Default group.

Sorting Entries Click a column heading in the Rules pane to sort the report entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Viewing Statements Select a group in the Rules pane to view its statements in the Original Statements pane.

Viewing Substitution Rules Right-click a group and choose **Properties** in the pop-up menu to view its substitution rules.

Renaming a Group Right-click a group and choose **Rename** in the pop-up menu to rename the group. In the text field, enter the new name of the group, then click outside the field.

Copying a Group Right-click a group and choose **Copy** in the pop-up menu to copy the group. In the text field, enter the name of the new group, then click outside the field. The new group inherits the substitution rules of the copied group.

Merging Groups Right-click a group and choose **Merge with:Group** in the pop-up menu to merge two groups. The merged group inherits the substitution rules of both groups.

Deleting a Group Right-click a group and choose **Delete** in the pop-up menu to delete the group. You are prompted to confirm the deletion. Click **Yes**.

Creating a Group Right-click in the Rules pane and choose **Create** in the pop-up menu to create a new group. In the text field, enter the name of the new group, then click outside the field. For information on how to add statements to the new group, see [“Moving Statements to Another Group”](#) below.

Original Statements Pane

The Original Statements pane lists the statements in the statement group selected in the Rules pane. Place your cursor over a statement for a moment to display a tool tip that identifies the error the parser generated for the statement.

Moving Statements to Another Group Right-click a statement and choose **Move to:Group** in the pop-up menu to move the statement to the selected group. You can select all the statements in the pane by right-clicking a statement and choosing **Select All** in the pop-up menu.

Viewing Source Select a statement to view its source in the Source and Modified Statement panes.

Sorting Entries Click the column heading in the Original Statements pane to sort the report entries by that column.

Source and Modified Statement Panes

The Source pane displays the source code for the statement selected in the Original Statements pane. The Modified Statement pane displays the modified source code for the statement selected in the Original Statements pane. All source is view-only. Source is displayed in the following colors:

- Blue if the statement has been modified.
- Red if the statement has not been modified.
- Green if the statement has been commented out.

Generating Statement Groups

You correct unknown statements in the Unknown Statements Resolution tool by classifying the statements in a group and then applying a substitution rule to the group. You can generate groups automatically or manually.

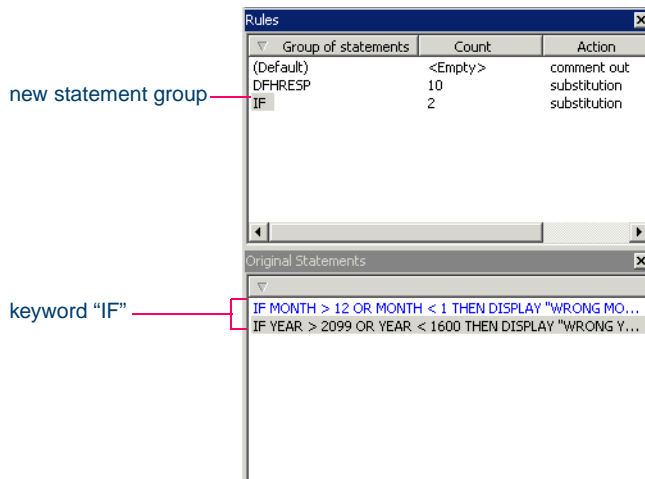
The automatic generation feature creates groups based on the first keyword it encounters in each unknown statement (Figure 8-8). If you want to specify narrower criteria for group generation, you can create groups manually.

To generate statement groups automatically:

- 1 To generate statement groups automatically, choose **Create Classification for '(Default)'** in the **File** menu. The tool generates statement groups based on the first keyword in each unknown statement. Select a group in the Rules pane to view its statements in the Original Statements pane (Figure 8-8).

Tip: You can rename groups as described in [“Rules Pane” on page 8-12](#).

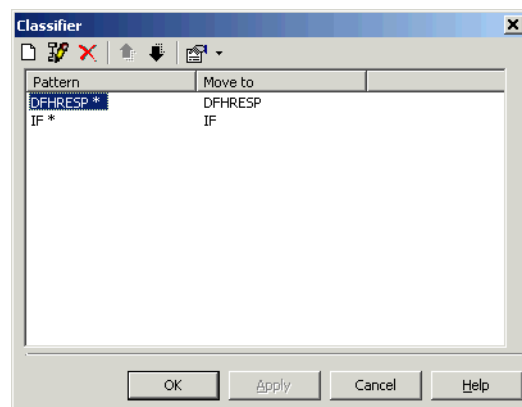
Figure 8-8 *Create Classification Results*



To generate statement groups manually:

- 1 To generate a statement group manually, create the shell for the group as described in [“Creating a Group”](#) on page 8-13.
- 2 In the **File** menu, choose **Classify**. The Classifier window opens (Figure 8-9).

Figure 8-9 *Classifier Window*



- 3 Click the button to create the pattern you want to use. In the text field, enter the text for the pattern, then click outside the field. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).
- 4 Assign the pattern to a group by selecting it and choosing the group from the drop-down.

Tip: To edit a pattern, select it and click the button. To move an entry up or down in the window, click the or buttons. To delete a pattern, select it and click the button.

- 5A Click **Apply** if you want to generate groups without dismissing the Classifier window.
- 5B Click **OK** if you want to generate groups and dismiss the Classifier window.

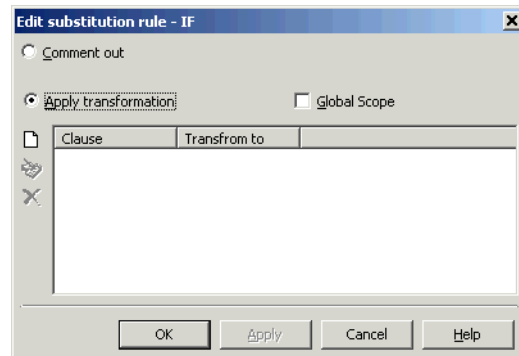
Applying Substitution Rules


Once you have generated groups, you can define the substitution rules for the statements they contain. Substitution rules determine how the Unknown Statements Resolution tool modifies unknown statements.

To apply substitution rules:

- 1 In the Rules pane, select the group whose substitution rules you want to define and choose **Group Properties** in the **File** menu. The Edit Substitution Rule window opens (Figure 8-10).

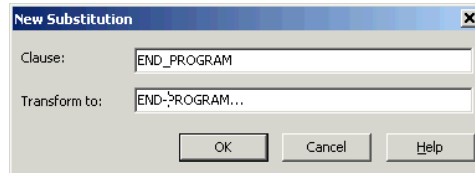
Figure 8-10 *Edit Substitution Rule Window*



- 2 Select either of the following:
 - **Comment out** if you want to comment out the unknown statements in the group.
 - **Apply transformation** if you want to modify the unknown statements in the group. Select **Global Scope** if you want to apply the substitution rule not only to the unknown statements in the group, but to the entire source file that contains an unknown statement.
- 3 If you selected **Apply transformation** in [step 2](#), click the  button to define the substitution rule. The New Substitution Dialog opens (Figure 8-11 on page 8-17).
- 4 In the **Clause** field, enter the text of the keyword you want to modify. In the **Transform to** field, enter the new text for the keyword. A blank means delete the keyword. If you want to preserve the rest of

the clause after the keyword that is to be replaced, insert an ellipsis (...) after the replacement keyword. Click **OK**.

Figure 8-11 *Edit Substitution Dialog*



- 5A Click **Apply** in the Edit Substitution Rule window if you want to define the rule without dismissing the window.
 - 5B Click **OK** in the Edit Substitution Rule window if you want to define the rule and dismiss the window. The rule is applied.
 - 6 If you want to specify a comment or marker the tool will include in application source when it inserts the modified statements, choose **Options** in the **View** menu. The Unknown Statements Resolution Options window opens. Select **Comment for**, then enter your comment or marker for:
 - Modified statements in the **Transformed statements** field.
 - Commented out statements in the **Commented statements** field.
- Note:** The comment or marker may not contain more than six single-byte or two double-byte characters. It's best not to enter double-byte characters.
- 7 If you want to be able to return to your classification and substitution rules the next time you open the Unknown Statements Resolution tool, choose **Save Classification and Substitution Rules** in the **File** menu.
 - 8 In the **File** menu, choose **Apply Changes** to insert the modified statements in application source. You are prompted to confirm that you want to insert the modified statements. Click **OK**.
- Note:** You can simply comment out all unknown statement by choosing **Comment Out All Unrecognized Statements** in the **File** menu.

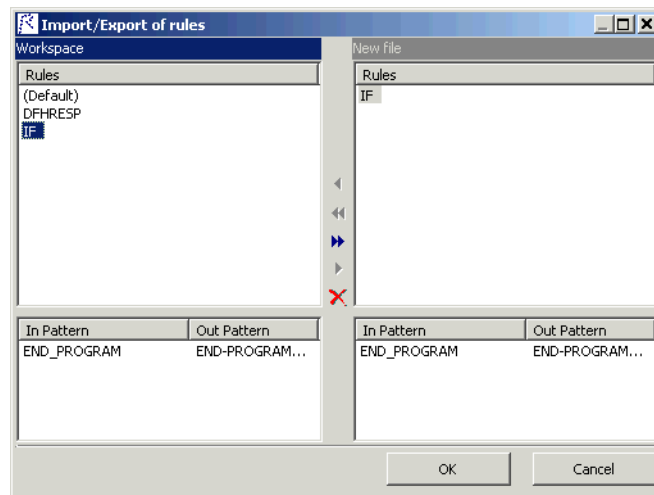
Exporting and Importing Substitution Rules

You can export substitution rules to a file, so that they will be available to import the next time you open the Unknown Statements Resolution tool.

To export substitution rules:

- 1 In the File menu, choose **Import/Export of Rules**. The Import/Export of Rules window opens (Figure 8-12).

Figure 8-12 *Import/ Export of Rules Window*



- 2 In the Workspace pane, select the rule you want to export and click the **▶** button. The rule is displayed in the New File pane.

Tip: To move all the rules in the Workspace pane to the New File pane, click the **▶▶** button. To delete a rule from the New File pane, select it and click the **✖** button.

- 3 When you are satisfied with the list of rules in the New File pane, click **OK**. A Save As dialog opens, where you can specify the name and folder for the export file.

