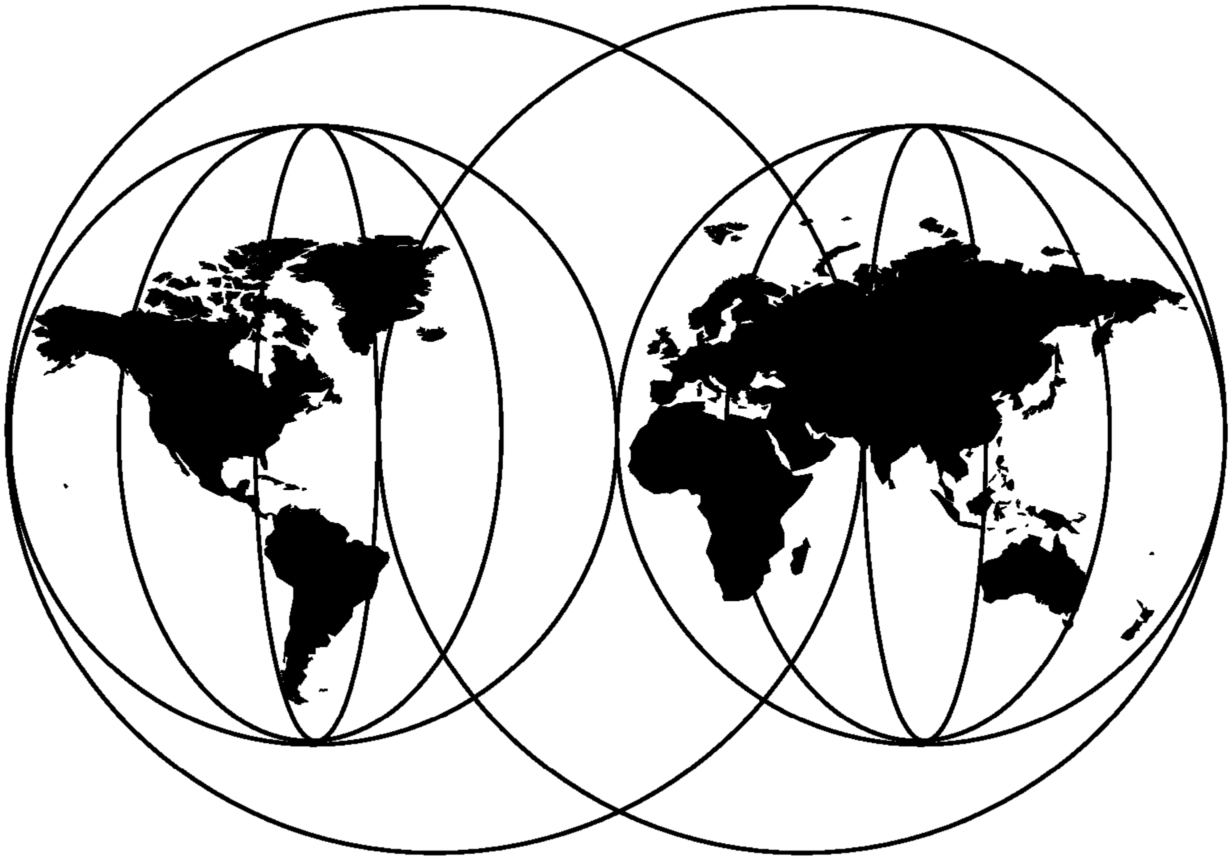
VisualAge 2000 Test Solution: Testing Your Year 2000 Conversion

Neil Bloomfield, Merrill Bani, Beth Flint



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2230-01

International Technical Support Organization

SG24-2230-01

VisualAge 2000 Test Solution: Testing Your Year 2000 Conversion

April 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix J, “Special Notices” on page 177.

Second Edition (April 1998)

This edition is for use with the versions of tools listed in Chapter 2, “The Tools Used by VisualAge 2000 Test Solution” on page 9.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Figures

Tables

Preface

This redbook, *VisualAge 2000 Test Solution: Testing Your Year 2000 Conversion*, demonstrates the VisualAge 2000 Test Solution in action, through a series of tutorials.