To import substitution rules:

- 1 In the File menu, choose **Import/Export of Rules**. The Import/Export of Rules window opens (Figure 8-12 on page 8-18).
- 2 Right-click in the New Files pane and choose **Open** in the pop-up menu. An Open dialog appears, where you can select the file you want to import.

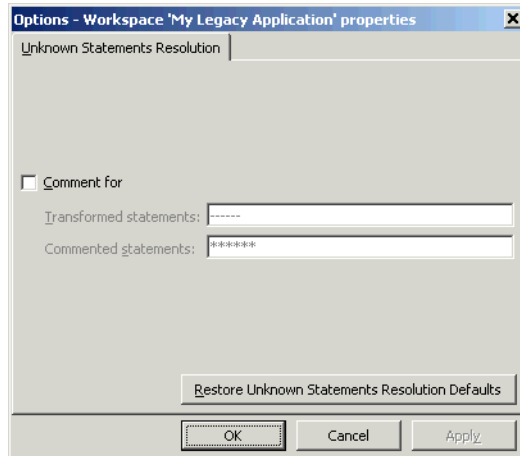
Setting Unknown Statements Resolution Options

Options let you specify a comment or marker that the Unknown Statements Resolution tool will include in application source when it inserts modified statements.

To set Unknown Statements Resolution options:

- 1 In the **View** menu, choose **Options**. The Unknown Statements Resolution Options window opens (Figure 8-13).

Figure 8-13 *Unknown Statements Resolution Options Window*



- 2 Select **Comment for**, then enter your comment or marker for:
 - Modified statements in the **Transformed statements** field.
 - Commented out statements in the **Commented statements** field.

Note: The comment or marker may not contain more than six single-byte or two double-byte characters. It's best not to enter double-byte characters.

Tip: You can restore the default option settings by clicking **Unknown Statement Resolution Defaults**, then choosing **Restore Defaults** in the drop-down menu. Choose **Save To** in the drop-down menu to save the option settings to a file. Choose **Load From** in the menu to restore the option settings from a file.

Exporting Relaxed Parsing Tool Reports

Choose the appropriate option in the **Reports** menu to export missing copybooks or unknown statements reports. A report window opens. Choose **Save** to export the report to HTML, Excel, RTF, Word, or formatted text. You can also print the report.

Tip: If you plan to export a missing copybooks report, make sure to do so *before* you generate copybooks. The report is empty after copybooks are generated.

What's Next?

Now that you have resolved undefined variables in missing copybooks and corrected unknown statements in application source, you are ready to re-verify the application with the regular parser. After you have successfully completed verification, you can use Application Partitioner to restructure your workspace into more meaningful projects, as described in the next chapter.

Partitioning Applications



The *Application Partitioner* identifies legacy subsystems and partitions them into self-contained projects based on an analysis of repository contents. The projects can serve as the basis for coarse-grained components you create during the legacy transformation stage.

Understanding Application Partitioning

Suppose you are analyzing a large system comprised of three subsystems, GSS1, GSS2, and GSS3, and that the system is organized in a single project in the ATW repository. Your goal is to restructure the system into three projects named GSS1, GSS2, and GSS3.

You could restructure the workspace manually, of course, as described in [Chapter 2, “Setting Up a Workspace and Projects.”](#) If the system is very large, however, it will be more convenient to use Application Partitioner to restructure the workspace automatically. You will still have to

do some manual reorganization after the automatic restructuring, but much less than you would have to do without the tool.

Partitioning Algorithms

Application Partitioner uses two partitioning algorithms to determine how to assign source files to projects:

- *Name-based partitioning* assigns source files to projects based on text matching of source file names with the patterns you specify.
- *Relationship-based partitioning* assigns source files to projects based on the extent to which the source files are related. If two source files reference the same copybook, for example, they can be regarded as “tightly related,” at least as compared with source files that do not reference the same copybook (Figure 9-1 on page 9-3).

Relationship-based partitioning produces the kinds of results that would satisfy our goal of restructuring a very large system in terms of its subsystems. One would expect that the objects in GSS1, for example, are more closely related to each other than they are to the objects in GSS2 and GSS3. Relationship-based partitioning would see to it that these objects are automatically assigned to the appropriate project.

Relationship Weights

The *weight* accorded to a relationship determines the importance of that relationship in calculating the connection between source files in relationship-based partitioning. The weight you assign to the relationship between source files and data stores, for example, might be greater than the weight you assign to the relationship between source files and copybooks.

Drivers and Utilities

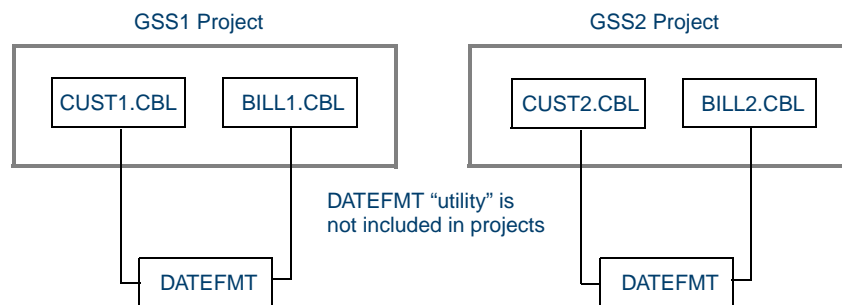
A *driver* is a source file that references many source files — a startup program, for example. A *utility* is a source file that is referenced by many source files — DATEFMT, for example.

Together, the relationships of drivers and utilities to other source files effectively close a system. These relationships to otherwise unrelated

files obscure the extent to which subsystems are distinct, and for that reason, make partitioning difficult.

Application Partitioner lets you temporarily remove drivers and utilities from projects before relationship-based partitioning takes place. Once you have an accurate partitioning result, you can restore the files to projects as appropriate.

Figure 9-1 *Relationship-Based Partitioning*



Relationship-based partitioning assigns closely related application files to the same project.

Seed Objects

You can use *seed objects* to customize the partitioning algorithms. Suppose you use name-based partitioning with a pattern like `GSS1*.*` — but you do not want one file, `GSS1X.CBL`, to be included in the project. In that case, you could make the file a seed object and assign it to a different project before performing name-based partitioning. That tells Application Partitioner to ignore the file when it executes the algorithm.

Evaluating Partitioning Results

Ordinarily, you would prefer that each project you set up be self-contained — that it consist of highly related objects that are unrelated to objects in other projects. Application Partitioner provides two tools for evaluating these measures:

- The *Interproject Diagram* (Figure 9-5 on page 9-8) shows the relationships between projects in diagrammatic form.
- *Clustering Quality* graphs (Figure 9-6 on page 9-9) show project size and relationships. The relationships graph is especially useful in measuring internal and external relationships.

Getting Started in Application Partitioner

Application Partitioner is a complex tool that takes some getting used to. This section describes a sample use of the tool that should help you get oriented in it. Assume that your workspace consists of a single project named GSS. You want to partition the workspace into projects named GSS1, GSS2, and GSS3.

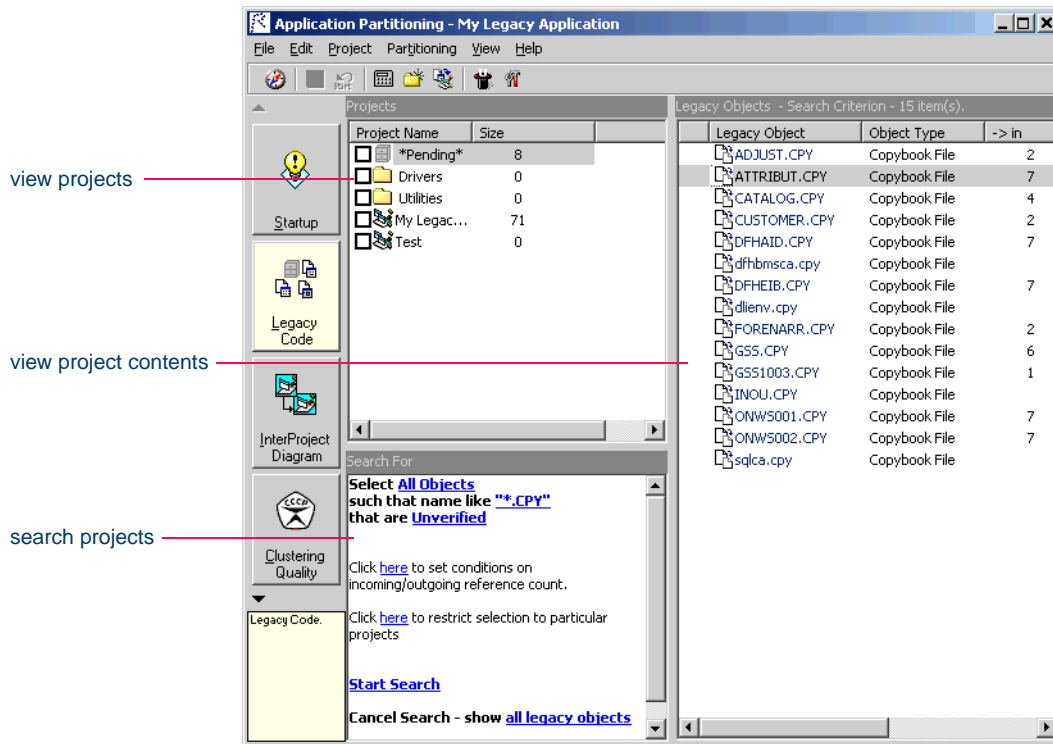
To partition a workspace:

- 1 In the Repository Browser, select the GSS project and choose **Application Partitioning** in the **Prepare** menu. The Application Partitioner window opens, with the Startup view selected. Click the **Legacy Code** button on the left side of the window. Application Partitioner dismisses the Startup view and displays the Legacy Code view (Figure 9-2 on page 9-5).
- 2 The Projects pane lists the projects in your repository. Double-click the GSS project to view its contents in the Project Contents pane.
- 3 To search for copybooks to designate as utilities, set the following parameters in the Search For pane:
 - All Objects
 - such that name like “*.CPY”
 - that are unverified

Note: For background on utilities, see [“Drivers and Utilities” on page 9-2](#).

- 4 Click the **Start Search** link. Application Partitioner displays the results of the search in the Legacy Objects pane. The Legacy Objects pane shows every copybook in the GSS project.