It is for managers or executives who wish to have a clearer understanding of Year 2000 testing, applications programmers who are involved in testing, and system programmers who support the applications programmers.

Part 1 sets out the framework of the VisualAge 2000 Test Solution, describes the tools that are used in the redbook, and describes how to install the sample files.

Part 2 presents the tutorials, where each tutorial works through one of the steps in the Test Solution. The tutorials provide detailed instructions for using the nominated tools.

By working through the tutorials, you will have a much clearer understanding of what is involved in Year 2000 testing.

This edition of the redbook is for OS/390 or VSE users running COBOL or PL/I programs, and using an OS/2- or Windows NT- or Windows 95-based work station.

Note that MVS is also supported by the book. MVS users should assume that for “OS/390” they can read “MVS,” except in product names.

A CD-ROM is provided with the redbook. The CD-ROM contains sample programs, data, and JCL, to work through the tutorials. As well, the CD-ROM contains a “slide-show” demonstration, which runs under Windows and Win-OS/2. The demonstration shows the various tools in action, and is a quick introduction to the tutorials, or for people who do not have access to a particular tool.

The tools used extensively in the tutorials of this redbook are:

- Enhanced SuperC
- Auto Test Performer (from WITT Year2000 for OS/2 or from VisualAge for COBOL, Test for OS/2) or WITT Year2000 for Windows
- DFSORT

The following tools are used in one tutorial, or are discussed in a tutorial:

- VisualAge for COBOL, Test for OS/2, COBOL Tester
- IBM Application Testing Collection, Coverage Assistant
- IBM Application Testing Collection, Distillation Assistant
- IBM Application Testing Collection, Source Audit Assistant

Debug Tool is used in three tutorials.

What Is Special About Year 2000 Testing

Year 2000 testing is different from other testing. Here are some of the differences:

- The Year 2000 has a deadline. This deadline is fixed and firm. You cannot push it out a few days. If your applications are not working by 2000, the outcome could be disastrous.
- With Year 2000 conversions, you are not trying to change anything. You are trying to make sure that things stay the same.
- When you test, you are testing two different situations. One is that your applications are able to handle dates beyond 1999. The other is that your applications are able to run beyond 2000. You have to test both of these situations.

Your IT organization already knows about building testbeds, change control, parallel running, regression testing, and putting changes into production. All of these skills are used in Year 2000 testing. But your IT organization also needs to understand how to test your Year 2000 conversion properly. When the calendar ticks over to 2000, you do not have time for major bug fixing. You only get one chance to get your Year 2000 conversion right.

This redbook will help you along the way.

Introducing VisualAge 2000 Test Solution

This redbook is part of VisualAge 2000 Test Solution.

VisualAge 2000 Test Solution is IBM′s comprehensive Year 2000 test process, which is designed to facilitate Year 2000 testing, which is fundamentally different from traditional testing.

This redbook does not propound a revolutionary method of testing. Rather, it extends, in a practical way, the test process outlined in *VisualAge 2000: Methodology and Tools Implementation*, available from `http://www.software.ibm.com/ad/va2000/y2k/`.

This redbook is a set of related tutorials. Each tutorial is one step in the complete process of testing Year 2000 changes. The complete process builds a prototype, and establishes that it works.

The framework of the test process set out in the Internet article suggests appropriate tools for each step. However, some steps are not explicitly spelled out. For example, what tool do you use to convert a distilled testbed? And what tool do you use to age a converted testbed?

This redbook extends the test process by testing it. And along the way, it provides extra tools or sets of procedures which fill the gaps. By the end of the redbook, the example application has been successfully tested for Year 2000 changes.

The purpose of this redbook is not to set down the only way to test Year 2000 fixes. Rather, it is to suggest a sound framework for testing, and to demonstrate tools that can fill the framework.

The problems that were encountered while the redbook was being developed are listed in Appendix A, "Other Matters to Consider Before You Test" on page 129. If you think that Year 2000 testing poses no challenges, have a read.

The tutorials show in detail the ways in which the tools and procedures work. Your challenge is to take these tools and procedures, and adapt them to the needs of your particular environment and applications. By doing so, you can hopefully turn our narrow track into a six lane highway.

Start Thinking About Testing Today

Traditionally, testing comes last. After you have identified the problem, and planned how to deal with it, and coded the changes, you finally get around (if you have the time) to testing the changes.

To make Year 2000 testing successful, you must start thinking about it immediately. The only way to successfully develop test data is to have an intimate knowledge of your system. Here are some examples:

- After you have converted your application, is a particular date held in an expanded or windowed format? If you do not know the answer, you cannot develop an appropriate test regime for the date. And the best time to find this out is when you are identifying code that needs change, and planning what sort of changes to make.
- There are a set of dates that in particular need thorough testing. These include 31 December 1999, 1 January 2000, 28 February 2000, 29 February 2000, 1 March 2000. For your organization, there are sure to be more. If you are aware of these high-profile dates when you are coding, then you build your code to make sure it handles these dates correctly. This is much better than finding out when you test that these dates are not handled correctly. The whole point of testing is to find problems, but you want to find as few problems as possible.
- When you age your data for testing, should you age it by a few years, or by 28 years? Unless you have an intimate understanding of your applications, you will not know the answer to this question. If you learn the answer to this question while you are identifying problems, then you can start to plan for proper testing. And you need to know the answer early, because it is possible that the answer is a few years, ten years and 28 years—so you need to be ready to age data not once, but many times.

From a distance, Year 2000 testing may look straight-forward. Our experience, in developing this set of tutorials, has been that thorough Year 2000 testing is tedious, repetitious, and full of challenges. The sooner you start thinking about testing, the greater your chance of actually getting everything finished before January 1, 2000.

Before You Begin Testing

If you have done no previous reading about Year 2000 fixes, you will find it useful to read about the steps that come before testing, bearing in mind the previous comments that testing begins now.

The steps are to:

- Inventory your software and hardware portfolio.
- Identify Year 2000 exposure.
- Determine what date changing technique or techniques you are going to use to fix Year 2000 problems.
- Apply the techniques to convert your code and data.

This redbook assumes that you have an understanding of these previous steps, and provides no guidance on these steps.

There are many sources available to provide this sort of information, in particular:

- The document *VisualAge 2000: Methodology and Tools Implementation*, available from `http://www.software.ibm.com/ad/va2000/y2k/`, which as well as outlining the test methodology deals with the prior steps.
- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation* available from `http://www.software.ibm.com/year2000/resource.html`.

Who Should Read This Book

This book is for the following people:

The manager or executive who should focus on Chapter 1, "An Outline of the Testing Process" on page 3, which provides an outline of the whole testing process. Your system programmers and applications programmers worry about the fine details of the testing. (You may need to consult with these people to find out how long the process can be expected to take, and what tools they want to use.)

The system programmer who should find Chapter 2, "The Tools Used by VisualAge 2000 Test Solution" on page 9, and Chapter 3, "Installation and Customization" on page 19, of most interest.

Chapter 2, "The Tools Used by VisualAge 2000 Test Solution" on page 9, lists the tools that are used in the VisualAge 2000 Test Solution testing process. As well, the chapter points you to the places in the documentation of the tools that tells you how to install the tools onto your test computer.

Chapter 3, "Installation and Customization" on page 19, provides explicit instructions for copying sample files from the provided CD-ROM to host and work station, and for customizing the host files. This chapter also lists the setup steps needed before you can run the tutorials.

The applications programmer who is the prime user of the tutorials, which start at Chapter 4, "Introducing The Tutorials" on page 35. This chapter introduces the tutorials, and the following tutorial chapters work through the testing process in detail. For each process, the tutorial sets out the aim of the process, the input to the process and the output of the process. It then explains the process, and breaks it up into the individual tasks and steps needed with the recommended tool.

The Team That Wrote This Redbook

This redbook was produced by a team based at the Australian Programming Centre.

Neil Bloomfield has 15 years of experience as an applications programmer, working on a wide variety of commercial systems, and ten years of experience in Information Development, writing software user guides and on-line help, and testing interactive systems.

Merrill Bani has 21 years experience as an applications developer, working on the VSE and MVS platforms. She has designed and implemented commercial-based systems for a variety of companies ranging from large cooperatives to banks.