Figure 9-2 Application Partitioner Window — Legacy Code View



- 5 To designate copybooks as utilities, choose **Select All** in the **Edit** menu, then hold down the CTRL key and drag-and-drop the copybooks to the Utilities folder in the Projects pane. The Size column for the Utilities folder should reflect the move.
- 6 To create the new projects GSS1, GSS2, and GSS3, choose **New Project** in the **Project** menu. In the Project dialog, enter GSS1 in the text field and click **OK**. Repeat this step for GSS2 and GSS3.
- 7 To remove the original GSS project, select it in the Projects pane and choose **Delete Project** in the **Project** menu. You are prompted to confirm the deletion. Click **OK**. Application Partitioner moves the contents of GSS to the Pending project in the Projects pane. The Size column for the Pending project should reflect the move.

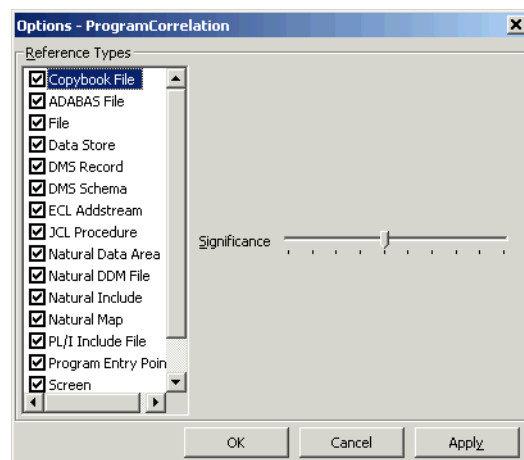
- 8 To designate seed objects, double-click the Pending project to view its contents in the Legacy Objects pane. Select a file and choose **Mark as Seed** in the **Edit** menu. Hold down the CTRL key and drag-and-drop the file to the GSS1 project in the Projects pane. The Size column for the GSS1 project should reflect the move. Repeat this step for GSS2 and GSS3.

Note: For background on seed objects, see [“Seed Objects” on page 9-3](#).

- 9 To change the default relationship weights, choose **Options** in the **View** menu. The Options window opens (Figure 9-3). Select a file type and use the Significance slider on the right side of the window to change the relationship weight for the file type. Deselect a file type if you do not want its relationships to be used in the partitioning calculation.

Note: For background on relationship weights, see [“Relationship Weights” on page 9-2](#).

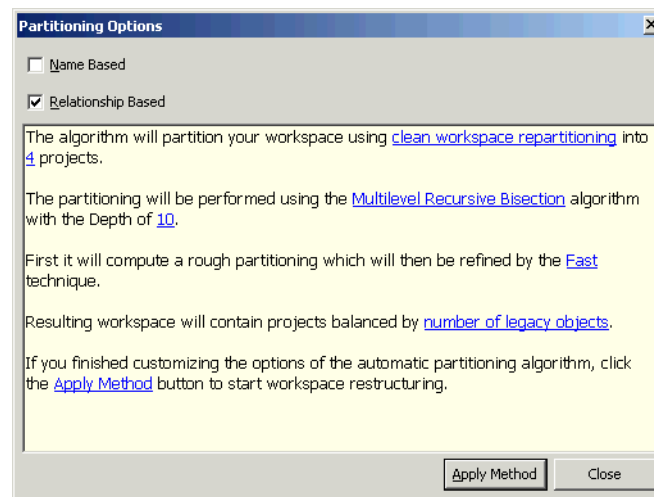
Figure 9-3 Options Window



- To perform relationship-based partitioning, choose **Select Partitioning Method** in the **Partitioning** menu. The Partitioning Options window opens (Figure 9-4).

Note: For background on partitioning algorithms, see [“Partitioning Algorithms”](#) on page 9-2.

Figure 9-4 *Partitioning Options Window — Relationship-Based*

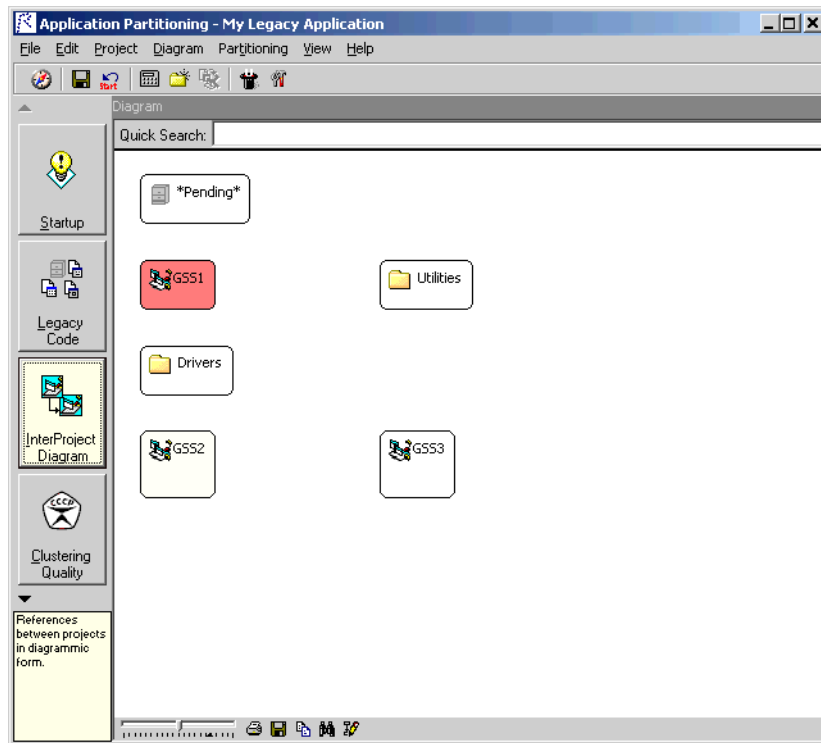


- In the Partitioning Options window, select the **Relationship-Based** check box, then set the following parameters:
 - current partitioning as initial**
 - Multilevel Recursive Bisection**
 - Fast technique**
 - number of legacy objects**
- Click **Apply Method** to perform relationship-based partitioning. The Size column for the GSS1, GSS2, and GSS3 projects should reflect the partitioning results. Double-click a project to view its contents.
- To restore the copybooks to the new projects, select the Utilities folder and choose **Reallocate** in the **Project** menu. Copybooks are

moved to the projects with which they have the strongest relationships.

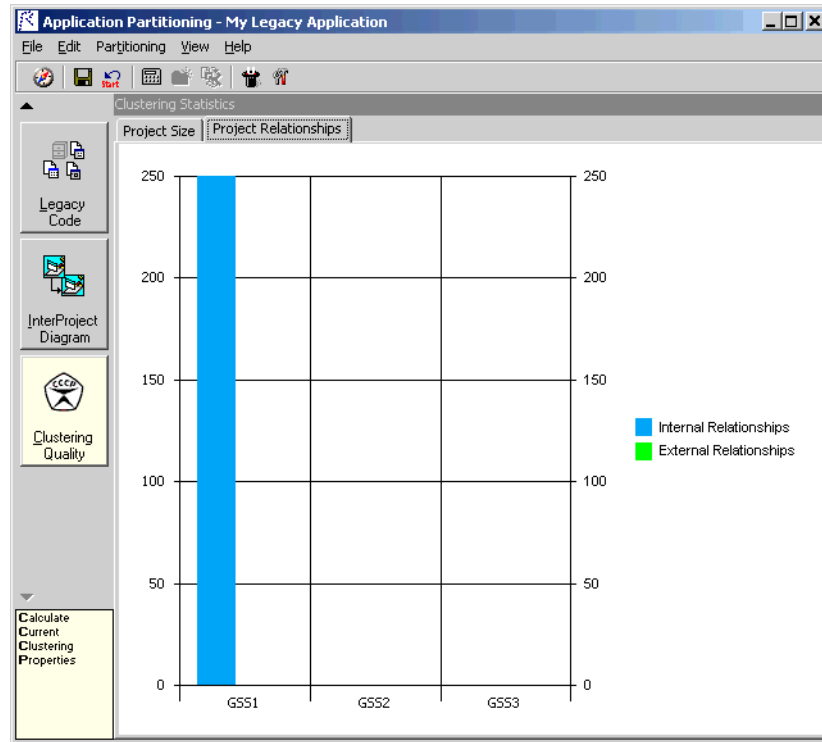
- 14 To view the interproject diagram for the partitioned workspace, click the **InterProject Diagram** button on the left side of the Application Partitioner window. The Interproject Diagram window opens (Figure 9-5).

Figure 9-5 *Interproject Diagram Window*



- 15 To view the clustering quality graphs for the partitioned workspace, click the **Clustering Quality** button on the left side of the Application Partitioner window. The Clustering Statistics window opens. Click the Project Relationships tab to view a graph of the relationship statistics for a project (Figure 9-6 on page 9-9).

Figure 9-6 Clustering Statistics Window — Project Relationships Graph



- 16 To commit the new partitioning to the repository, choose **Commit changes** in the **File** menu. Application Partitioner commits the changes to the repository.

Using Application Partitioner

Partitioning is an iterative process in which you work closely with inter-project diagrams and clustering quality graphs to achieve the distribution of objects you want. Application Partitioner provides a variety of tools you can use to achieve the right result for your application.

Working with Partitioning Information

The Application Partitioner window consists of a Projects pane, Search pane, and Project Contents pane. The panes are always displayed.

Projects Pane

The Projects pane displays the projects in the workspace and the number of application files they contain. Double-click a project to display its contents in the Project Contents pane.

The Pending project contains the system files that ATW provides for any applications that reference them. It's also the temporary location for the contents of projects you delete. You can move objects manually into the Pending project.

The Drivers folder contains the application files you designate as drivers. The Utilities folder contains the application files you designate as utilities. For background on drivers and utilities, see [“Drivers and Utilities” on page 9-2](#).


Sorting Entries Click a column heading in the Projects pane to sort the project entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Creating Projects Choose **New Project** in the **Project** menu to create a new project. In the Project dialog, enter the name of the new project in the text field and click **OK**.

Renaming Projects Select a project and choose **Rename Project** in the **Project** menu to rename the project. In the Project dialog, enter the new name of the project in the text field and click **OK**.

Deleting Projects Select a project and choose **Delete Project** in the **Project** menu to delete the project. You are prompted to confirm the deletion. Click **OK**. Application Partitioner moves the contents of the project to the Pending project.

Locking Projects Select a project and choose **Locked** in the **Project** menu to lock the project. The icon for the project changes to a  and a check mark appears beside the **Locked** choice in the **Project** menu. Select **Locked** again to unlock the project. Application Partitioner ignores locked projects when it partitions a workspace or allocates projects.