Beth Flint edited this redbook. Beth is an Advisory Programmer in IBM USA with 20 years programming experience in the fields of operating systems, utility programs, and compilers. Currently, she is the web master for the IBM Year 2000 Technical Support Center web site. In life prior to IBM, she taught Computer Science at North Carolina State University.

This Redbook was commissioned by **Tom Dunham**. Tom is Program Director for the IBM Year 2000 Technical Support Centers and Manager of IBM VisualAge Services for Application Development. He has been involved with the development and delivery of a wide number of language products and solutions from IBM′s Santa Teresa Laboratory for 12 years.

Valuable support for this project was provided by experts from the Australian Programming Centre.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Part 1. Preparing for the Test Tutorials

Chapter 1. An Outline of the Testing Process

This redbook demonstrates a process that you can apply to your Year 2000 testing. In addition to describing the concepts and structure of the process, this redbook provides instructions for using particular tools. Where possible, these instructions are presented as detailed, step-by-step procedures that use sample files supplied on the CD-ROM.

1.1 Previous Steps in the Year 2000 Conversion

The testing process is the final stage of a Year 2000 conversion.

The other stages in a Year 2000 conversion are:

- Developing an inventory and assessing Year 2000 impact
- Analyzing code and data, and planning for implementation
- Modifying code—finding and fixing
- Modifying data

Of course, while you are working through these four steps, you need to be aware of what is required in testing. In fact, you must start preparing for testing before you start modifying code and data. If you do not, testing will become more difficult because you will not have a sensible pre-change benchmark for comparison.

1.2 The Testing Process

The overall testing process is displayed in Figure 1 on page 4.

This redbook is basically concerned with the process of validating your Year 2000 changes. The focus is very much on making sure that you haven′t introduced any bugs into your code, and this is what is shown in the figure. The final steps of testing, performance and acceptance testing, are mentioned only in passing in this redbook. You should be able to use or adapt your standard performance and acceptance testing to Year 2000 testing.

A “testbed” is a set of master files (or databases) and the transactions that are applied to the master files to update them. The updated master files and any reports that are produced constitute the “results.”