Search Pane

The Search pane lets you search for application files in the repository. Use the links in the Search pane to specify search parameters. You can search by any combination of parameters.

To start the search, click the **Start Search** link. Search results are displayed in the Project Contents pane.

Note: You must set the seed category and verification status parameters.

Setting the Seed Category Parameter Click the Seed category link to specify the seed category parameter. In the Seed Category window, select the seed category.

Setting the Verification Status Parameter Click the Verification Status link to specify the verification status parameter. In the Verification Status window, select any combination of verification statuses.

Setting the Object Name Parameter Click the object name link to specify the object name parameter. In the Name Pattern window, select:

- **Name Like** to specify a pattern that is like the name of the file you are searching for, then enter the pattern in the **Pattern** field.
- **Name Not Like** to specify a pattern that is unlike the name of the file you are searching for, then enter the pattern in the **Pattern** field.

You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

Setting the Incoming/Outgoing Reference Count Parameter Click the incoming/outgoing reference count link to specify the reference count parameter. In the Reference Count window, select:

- **Is Referenced At Least** to specify that the file is referenced at least the number of times you indicate in the adjoining combo box.
- **Is Referenced At Most** to specify that the file is referenced at most the number of times you indicate in the adjoining combo box.
- **Refers To At Least** to specify that the file references at least the number of files you indicate in the adjoining combo box.
- **Refers To At Most** to specify that the file references at most the number of files you indicate in the adjoining combo box.

Tip: This parameter is useful when you are searching for candidates to designate drivers or utilities:

- For drivers, select **Refers To At Least** and enter a value appropriate for the application. Then select **Is Referenced At Most** and enter 0.
- For utilities, select **Is Referenced At Least** and enter a value appropriate for the application. Then select **Refers To At Most** and enter 0.

For background on drivers and utilities, see [“Drivers and Utilities”](#) on page 9-2.

Setting the Projects Parameter Click the projects link to specify the projects parameter. In the Projects window, select the relationship of the files you are searching for to workspace projects. You can also specify whether the files you are searching for are shared by projects.

For the following options, select projects by placing a check mark next to them in the Projects pane:

- **Belong to Some of the Selected Project(s) and Belong to Each of the Selected Project(s)**
- **Do Not Belong to Some of the Selected Project(s) and Do Not Belong to Each of the Selected Project(s)**

Projects Contents Pane

The Project Contents pane displays the contents of the project selected in the Projects pane. After a search, it displays the results of the search.

The Project Contents pane contains five columns:

- The Legacy Object column lists the files in the selected project, or the files returned by a search. Verified files are displayed in black. Unverified files are displayed in blue.
- The Object Type column displays the file type.
- The In column displays the number of objects that reference the file.
- The Out column displays the number of objects the file references.

Sorting Entries Click a column heading in the Project Contents pane to sort the entries by that column.

Sizing Columns Grab-and-drag the border of a column heading to increase or decrease the width of the column.

Displaying Related Files To list objects related to a file, select the file in the Project Contents pane and choose **Show Related** in the **Edit** menu. You can view objects that reference the file, objects the file references, or both. To return to the original display, choose **Backtrack** in the **Edit** menu.

Assigning Files to Projects To move a file to a project, select the file in the Project Contents pane. Hold down the CTRL key and drag-and-drop the file to the new project in the Projects pane. To copy a file to a project, select the file in the Project Contents pane and drag-and-drop the file to the new project in the Projects pane.

Allocating Files to Projects To allocate the files in a project or folder to other projects, select the project or folder and choose **Reallocate** in the **Project** menu. Files are moved to the unlocked projects with which they have the strongest relationships. Files with no relationships to other projects are not moved. Relationship weights are taken into account in performing the calculation. For background on relationship weights, see [“Relationship Weights” on page 9-2](#).

Designating Seed Objects To designate a file as a seed object, select the file and choose **Mark as Seed** in the **Edit** menu. An @ symbol appears beside the object in the Project Contents pane. To remove a seed object designation, select the file and choose **Unmark** in the **Edit** menu. For background on seed objects, see [“Seed Objects” on page 9-3](#).

Creating and Restoring Drivers and Utilities

A *driver* is a source file that references many source files — a startup program, for example. A *utility* is a source file that is referenced by many source files — DATEFMT, for example. Application Partitioner lets you temporarily remove drivers and utilities from projects before relationship-based partitioning takes place. For background on drivers and utilities, see [“Drivers and Utilities” on page 9-2](#).

Designating Files as Drivers or Utilities

You designate files as drivers by selecting them and assigning them to the Drivers folder in the Projects pane. You designate files as utilities by selecting them and assigning them to the Utilities folder in the Projects pane. For more information on assigning files, see [“Assigning Files to Projects” on page 9-13](#).

Restoring Drivers and Utilities to Projects

You can restore drivers and utilities to projects manually, as described in [“Assigning Files to Projects” on page 9-13](#). You can restore drivers and utilities to projects automatically in either of two ways:

- To *fill up* a project with referenced or referencing files in the Drivers, Utilities, and Pending folders, select the project and choose **Fill Up** in the **Project** menu. Pending files are moved to the project, drivers and utilities are copied to the project. You may have to repeat this action to pick up files that reference or are referenced by files detected in earlier invocations.
- To *allocate* drivers and utilities to other projects, select the Drivers or Utilities folder and choose **Reallocate** in the **Project** menu. Drivers and utilities are moved to the unlocked projects with which they have the strongest relationships. Drivers and utilities with no relationships to other projects are not moved. Relationship weights are

taken into account in performing the calculation. For more information, see [“Editing Relationship Weights” on page 9-15](#).

Editing Relationship Weights

The *weight* accorded to a relationship determines the importance of that relationship in calculating the connection between source files in relationship-based partitioning. The weight you assign to the relationship between source files and data stores, for example, might be greater than the weight you assign to the relationship between source files and copybooks.

To edit relationship weights:

- 1** In the **View** menu, choose **Options**. The Options window opens (Figure 9-3 on page 9-6).
- 2** Select a file type in the Reference Types pane and use the Significance slider on the right side of the window to change the relationship weight for the file type. Deselect a file type if you do not want its relationships to be used in the partitioning calculation.
- 3** Click **OK** to save your changes and dismiss the window.

Executing the Partitioning Methods

Application Partitioner uses two partitioning methods to determine how to assign source files to projects:

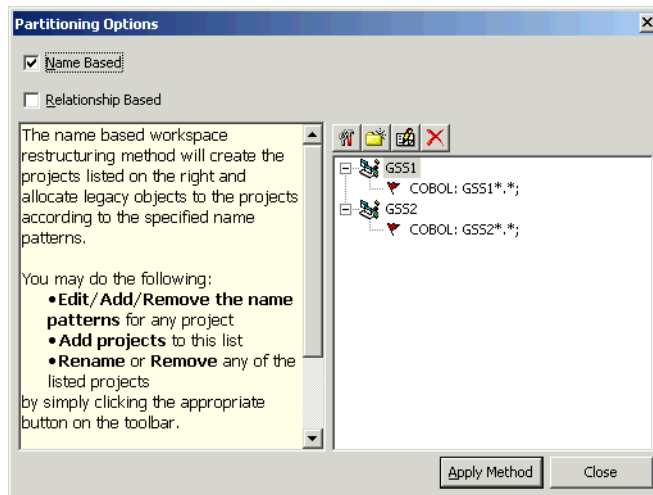
- *Name-based partitioning* assigns source files to projects based on text matching of source file names with the patterns you specify.
- *Relationship-based partitioning* assigns source files to projects based on the extent to which the source files are related. If two source files reference the same copybook, for example, they can be regarded as “tightly related,” at least as compared with source files that do not reference the same copybook (Figure 9-1 on page 9-3).


Tip: Relationship-based partitioning is a heuristic, randomized method that approximates the result that best minimizes inter-project dependencies. In all likelihood, you will have to manually adjust projects after performing the method.


To perform name-based partitioning:

- 1 In the **Partitioning** menu, choose **Select Partitioning Method**. The Partitioning Options window opens. In the Partitioning Options window, select the **Name-Based** check box. The window changes to reflect your selection (Figure 9-7).

Figure 9-7 Partitioning Options Window — Name-Based



- 2 To specify the project you want to assign source files to, click the  button. The Project dialog opens. Enter the name of the project in the text field and click **OK**.

Tip: To rename a project, select the project and click the  button. Application Partitioner creates an edit field for the project. Enter the new name in the edit field and click outside the field.


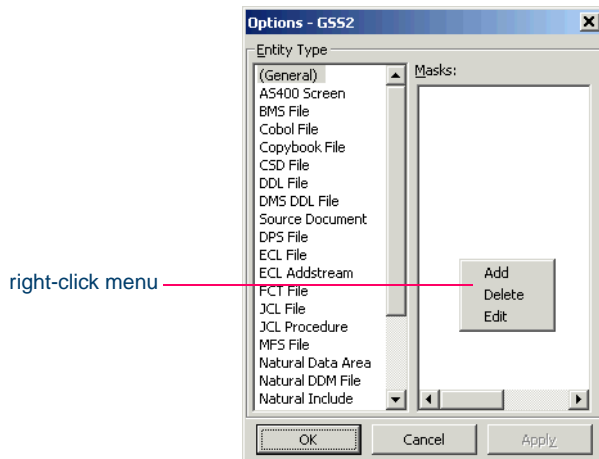
- 3 To assign a pattern to a project, select the project and click the  button. The Options window opens (Figure 9-8 on page 9-17).

Figure 9-8 *Options Window*




- 4 In the Entity Type pane, select the file type of the files you want to match. The Masks pane lists the patterns assigned to the selected file type. Right-click in the Masks pane and choose **Add** from the pop-up menu to add a pattern to the list. The system displays an empty text field next to a selected check box. Enter the name of the new pattern in the field and click outside the field. You can use wildcard patterns allowed in LIKE statements by Visual Basic for Applications (VBA).

Tip: You can edit a pattern by right-clicking it and choosing **Edit** from the pop-up menu. You can delete a pattern by right-clicking it and choosing **Delete** from the pop-up menu. Deselect an extension if you want the partitioning method to ignore it.

Click **Apply** if you want to save your settings without dismissing the Options window. Click **OK** if you want to save your settings and dismiss the Options window.