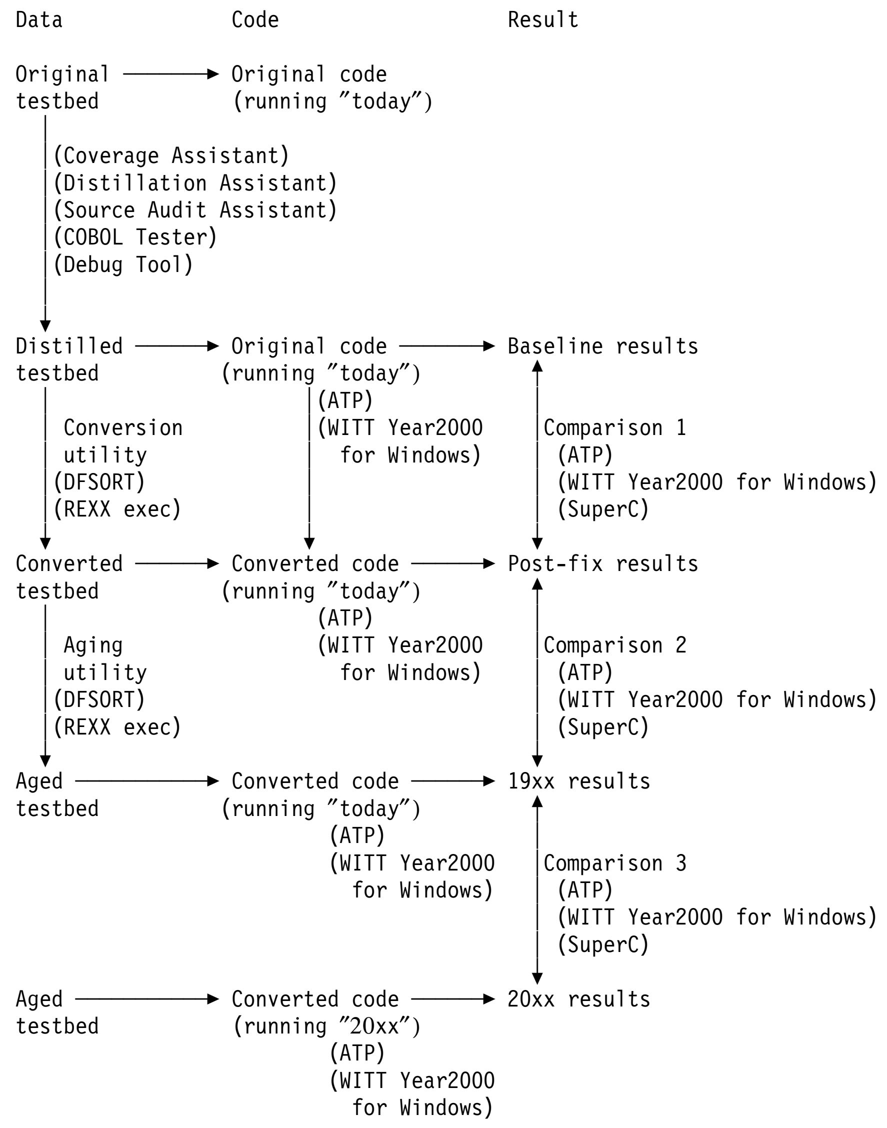


Figure 1. The Testing Process

The first column shows how the data changes over the period of testing. The first set of data is the original testbed. As a result of coverage and distillation, this is transformed to the distilled testbed. The distilled testbed is converted to take into account whatever data structure changes are applied for Year 2000 conversion. At this stage, the testbed holds only 19xx dates. The final step for the data is to age dates, so that some dates fall into 20xx. This aged testbed is used to produce two sets of results.

The second column shows the progress with code and the test environment. The first set of code is the original (current) code, running “today” (with a 19xx system clock date). This code is used twice, to distill the data, and to provide the baseline results. The code is then converted, to account for Year 2000 changes. The converted code is used three times, twice “today” with different input data, and finally at some 20xx system clock date, using data that includes 20xx dates.

The third column shows the results. Each set of results should vary from the previous set in a predictable way. The whole point of testing is to try and find variations that are not predicted.

The data in the results can be described, so that the comparison tool is able to take into account the known, predicted, differences. The comparison tool rejects a comparison as a mismatch only if the difference has not been described.

Comprehensive testing has ten steps:

Step 1. Developing Path Coverage and Distilling the Testbed
Build a testbed that exercises the maximum amount of code, but contains no superfluous data. This testbed is used at least four times during Year 2000 testing, so it makes sense to keep it as small as possible.

Step 2. Creating the Baseline Results
Process the original testbed through the legacy code, to produce updated master files, reports, and screen images.

Step 3. Building the Converted Testbed
Apply conversion processes to the distilled testbed. The conversion processes change dates, so that the dates are formatted in the manner that the converted programs expect. Different conversion techniques are applied to the master files and the transactions. The outcome after conversion is the converted testbed.

Step 4. Creating the Post-Fix Results
Process the converted testbed through the converted code, to produce converted master files, reports, and screen images.

Step 5. Comparing the Baseline with the Post-Fix Results
Compare the baseline results with the post-fix results. All matches should be exact, after accounting for conversion. If a match is not exact, then the converted code may be in error, though the error might also be in the conversion or the comparison.

Step 6. Building the Aged Testbed
Apply aging techniques to the converted testbed, to produce an aged testbed. The information in this testbed is the same as the converted testbed, except all dates are “older.” For example, a date in 1997 might move to 2005.

Step 7. Creating the 19xx Results
Process the aged testbed through the converted code, to produce aged master files, reports, and screen images.

Step 8. Comparing the Post-Fix Results with the 19xx Results
Compare the post-fix results with the 19xx results. All matches should be exact, after accounting for aging. If a match is not exact, then the converted code may be unable to correctly process dates that are beyond 1999, though the error may arise from aging techniques or be in the comparison.

Step 9. Creating the 20xx Results
Process the aged testbed through the converted code running after 1999, to produce updated aged master files, reports, and screen images. 19xx results and 20xx results are produced from the same testbed. The only difference is the system clock date.

Step 10. Comparing the 19xx Results with the 20xx Results
Compare the 19xx results with the 20xx results. All matches should be exact. If a match is not exact, then the converted code may be unable to process correctly when it is run after 1999.