- 5 Repeat steps 2-4 for each project you want to assign source files to.
- 6 Click **Apply Method** to run the partitioning method.

To perform relationship-based partitioning:

- 1 In the **Partitioning** menu, choose **Select Partitioning Method**. The Partitioning Options window opens. In the Partitioning Options window, select the **Relationship-Based** check box. The window changes to reflect your selection (Figure 9-4 on page 9-7).
- 2 To specify the project you want to assign source files to, click the  button. The Project dialog opens. Enter the name of the project in the text field and click **OK**.
- 3 Choose either:
 - **current partitioning as initial** if you want the method to be applied to your current set of projects.
 - **clean workspace repartitioning** if you want the method to create a new set of projects. Specify the number of projects in the projects link.
- 4 Choose either:
 - **Multilevel Recursive Bisection** if you want the method to recursively partition the workspace into projects using an initial phase of object-grouping iterations. Specify the number of iterations in the depth link. Generally, the higher the number, the better the clustering quality and poorer the method performance.
 - **Single-Level Recursive Bisection** if you do not want the method to recursively partition the workspace into projects using an initial phase of object-grouping iterations. Choosing this option will improve performance but degrade clustering quality.
- 5 Choose either:
 - **Thorough** if you want to enhance thoroughness at the expense of performance.
 - **Fast** if you want to enhance performance at the expense of thoroughness.
- 6 Click **Apply Method** to run the partitioning method.

Viewing Diagrams and Graphs

Use interproject diagrams and clustering quality graphs to evaluate the dependencies between projects. To view the diagrams and graphs, follow steps [14-15](#) on page 9-8.

Interproject Diagram

Interproject diagrams (Figure 9-5 on page 9-8) show the relationships between projects in diagrammatic form. Projects or relationships displayed in red have the greatest relationship weight, projects or relationships displayed in black have the least relationship weight. Projects or relationships with weights in between are displayed in an appropriate combination of red and black. You can work with diagrams in much the same way you work with other ATW diagrams. For diagrammer usage, see *Analyzing Projects* in the ATW document set.

You can export an interproject diagram to a wide variety of standard formats. For more information, see *Getting Started* in the ATW document set.

Clustering Quality Graphs

Clustering quality graphs (Figure 9-6 on page 9-9) show project size and relationships. The relationships graph is especially useful in measuring internal and external relationships.

Saving and Restoring Sessions

If you need to come back to Application Partitioner to finish your work, you can save your session in its current state and restore it when you are ready to start work again.

Saving Sessions Choose **Save Session** in the **File** menu to save the current Application Partitioner session. A Save Session dialog opens, where you can specify the name and folder for the session file.

Loading Sessions Choose **Load Session** in the **File** menu to load a saved session. A Load Session dialog opens, where you can select the session file you want to load.

Committing or Rolling Back Your Work

When you are satisfied with your partitioning results, you can commit the results to the repository. You can roll back any changes you make since your last commit.

Committing Partitioning Results Choose **Commit changes** in the **File** menu to commit your changes to the repository.

Rolling Back Partitioning Results Choose **Rollback** in the **File** menu to roll back changes you have made since your last commit. You are prompted to confirm the rollback. Click **OK**.

What's Next?

You should now have a completely verified and appropriately structured workspace. That means you're ready to start analyzing your application, as described in *Analyzing Projects* in the ATW documentation set.

Using the Batch Refresh Feature



The ATW *batch refresh* feature lets you register and verify source files in batch mode. Other utilities packaged with the feature let you analyze application complexity, generate HTML views of workspace repositories, and, if you are licensed to use the Application Architect product, perform Dead Code Elimination (DCE).

Understanding the Batch Refresh Feature

Figure A-1 on page A-2 shows the batch refresh process. The Source Distribution process (Distrib.exe) transfers legacy source files downloaded from the mainframe to a location you specify in the X-Ref database — the *staging area* for a workspace. If a source file is not used by a workspace, Source Distribution transfers it to a holding directory for unallocated files.

Note: Source files must be in ASCII format and have recognized file extensions, as described in [step 2 on page 2-7](#). The extensions recognized for each workspace may differ.

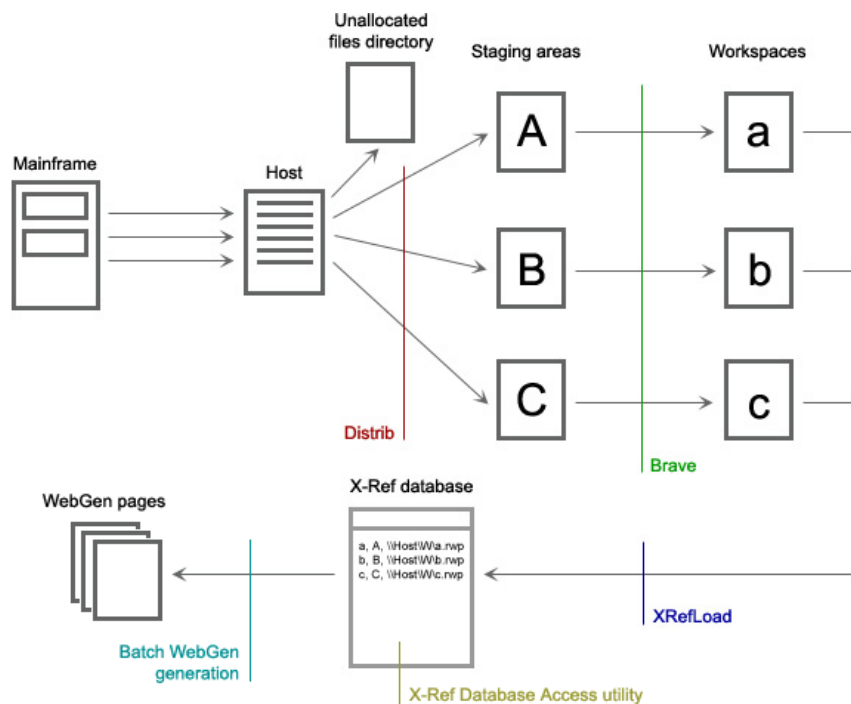
A-2 Using the Batch Refresh Feature
Understanding the Batch Refresh Feature

The X-Ref database cross-references the files in a workspace with their staging areas. The database consists of two tables:

- The Location table identifies the names, workbench addresses, and staging areas of workspaces. *You must populate this table manually.*
- The Objects table identifies the source files in workspaces. This table is automatically populated by the X-Ref Table Load process (XRefLoad.exe).

If a source file in the staging area differs from the version of the source file maintained by the system, the Batch Registration and Verification process (Brave.exe) overwrites the system version of the source file and verifies the new version. It also registers and verifies source files added to staging areas manually, and optionally performs Dead Code Elimination. The Batch WebGen Generation process (BWG.exe) generates HTML views of workspace repositories.

Figure A-1 *Batch Refresh Process*



How the Batch Refresh Feature Is Packaged

The batch refresh package consists of:

- *workbench home\Bin\XrefLoad.exe* — contains the X-Ref Table Load utility.
- *workbench home\Bin\Plexity.exe* — contains the Complexity Statistics Load utility.
- *workbench home\Bin\XrefAccess.exe* — contains the X-Ref Database Access utility.
- *workbench home\Bin\Distrib.exe* — contains the Source Distribution utility.
- *workbench home\Bin\Brave.exe* — contains the Batch Registration and Verification utility.
- *workbench home\Scripts\BRP\Refresh.bj* — contains the script file for Brave.exe.
- *workbench home\Bin\BWG.exe* — contains the Batch WebGen Generation utility.
- *workbench home\Templates\X-Ref.mdb* — contains the MS Access 2000 template for the X-Ref database.
- *workbench home\Templates\X-Ref97.mdb* — contains the MS Access 97 template for the X-Ref database.

Running the Batch Refresh Feature

Execute the commands described in this section in the following order:

- XrefLoad.exe
- Plexity.exe
- XrefAccess.exe
- Distrib.exe
- Brave.exe
- BWG.exe

Important: Rerun XrefLoad and Plexity after running Brave to populate the X-Ref database with changes to the repository. The administrator may have manually distributed unallocated files to staging areas after running Distrib, for example. By running XRefLoad again, you ensure that the database is populated with the new files.

X-Ref Table Load

Use the X-Ref Table Load tool to populate the X-Ref database. To execute X-Ref Table Load, enter the following command:

```
XRefLoad.exe X-RefDatabase Workspace ReportFolder
```

where:

- *X-RefDatabase* is the fully qualified pathname of the X-Ref database.
- *Workspace* is the workspace name in the X-Ref database. Use an asterisk (*) wildcard to specify all the workspaces in the database.
- *ReportFolder* is the folder for database load reports.

You can expect the following files as output:

- XRef.Notification describes the completion status of the database load process.
- XRef.Report shows the number of files added to workspaces and the number of files removed from workspaces.

Important: The Location table in the X-Ref database cross-references workspaces with their network addresses and staging areas. You must populate this table manually before running X-Ref Table Load.

Complexity Statistics Load

Use the Complexity Statistics Load tool to populate the X-Ref database with complexity indices and values. To execute Complexity Statistics Load, enter the following command:

```
Plexity.exe X-RefDatabase Workspace ReportFolder
```

where:

- *X-RefDatabase* is the fully qualified pathname of the X-Ref database.
- *Workspace* is the workspace name in the X-Ref database. Use an asterisk (*) wildcard to specify all the workspaces in the database.
- *ReportFolder* is the folder for database load reports.

You can expect the following files as output:

- *Plexity.Notification* describes the completion status of the complexity statistics load process.
- *Plexity.Report* describes the complexity indices and values added to the database.

X-Ref Database Access

Use the X-Ref Database Access tool to edit the X-Ref database manually. The tool opens a GUI window where you can add, modify, or delete cross-references to workspaces in the X-Ref database and delete cross-references to objects. You can also create standard reports for the Location and Objects tables.

Usage is similar to that for other report tools in ATW. To execute X-Ref Database Access, enter the following command:

```
XRefLoad.exe X-RefDatabase
```

where:

- *X-RefDatabase* is the fully qualified pathname of the X-Ref database.

Source Distribution

Use the Source Distribution tool to transfer downloaded source files to staging areas. To execute Source Distribution, enter the following command:

```
Distrib.exe XRefDatabase SourceFolder  
UnallocatedFolder ReportFolder
```

where:

- *XRefDatabase* is the fully qualified pathname of the X-Ref database.
- *SourceFolder* is the folder that contains the source files downloaded from the mainframe.
- *UnallocatedFolder* is the folder for unallocated files.
- *ReportFolder* is the folder for source distribution reports.