Figure 2 provides a summary of testing, and shows how each new set of results is the result of one more factor being changed from “today” to Year 2000, 20xx, readiness. “Time” in this chart represents the system clock date.

Results	Code	Dates	Time
Baseline	Today	Today	Today
Post-fix	20xx	Today	Today
19xx	20xx	20xx	Today
20xx	20xx	20xx	20xx

Figure 2. A Summary of the Testing Process

The testing process may seem complicated and cumbersome, but the summary shows how only one factor is changed at a time. This means that if a comparison brings up an unexpected result, you need focus only on the change of one factor. By using an approach that is methodical and thorough, you should end up saving time.

1.3 High- and Low-Risk Applications

A high-risk application is an application that, if it were to fail when “00” arrives, would have catastrophic results—such as putting you out of business or halving your profit. You must test such applications thoroughly. This means that, in particular, you make sure that the testbed is built using coverage and distillation. And you do not leave out any step in the testing process.

A low-risk application is one where failure is inconvenient, but the application can be put to one side till you get around to fixing it. Although the testing process is the same, you may not bother to develop a testbed using coverage and distillation. Instead, you may use whatever you have on hand, without testing for particular conditions. And you may not even bother to go beyond comparing baseline and post-fix results, with the thought that you can fix any problems with the converted programs if and when they arise.

The “best” strategy is to treat every application as a high-risk application. However, you may simply not have enough time to deal with all applications equally. In this case, it makes sense to prioritize applications, so that you can ensure the high-risk applications are ready in time.

1.4 What To Do When Testing Finds a Bug

Imagine that you are comparing baseline results with post-fix results. They do not match. You have found a bug in the code! Well, maybe you have and maybe you haven′t.

While the tutorials were being developed, the comparisons failed. Investigation found three different problems:

- The conversion utility introduced spurious characters into the master file.
- The window parameter for the conversion utility was set incorrectly.
- The converted program was incorrect.

One of these problems has been left in the tutorials, to demonstrate the “restart” process.

An error in a conversion routine must be attended to, because at some stage production files will be converted, and errors must not be introduced into production files. An error in an aging utility is not of great concern, because production data is not aged through an aging utility. It just grows old “naturally.”

So when testing finds a bug, keep an open mind, and look not just at the converted code, but also at the other tools that support the testing process.

Chapter 2. The Tools Used by VisualAge 2000 Test Solution

This chapter briefly describes tools that are part of the VisualAge 2000 Test Solution. The chapter also suggests where you can find information about installing and using each tool.

For information about ordering the tools, see Appendix G, “Program Information for Tools Used by VisualAge 2000 Test Solution” on page 163.

2.1 Auto Test Performer

Auto Test Performer (ATP) is a component of WITT Year2000 for OS/2 and of VisualAge COBOL, Test for OS/2. It runs on the OS/2 desktop.

ATP is a record, replay, and compare tool for automated testing of client applications. You can use ATP to test interactive applications that have an OS/2, a 3270, or a 5250 user interface. ATP does not support DOS or WIN-OS/2 applications.

The equivalent Windows 95 and Windows NT product is WITT Year2000 for Windows (see 2.10, “WITT Year2000 for Windows” on page 17).

ATP can create a script by recording keystrokes and mouse actions as you use the application. At a later stage, you can replay the script with ATP. This means that ATP is effectively typing the keys and controlling the mouse. The big advantage of this approach is that it provides identical input to the application every time, which is critical for accurate regression testing.

ATP also captures screens, and provides a means of comparing equivalent screens from two different processing runs.

If screens are changed structurally as part of the Year 2000 conversion, the screen comparison will always fail. An example of a structural change is adding the century in front of the year in a date (for example, “970318” becomes “19970318”). Looking at many screens to find differences, and make sure that the differences are OK, is a tiring and difficult process. Masking areas of a screen to avoid comparing these areas makes the comparisons meaningless.

Thus, the preferred process is to use ATP to capture screens, then use Enhanced SuperC to compare the screens. This works provided that the screens hold only text information. The advantage of using Enhanced SuperC is that it can be configured to take into account structural differences when doing the comparison.