You can expect the following files as output:

- *Distrib.Notification* describes the completion status of the Source Distribution process.
- *Summary.Report* summarizes the distribution of files to workspaces.
- *Unallocated.Report* lists source files that were not allocated.

Batch Registration and Verification

Use the Batch Registration and Verification tool to register and verify downloaded source files. Run the tool again *after* registration and verification to perform Dead Code Elimination, as described in [“Batch Dead Code Elimination \(DCE\)” on page A-7](#).

To execute Batch Registration and Verification, enter the following command:

```
Brave.exe JobFile LogFile ParameterList
```

where:

- *JobFile* is the fully qualified pathname of Refresh.bj, the script file for Batch Registration and Verification.
- *LogFile* is the fully qualified pathname of the Brave log file.
- *ParameterList* contains the following parameters:
 - *Workspace=WorkspaceDirectory*, where *WorkspaceDirectory* is the fully qualified pathname of the workspace to register and verify (required).
 - *StageDir=StagingArea*, where *StagingArea* is the fully qualified pathname of the staging area for the workspace (required).

- `Project=ProjectName`, where *ProjectName* is the name of the project to register and verify (optional). Only source files in this project are registered and verified.
- `Options=OptionsScriptFile`, where *OptionsScriptFile* is the verification options override file (optional). For more information, see the note below.
- `Notify=NotificationFile`, where *NotificationFile* is a file that reports success or failure (optional).

You can expect the following files as output:

- `Brave.Notification` reports success or failure of the Batch Registration and Verification process.
- `logfile.Log` is the Brave log file.

Note: Brave uses, in order, the verification option settings of the project specified in the command line; or, if there is no project specified in the command line, the project with the same name as the workspace; or, if there is no such project, the first project in the tree below the workspace.

For information on how you can override project options on the Brave command line, contact support services.

Batch Dead Code Elimination (DCE)

Use the Batch Registration and Verification tool again *after* registration and verification to perform batch Dead Code Elimination. For each program analyzed for dead code, batch DCE generates a *component* that consists of the original source code minus any unreferenced data items or unreachable procedural statements.

Note: To perform batch DCE, you must be licensed to use the Application Architect product. For background on Dead Code Elimination, see *Creating Components* in the ATW document set.

To execute Batch DCE, enter the following command:

```
Brave.exe JobFile LogFile ParameterList
```

where:

- *JobFile* is the fully qualified pathname of DCE.bj, the script file for Batch DCE.
- *LogFile* is the fully qualified pathname of the Brave log file.
- *ParameterList* contains the following parameters:
 - *Workspace=WorkspaceDirectory*, where *WorkspaceDirectory* is the fully qualified pathname of the workspace to analyze for dead code (required).
 - *Entity=EntityName*, where *EntityName* is the source type of the workspace objects — Cobol, PLI, or Natural, for example (required).
 - *Pattern=*-DCE*, where *** is replaced by the name of the program analyzed for dead code (optional). If this option is set, components have names of the form *ProgramName-DCE*. If this option is not set, components have names of the form *DCEnn*, where *nn* is an incrementing number.
 - *Project=ProjectName*, where *ProjectName* is the name of the project to analyze for dead code (optional). Only this project is analyzed for dead code.
 - *Options=OptionsScriptFile*, where *OptionsScriptFile* is the DCE options override file (optional). For more information, see the note below.
 - *Notify=NotificationFile*, where *NotificationFile* is a file that reports success or failure (optional).

You can expect the following files as output:

- For each program analyzed for dead code, a component with a name of the form *ProgramName-DCE* (if you set the *Pattern* parameter) or *DCEnn*, where *nn* is an incrementing number.
- Brave.Notification reports success or failure of the Batch DCE process.
- *logfile.Log* is the Brave log file.

Note: Brave uses, in order, the DCE option settings of the project specified in the command line; or, if there is no project specified in the command line, the project with the same name as the workspace; or, if there is no such project, the first project in the tree below the workspace.

For information on how you can override project options on the Brave command line, contact support services.

Batch WebGen Generation Utility

Use the Batch WebGen Generation tool to generate HTML views of workspace repositories. To execute Batch WebGen Generation, enter the following command:

```
BWG.exe X-RefDatabase Workspace LogFile [Switches]
```

where:

- *X-RefDatabase* is the fully qualified pathname of the X-Ref database.
- *Workspace* is the workspace name in the X-Ref database. Use an asterisk (*) wildcard to specify all the workspaces in the database.
- *LogFile* is the fully qualified pathname of the BWG log file.
- *Switches* are:
 - /s stops processing if a verification error occurs for programs (default).
 - /sn stops processing if a verification error occurs for added or refreshed objects.
 - /w continues processing but issues a warning if any verification error occurs.

You can expect the following files as output:

- *workspace.WebGen.Notification* describes the completion status of the Batch WebGen Generation process.
- *log.file.Log* is the Batch WebGen Generation log file.

A-10 Using the Batch Refresh Feature
Running the Batch Refresh Feature

Identifying Interfaces for Generic API Analysis



Many legacy programs use a call to another program to interface with a database or transaction manager. In these circumstances, the program call is usually of less interest than its parameters — the file or table the called program writes to or reads from, for example. And yet it's the call that is modeled in the workbench repository.

When you select the **Perform Generic API Analysis** option in the Verification tab of the Project Options window ([step 10 on page 3-18](#)), you tell the parser to define relationships with the objects passed as parameters, in addition to relationships with the called programs themselves. This appendix shows you how to identify the programs and parameters to the parser.

To identify interfaces to the parser:

- 1 Open the file `workbench home\Data\Legacy.xml` in an editor.
- 2 Locate the `<APICalls>` section for the Cobol dialect you use.
- 3 Create `Call` name entries for each called program, of the form:

B-2 Identifying Interfaces for Generic API Analysis

```
<APICalls>
  <Call name="program_name" terminates="Yes|No">
    <param
      number="n"
      reltype="relationship_type"
      type="entity_type"
      memory="first_byte:number_of_bytes"
      usage="r|wr|w">
    </param>
    <param
      .
      .
      .
      .
      .
    </param>
  </Call>
</APICalls>
```

where:

- `name` is the name of the called program.
- `terminates` is Yes if the program terminates after the call (control is not received back from the program), No if the program returns normally.
- `number` is the number of the parameter that references the object of interest.
- `reltype` is the type of relationship to create with the object of interest.
- `type` is the entity type of the object of interest.
- `memory` specifies which bytes of the parameter should be used for the name of the object of interest, expressed as *first_byte:number_of_bytes*.
- `usage` specifies the usage of the parameter: r for input, w for output, rw for input/output.
- `value` is the parameter value to check for call matching purposes (future use).

For example:

```
<APICalls>
  <Call name="XREAD" terminates="No">
    <param number="1" reltype="ReadsDataport"
      type="Dataport" memory="4:6" usage="r">
    </param>
  </Call>
  <Call name="REWRITE" terminates="No">
    <param number="1" reltype="UpdatesDataport"
      type="Dataport" memory="4:6" usage="r">
    </param>
  </Call>
</APICalls>
```

Note: APIs using the same routine for multiple operations currently are not supported.

B-4 Identifying Interfaces for Generic API Analysis

Glossary

Activity Log

The Activity Log is a chronological record of your activities in the current [Asset Transformation Workbench \(ATW\)](#) session.

ADABAS

ADABAS is a Software AG relational [DBMS](#) for large, mission-critical applications.

Animator

Animator lets you step through the code displayed in a [HyperView](#) pane. You can choose program branches yourself, or have the animator choose them randomly.

API

API stands for application programming interface, a set of routines, protocols, and tools for building software applications.

applet

See [Java applet](#).

Application Analyzer

Application Analyzer is a set of non-invasive interactive tools used to analyze and document legacy systems.

Application Architect

Application Architect uses advanced algorithms to partition code into new [components](#) and perform [Dead Code Elimination](#).

Application Namespace tool

The Application Namespace tool creates a conveniently organized dictionary that helps you navigate through your system's terminology and modify it as necessary.

Application Partitioner

Application Partitioner identifies legacy subsystems and partitions them into self-contained projects based on an analysis of [repository](#) contents.

Application Profiler

Application Profiler, consisting of [WebGen](#) and [Profiler](#), generates a set of HTML views of a legacy application based on the [object model](#) created in a previous analysis.

AS/400

The AS/400 is a midrange server designed for small businesses and departments in large enterprises.

Asset Transformation Workbench (ATW)

Asset Transformation Workbench (ATW) is a suite of PC-based software products for analyzing, re-architecting, and transforming legacy applications.

Batch Application Viewer

Batch Application Viewer performs low-level analysis of batch processes.

batch refresh

The batch refresh feature lets you register and verify source files in batch mode. Other utilities packaged with the feature let you analyze application [complexity](#), run [WebGen](#), and, if you are licensed to use the Application Architect product, perform [Dead Code Elimination](#).

Bird's Eye pane

The Bird's Eye pane works with the [HyperView](#) Source pane to let you quickly identify the location of a code construct relative to the entire program.

BMS

BMS stands for Basic Mapping Support, an interface between application formats and [CICS](#) that formats input and output display data.

BSTR

BSTR is a Microsoft format for transferring binary strings.

business rule

A business rule is a named container that identifies and documents code segments according to their business function. Business rules encapsulate an application's business logic, making the application easier to understand, document, maintain, and test.

Business Rule Manager

Business Rule Manager lets you generate [business rules](#) from code segments extracted manually from source or autodetected.

Callie pane

The [HyperView](#) Callie pane displays a diagram that shows the flow of control between paragraphs or procedures in a program.

CDML

CDML stands for Cobol Data Manipulation Language, an extension of the [Cobol](#) programming language that enables applications programmers to code special instructions to manipulate data in a [DMS](#) database and to compile those instructions for execution.

Change Analyzer

Change Analyzer identifies the class of data items used to perform a business function in a legacy application. Among other uses, it lets you answer the kinds of "What if?" questions posed in the recent past by the industry-wide changes for Y2K, Zip+4, and the Euro dollar: "What if I change the type of this variable, or the length of this field — what *other* fields will I also have to change?"

CICS

CICS stands for Customer Information Control System, a program that allows concurrent processing of [transactions](#) from multiple terminals.