2.1.1 Hardware Requirements

- Processor: i486 66MHz minimum (i486 100MHz or later recommended)
- Display: VGA minimum (SVGA recommended)
- RAM:
  - Base Package - 16MB minimum, 24MB recommended
  - Full Package - 24MB minimum, 32MB recommended
- Disk Space:
  - 30MB for Base Package (excluding prerequisite software)
  - 50MB for Full Package (excluding prerequisite software)

2.1.2 Software Requirements

- OS/2 Warp Version 3.0 or later
- To use Auto Test Performer′s 3270/5250 capability, at least one of the following:
  - Communications Manager/2 Version 1.11
  - Personal Communications AS/400 and 3270 for OS/2 Version 4.1
  - Personal Communications 3270 for OS/2 Version 4.1 (3270 only)
  - Personal Communications AS/400 for OS/2 Version 4.1 (5250 only)

2.1.3 Installation and Usage Hints

If you want to use ATP with OS/2 Warp Version 4.0, you must install ATP Version 1.10 or later. VisualAge Test Version 1.0 contains ATP Version 1.01, for which you can obtain an upgrade to ATP Version 1.11 from:

`ftp://ftp.software.ibm.com/ps/products/visualagetest/fixes/atp/atp111.zip`

or (internally to IBM)

`http://cuvatest.yamato.ibm.com/atp111/index.html`

For online information about ATP, double-click the *Auto Test Performer User′s Guide* in the VisualAge Test Information group within the VisualAge Test group.

2.2 COBOL Tester

COBOL Tester is a component of VisualAge Test for OS/2. It runs on the OS/2 desktop.

You can use COBOL Tester to perform the initial testing of new and changed code in a program module written in a language compatible with COBOL/370. COBOL Tester does not yet support testing of features introduced in COBOL for MVS & VM or COBOL for OS/390 & VM, such as object-oriented features.

Without your preparing special test drivers for the module, COBOL Tester assists you in defining the input values and expected output values of data, running the module, verifying the actual output values of data, and assuring adequate path coverage.

COBOL Tester allows you to interactively increase the path coverage of a program. To increase the path coverage, you must provide input values. After you have achieved 100% coverage, the input values constitute a distilled data set.

COBOL Tester can compare calculated output values with expected output values for the given input values. This provides a variation of regression testing.

2.2.1 Hardware Requirements

- Processor: i486 DX2 66MHz minimum (i486 100MHz or higher recommended)
- Display: VGA minimum (SVGA recommended)
- RAM: 24MB minimum, (32MB recommended)
- Disk Space: 50MB for Full Package (excluding prerequisite software)

2.2.2 Software Requirements

- OS/2 Warp Version 3.0 or later
- VisualAge for COBOL for OS/2 Version 1 Release 1 or later
- IBM WorkFrame (to use COBOL Tester with IBM WorkFrame)

If you use VisualAge for COBOL for OS/2 Version 1 Release 1, you require IBM WorkFrame Version 3 if you want to use COBOL Tester with WorkFrame. Later versions of VisualAge for COBOL for OS/2 contain WorkFrame.

VisualAge for COBOL for OS/2 Version 2.1 requires OS/2 Warp Version 4.0 or later.

2.2.3 Installation and Usage Hints

If you want to use COBOL Tester with VisualAge for COBOL, Version 2.0, install the fixpack ct1001, available from the ftp site:

```
ftp://ftp.software.ibm.com/ps/products/visualagetest/fixes/cobtest
```

For online information about COBOL Tester, double-click the *COBOL Tester User's Guide* in the VisualAge Test Information group within the VisualAge Test group.

2.3 Coverage Assistant

Coverage Assistant is a member of the IBM Application Testing Collection (ATC).

Coverage Assistant displays code coverage for application programs that are written in COBOL or PL/I and compiled by any of these compilers:

- IBM COBOL for MVS & VM Version 1 Release 2 Modification 0
- IBM VS COBOL II Version 1 Release 4 Modification 0
- IBM OS/VS COBOL Version 1 Release 2 Modification 4
- IBM PL/I for MVS & VM Version 1 Release 1 Modification 1
- IBM OS PL/I Optimizing Compiler Version 2 Release 3 Modification 0
- IBM PL/I Optimizing Compiler Version 1 Release 5 Modification 1

Coverage Assistant coverage reports provide the percentage of lines executed. The reports list unexecuted code segments or conditional branches that did not execute in both directions.