Clipper

The [HyperView](#) Clipper tool lets you create lists of candidates for [business rule](#) extraction, [event injection](#), and other tasks. Each list captures the results of a different stage of your analysis and serves as input for subsequent tasks.

Cobol

Cobol stands for Common Business-Oriented Language, a high-level programming language used for business applications.

COM

COM stands for Component Object Model, a software architecture developed by Microsoft to build [component](#)-based applications. COM objects are discrete components, each with a unique identity, which expose interfaces that allow applications and other components to access their features.

complexity

A [project's](#) complexity is an estimate of how difficult it is to maintain, analyze, transform, and so forth.

component

A component is a self-contained program that can be reused with other programs in modular fashion.

Component Maker

The [HyperView](#) Component Maker tool lets you “slice out” [components](#) from legacy applications — not only component executables but associated [Cobol](#) copybooks, [PL/I](#) includes, and [Natural](#) data areas as well.

computation-based component extraction

Computation-based [component](#) extraction lets you build a component that contains all the code necessary to calculate the value of a variable at a particular point in a program — the value of a DayOfTheWeek variable, for example, where it is used to populate a report attribute or screen.

Configuration Manager

Configuration Manager is a tool used to enable [Asset Transformation Workbench \(ATW\)](#) products and configure your workbench for the

tools, programming languages, legacy dialects, and character sets in use at your site.

construct

A construct is an item in the [parse tree](#) for a source file — a section, statement, condition, variable, or the like. A variable, for example, can be related in the parse tree to any of three other constructs — a declaration, a dataport, or a condition. You view the parse tree for a source file in the HyperView [Context pane](#).

Context pane

The [HyperView](#) Context pane displays the [parse tree](#) for the selected source file. The parse tree displays source code constructs — sections, paragraphs, statements, conditions, variables and so forth — in hierarchical form, making it easy to locate code constructs quickly.

copybook

A copybook is a common piece of source code to be copied into many [Cobol](#) source programs. Copybooks are functionally equivalent to C and C++ include files.

CORBA

CORBA stands for Common Object Request Broker Architecture, an architecture that enables distributed objects to communicate with one another regardless of the programming language they were written in or the operating system they are running on.

CSD file

CSD stands for [CICS](#) System Definition. A CSD file is a [VSAM](#) data set containing a resource definition record for every resource defined to [CICS](#).

database schema

A database schema is the structure of a database system, described in a formal language supported by the [DBMS](#). In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables.

dataport

A dataport is an input/output statement or a call to or from another program.

DB/2

DB/2 stands for Database 2, an IBM system for managing relational databases.

DBCS

DBCS stands for double-byte character string, a character set that uses two-byte (16-bit) characters rather than one-byte (8-bit) characters.

DBMS

DBMS stands for database management system, a collection of programs that enable you to store, modify, and extract information from a database.

DDL

DDL stands for Data Description Language (DDL), a language that describes the structure of data in a database.

Dead Code Elimination

Dead code elimination is a type of [component](#) extraction that removes unused (“dead”) code from a legacy application.

decision resolution

Decision resolution lets you identify and resolve dynamic calls and other relationships that the [parser](#) cannot resolve from static sources.

Diagrammer

Diagrammer lets you view the [relationships](#) between the objects in a [project](#) interactively — programs, files, [DDL](#), [Java](#), screen maps, and more. These relationships describe the ways in which application objects interact. Compare [Quick Diagrammer](#).

DMS

DMS stands for Data Management System, a Unisys database management software product that conforms to the CODASYL (network) data model and enables data definition, manipulation, and maintenance in mass storage database files.

domain-based component extraction

Domain-based component extraction “specializes” a program based on the values of one or more variables. The specialized program is typically intended for reuse “in place” — in the original application but under new external circumstances.

DPS

DPS stands for Display Processing System, a Unisys product that enables users to define forms on a terminal.

ECL

ECL stands for Executive Control Language, the operating system language for Unisys OS 2200 systems.

effort

Effort is an estimate of the time it will take to complete a task related to a [project](#), based on weighted values for selected [complexity](#) metrics.

EJB

EJB stands for Enterprise JavaBeans, a [Java API](#) developed by Sun Microsystems that defines a [component](#) architecture for multi-tier client/server systems.

EMF

EMF stands for Enhanced MetaFile, a Windows format for graphic images.

entity

An entity is an object in the [repository](#) model for a legacy application. The relationships between entities describe the ways in which the elements of the application interact.

entry point isolation

Entry point isolation extracts a [component](#) that contains only the functionality and data definitions required for invocation from the selected entry point.

event injection

Event injection is a type of [component](#) extraction that adapts a legacy program to asynchronous, event-based programming models.

Execution Path pane

The [HyperView](#) Execution Path pane displays a hierarchical view and diagram of the conditions that determine the flow of control in a program.

external subroutine extraction

External subroutine extraction is a type of [structure-based component extraction](#) that replaces a single internal subroutine in a [Natural](#) program with an external subroutine.

FCT

FCT stands for File Control Table (FCT), a [CICS](#) table that contains processing requirements for output data streams received via a remote job entry session from a host system. Compare [PCT](#).

Flowchart pane

The [HyperView](#) Flowchart pane displays a diagram of the flow of control between statements in a paragraph or procedure.

Global Data Flow tool

The [HyperView](#) Global Data Flow tool performs low-level analysis of program data flows.

HTML

HTML stands for HyperText Markup Language, the authoring language used to create documents on the World Wide Web.

HyperView

HyperView is a set of program analysis tools that let you analyze legacy programs interactively, by examining synchronized, complementary views of the same information — source, context, impacts, and so forth.

IDL

IDL stands for Interface Definition Language (IDL), a generic term for a language that lets a program or object written in one language communicate with another program written in an unknown language.

IDMS

IDMS stands for Integrated Database Management System, a Computer Associates database management system for the IBM mainframe and compatible environments.

Impact pane

The [HyperView](#) Impact pane displays a hierarchical view and diagram of the *impact trace* for a program variable. An impact trace de-

scribes how data items interact with each other in a program — exchange values, use each other in computations, and so forth.

Impact Report pane

The [HyperView](#) Impact Report pane shows the flow of data from a startup item to every data item that would be affected by its modification. The report is organized in hierarchical form according to the depth of the affected item.

IMS

IMS stands for Information Management System, an IBM program product that provides transaction management and database management functions for large commercial application systems.

Java

Java is a high-level [object-oriented programming](#) language developed by Sun Microsystems.

Java applet

A [Java](#) applet is a program that can be sent with a Web page. Java applets perform interactive animations, immediate calculations, and other simple tasks without having to send a user request back to the server.

JavaBeans

JavaBeans is a specification developed by Sun Microsystems that defines how [Java](#) objects interact. An object that conforms to this specification is called a JavaBean.

JCL

JCL stands for Job Control Language, a language for identifying a [job](#) to OS/390 and for describing the job's requirements.

JDBC

JDBC stands for Java Database Connectivity, a standard for accessing diverse database systems using the [Java](#) programming language.

job

A job is the unit of work that a computer operator or a program called a *job scheduler* gives to the operating system. In IBM mainframe operating systems, a job is described with job control language ([JCL](#)).

job dependencies

[Batch Application Viewer](#) treats [jobs](#) as dependent if one writes to a dataset and the other reads from the same dataset. Occasionally, you may want to define dependencies between jobs based on other criteria — administrative needs such as scheduling, for example.

logical component

A logical component is an abstract [repository](#) object that gives you access to the source files that comprise a [component](#).

MFS

MFS stands for Message Format Service, a method of processing [IMS](#) input and output messages.

Missing Copybooks Resolution tool

The Missing Copybooks Resolution tool resolves undefined variables in missing [copybooks](#) for [Cobol](#) programs verified with the [relaxed parsing](#) option.

Model Reference pane

The [HyperView](#) Model Reference pane displays the [parse tree](#) meta-model in text and diagram form.

name-based partitioning

Name-based partitioning is a partitioning algorithm that assigns source files to projects based on text matching of source file names with specified patterns.

Natural

Natural is a programming language developed and marketed by Software AG for the enterprise environment.

object model

An object model is a representation of an application and its encapsulated data.

object-oriented programming

Object-oriented programming organizes programs in terms of objects rather than actions, and data rather than logic.

ODBC

ODBC stands for Open Database Connectivity, a standard for accessing diverse database systems.

orphan

An orphan is an object that does not exist in the reference tree for any startup object. Orphans can be removed from a system without altering its behavior.

Orphan Analysis tool

The Orphan Analysis tool lets you analyze and resolve [orphans](#).

parser

The [Asset Transformation Workbench \(ATW\)](#) parser defines the [object model](#) and [parse tree](#) for a legacy application.

parse tree

A parse tree defines the relationships among the constructs that comprise a source file — its sections, paragraphs, statements, conditions, variables, and so forth.

PCT

PCT stands for Program Control Table, a [CICS](#) table that defines the transactions that the CICS system can process. Compare [FCT](#).

PL/I

PL/I stands for Programming Language One, a third-generation programming language developed in the early 1960s as an alternative to assembler language, [Cobol](#), and FORTRAN.

PL/I Call Diagrammer

PL/I Call Diagrammer performs low-level analysis of [PL/I](#) programs. Use it to examine call flows for internal procedures that the [Diagrammer](#) is unable to model.

profile

Profiles are HTML views into a [repository](#) that show all of the analysis you have done on an application. Profiles are convenient ways to share information about legacy applications across your organization.

Profiler

Profiler is a Web server-based tool that offers company-wide access to [profiles](#) of any repository in your organization. It gives managers, business analysts, testers, and customer support personnel convenient, browser-based access to analyzed legacy code.

project

A project is a logical subdivision of a [workspace](#). You might have a project for the batch portion of your application and another project for the online portion, for example. You can also use a project to collect items for discrete tasks — all the source files affected by a change request, for example.

QSAM

QSAM stands for Queued Sequential Access Method, a type of processing that uses a queue of data records—either input records awaiting processing or output records that have been processed and are ready for transfer to storage or an output device.

Quick Diagrammer

The Quick Diagrammer tool lets you view relationships for selected objects only, rather than an entire project. Compare [Diagrammer](#).

refactoring

Refactoring translates a program into a [component](#) with the same functionality and control flow, but a simpler syntax structure.

reference reports

[Asset Transformation Workbench \(ATW\)](#) offers three related reports that you can use to identify missing or unneeded program elements in application source: an unresolved report, an unREFERRED report, and a cross-reference report.

relationship

The relationships between entities in the [repository](#) model for a legacy application describe the ways in which the elements of the application interact.

relationship-based partitioning

Relationship-based partitioning is a partitioning algorithm that assigns source files to projects based on the extent to which the source

files are related. If two source files reference the same [copybook](#), for example, they can be regarded as “tightly related,” at least as compared with source files that do not reference the same copybook.

relationship weight

A relationship weight determines the importance of that [relationship](#) in calculating the connection between source files in [relationship-based partitioning](#).

relaxed parsing

Relaxed parsing lets you verify a source file despite errors. Ordinarily, the parser stops at a statement when it encounters an error. Relaxed parsing tells the parser to continue to the next statement.

repository

A repository is a database of program objects that comprise the model for a [workspace](#).

Repository Browser

The [Asset Transformation Workbench \(ATW\)](#) Repository Browser displays the contents of the current [workspace](#).

Resource Retriever

The Resource Retriever tool lets you identify and restore missing [CICS](#) file connectors and transactions in file control tables ([FCT](#)) and program control tables ([PCT](#)) for [Cobol](#) and [PL/I](#) programs.

Rules pane

The [HyperView](#) Rules pane lets you create [business rules](#) from code segments extracted manually from source or autodetected. You can also create business rules from candidates listed in [Clipper](#).

schema

See [database schema](#).

scope

The scope of a diagram determines the objects and [relationships](#) it displays. See [Diagrammer](#).

seed field

A seed field is the object of a [Change Analyzer](#) search for the class of data items that need to be changed.

Source pane

The [HyperView](#) Source pane displays view-only source code for the selected file and included files.

SQL

SQL stands for Structured Query Language, a standard language for relational database operations

structure-based component extraction

Structure-based component extraction is a type of [component](#) extraction that builds a component from a range of inline code — [Cobol](#) paragraphs, for example.

synonym

A synonym is a data field whose value is related to the value of the matched [seed field](#) — a field whose value is assigned by a MOVE or REDEFINE statement, for example.

system program

A system program is a generic program — a mainframe sort utility, for example — provided by the underlying system and used in unmodified form in the legacy application.

TIP

TIP stands for Transaction Processing, the Unisys real-time system for processing transactions under Exec control.

token

In the [Application Namespace tool](#), a token is an element in a program identifier delimited by a hyphen (-) or underscore (_). In the identifier WS01-CUST-FIELD, for example, there are three tokens: WS01, CUST, and FIELD.

transaction

A transaction is a sequence of information exchange and related work (such as database updating) that is treated as a unit for the purposes of satisfying a request and for ensuring database integrity.

Unknown Statements Resolution tool

The Unknown Statements Resolution tool resolves incorrect or unsupported statements in [Cobol](#) programs verified with the [relaxed parsing](#) option.

User Interface tool

The User Interface tool lets you analyze the interaction between legacy screens and program logic, and generate an [HTML](#) or [Java](#) GUI based on the interaction.

VALTAB

VALTAB stands for Validation Table, which contains the information the system needs to locate, load, and execute transaction programs. See also [TIP](#).

VSAM

VSAM stands for Virtual Storage Access Method, an IBM program that controls communication and the flow of data in a Systems Network Architecture network.

WebGen

WebGen generates HTML views of the repositories on your workstation. You can publish the views to [Profiler](#), where they can be accessed by any member of your organization with a browser.

workspace

A workspace is a named container for an application or a portion of an application. Workspaces can be divided into [projects](#).

XML

XML stands for Extensible Markup Language, a specification for creating common information formats.

Bibliography

- *IBM Asset Transformation Workbench v1.1 Getting Started (SC31-6877-00)*
- *IBM Asset Transformation Workbench v1.1 Preparing Projects (SC31-6879-00)*
- *IBM Asset Transformation Workbench v1.1 Analyzing Projects (SC31-6880-00)*
- *IBM Asset Transformation Workbench v1.1 Analyzing Programs (SC31-6878-00)*
- *IBM Asset Transformation Workbench v1.1 Profiling Projects (SC31-6881-00)*
- *IBM Asset Transformation Workbench v1.1 Creating Components (SC31-6876-00)*
- *IBM Asset Transformation Workbench v1.1 Parser Reference (SC31-6882-00)*
- *IBM Asset Transformation Workbench v1.1 Architecture Reference (SC31-6898-00)*

Notices

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

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

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

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

IBM World Trade Asia Corporation

Licensing

2-31 Roppongi 3-chome, Minato-ku

Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION •AS IS• WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
P.O. Box 12195, Dept. TL3B/B503/B313
3039 Cornwallis Rd.
Research Triangle Park, NC 27709-2195
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

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

Table 1. Trademarks

IBM	MVS
AS400	CICS
IMS	DB/2
Database 2	OS/390
S/390	z/OS

The following terms are trademarks of other companies:

Java and JavaScript are registered trademarks and Sun Solaris and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Index

A

- API analysis B-1
- application partitioning
 - assigning files to projects 9-13
 - creating and restoring drivers and utilities 9-14
 - creating projects 9-10
 - designating seed objects 9-13
 - editing relationship weights 9-15
 - locking projects 9-11
 - overview 9-1
 - performing name-based partitioning 9-15
 - performing relationship-based partitioning 9-18
 - sample use of tool 9-4
 - searching for files 9-11
- AS/400, generating copybooks 4-6
- autoresolving decisions 3-18, 3-20, 6-2, 6-6

B

- batch DCE A-7
- batch refresh
 - overview A-1
 - running A-3
- batch sort card analysis 3-9, 3-13

C

- calling chains analysis for JCL 3-8
- CICS resources
 - generating reports 7-3
 - restoring 7-1
- clustering quality graphs 9-4, 9-8
- COBOL-68 3-4
- copybooks, generating AS/400 4-6
- cross-reference report
 - exporting 5-12
 - generating 5-2
 - overview 5-1
- currency sign 3-3

D

- data operations 4-5
- dead code analysis 3-7, 3-11
- decision resolution
 - overview 6-1
 - resolving decision automatically 6-2, 6-6
 - resolving decision manually 6-2, 6-3
- dialects 3-2
- drivers
 - designating 9-14
 - overview 9-2
 - restoring 9-14

E

- entities 1-2
- Executive Report 4-10
- extensions 2-6
- extracted objects 3-10

F

- file connectors, restoring 7-3
- File Control Table (FCT) 7-1
- file extensions 2-6

G

- generic API analysis 3-19, B-1

H

- HyperView, enabling 3-7, 3-11

I

- IMS port analysis 4-5
- interproject diagram 9-4, 9-8, 9-19
- invalidating objects 4-2
- Inventory Report 4-9
- inventorying applications
 - overview 1-5

- resolving decisions 6-1
- restoring CICS resources 7-1
- using the relaxed parsing tools 8-1

J

- Japanese source files 2-10
- JCL
 - verification requirements for data flow analysis 3-8
 - verification requirements for IMS port analysis 4-5

L

- legacy dialects 3-2
- loading source files 2-12

M

- mainframe encoding 2-9
- MicroFocus Cobol 3-2
- missing copybooks resolution
 - generating reports 8-3, 8-20
 - overview 8-2
 - restoring copybooks 8-7
 - setting options 8-5

N

- name-based partitioning
 - overview 9-2
 - performing 9-16
- Natural library support 3-8, 3-13
- Natural line numbers 3-5

O

- orphan analysis
 - exporting reports 5-12
 - generating reports 5-7
 - overview 5-7

P

- PERFORM behavior 3-2
- post-verification tasks
 - enabling IMS port analysis 4-5
 - program analysis 4-2
 - viewing Executive Reports 4-10
 - viewing Inventory Reports 4-9
- Program Control Table (PCT) 3-9, 4-5, 7-1
- project
 - copying files 2-15
 - creating 2-14, 9-10
 - deleting 2-14, 9-11
 - exporting to a workspace 2-16
 - including objects 2-17
 - locking 9-11
 - moving files 2-15
 - overview 1-2
 - partitioning 9-1
 - renaming 9-10

R

- reference reports
 - enabling in verification options 3-7, 3-10
 - exporting 5-12
 - generating 5-2
 - overview 5-1
- refreshing source files 2-13, A-1
- refreshing the repository 4-2
- registering applications
 - loading source files 2-12
 - overview 1-3
 - setting options 2-6
- registering files 2-3, 2-12
- relationship weights
 - editing 9-15
 - overview 9-2

- relationship-based partitioning
 - overview 9-2
 - performing 9-18
- relationships 1-2
- relaxed parsing
 - discussed 3-12
 - overview 1-6
 - selecting 3-8
 - tools for use with 8-1
- repository
 - overview 1-2
 - refreshing 4-2
- resolving decisions 6-1

S

- seed objects
 - designating 9-14
 - overview 9-3
- Shift-JIS encoding 2-10
- sort card analysis 3-9, 3-13
- source files
 - encoding 2-9
 - exporting 2-14
 - refreshing 2-13
 - registering 2-12
 - verifying 4-1
- Sources folder 2-6
- staged parsing 3-10
- system programs 3-14, 5-6

T

- transaction-processing verification option 3-16
- transactions, restoring 7-6

U

- Unisys Cobol 3-2

- unknown statements resolution
 - applying substitution rules 8-16
 - exporting and importing substitution rules 8-18
 - generating groups 8-14
 - generating reports 8-11, 8-20
 - overview 8-10
 - setting options 8-19
- unreferred report
 - exporting 5-12
 - generating 5-2
 - overview 5-1
- unresolved report
 - exporting 5-12
 - generating 5-2
 - overview 5-1
- utilities
 - designating 9-14
 - overview 9-2
 - restoring 9-14

V

- verification options
 - and Configuration Manager 3-1
 - discussed 3-1
 - Legacy Dialects tab 3-2
 - project 3-16
 - propagating to other users 3-13
 - Settings tab 3-5
 - staged parsing 3-10
 - workspace 3-2
- verifying applications
 - invalidating objects 4-2
 - overview 1-4
 - verifying source files 4-1

W

- workspace
 - adding files 2-3, 2-12
 - creating 2-1
 - deleting 2-5
 - exporting files 2-14
 - exporting projects to 2-16
 - opening 2-4
 - overview 1-2
 - partitioning 9-1
 - registering files in 2-6
- workstation encoding 2-9



Product Number: 5724-L54

SC31-6879-00



(1P) P/N:5724-L54