The program listing displays symbols showing whether an instruction was executed or how a conditional branch instruction was executed.

Coverage Assistant reports on an entire program or particular segments of code. This makes it possible to focus on code that handles dates.

Running Coverage Assistant requires setup, execution and report steps.

An ISPF panel interface provides the means to create JCL to run the Coverage Assistant steps.

During the setup steps, Coverage Assistant analyzes assembler statements in the compiler output listings to determine where to place breakpoints in copies of the object modules to be examined, and inserts the breakpoints.

2.3.1 Hardware Requirements

OS/390 system resource requirements for Coverage Assistant are detailed in "Resources Needed by Coverage Assistant and Distillation Assistant" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide*. The OS/390 system disk space and storage space required are affected by the size of the object module, the number of breakpoints and the number and size of the report listings.

2.3.2 Software Requirements

- OS/390
- ISPF
- Language Environment for MVS and VM Version 1.5 (or later) runtime library

2.3.3 Installation and Usage Hints

To use Coverage Assistant, you must install the Application Testing Collection. This involves performing a system installation for each OS/390 system, and a user installation for each user.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for system installation directions.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for user installation directions.

It is recommended that you test the installation using the sample programs and their JCL, as appropriate, depending on the versions of compiler you use. The sample programs and their JCL are shipped with the tool.

2.4 Debug Tool

Debug Tool is a general-purpose debug tool that is available as a part of the following IBM COBOL C, C++, and PL/I compiler products for host systems:

- IBM COBOL for OS/390 & VM
- IBM COBOL for MVS & VM
- IBM C/C++ for MVS & VM
- OS/390 C/C++
- IBM PL/I for MVS & VM
- IBM COBOL for VSE/ESA
- IBM C for VSE/ESA
- IBM PL/I for VSE/ESA

Debug Tool is included also in the Cooperative Development Environment/370 (CODE/370) for the older languages:

- IBM SAA AD/Cycle C/370
- IBM SAA AD/Cycle COBOL/370
- IBM SAA AD/Cycle PL/I for MVS & VM
- IBM VS COBOL II
- IBM OS PL/I

Debug Tool is applicable for application programs that are written in COBOL, PL/I, or C/C++, and compiled by any of the above compilers.

Debug Tool runs on the host, but under OS/390 it can be run from a workstation using the remote debug facility. You can use Debug Tool interactively or in batch. You can record a debug session in a log file, which enables you to replay the session.

You can use Debug Tool to create and execute test scripts and to monitor program variables and modified lines of code.

Debug Tool allows you to set breakpoints in code, and to display the values of selected variables at each breakpoint. You can single-step through a program, following the program logic.

You can use Debug Tool to capture test cases for program validation. You can use the frequency count of Debug Tool to check the coverage of a program.

And you can use its general features to debug programs that are failing to perform to expectations.

Year 2000 scripts for Debug Tool are available from:

`http://www.software.ibm.com/year2000/tools19.html`

The Debug Tool script files show code examples that use date expansion, compression, and fixed and sliding windows. Input scripts for Debug Tool can be used by programmers and testers to help validate that Year 2000 conversion techniques have been properly implemented.

2.4.1 Hardware Requirements

Debug Tool runs on any processor supported by the operating systems listed in Software Requirements below.

2.4.2 Software Requirements

Debug Tool requires one of the following operating systems:

- MVS
- VM
- OS/390
- VSE/ESA

2.4.3 Installation and Usage

For installation directions, refer to the *Installation and Customization Guide* for Debug Tool for your operating system. For usage instructions, refer to the *User's Guide and Reference* for Debug Tool for your operating system.

2.5 Distillation Assistant

Distillation Assistant is a member of the IBM Application Testing Collection (ATC).

Distillation Assistant supports the distillation of QSAM or VSAM input data sets for applications compiled by the following IBM compilers:

- IBM COBOL for MVS & VM Version 1 Release 2 Modification 0
- IBM VS COBOL II Version 1 Release 4 Modification 0
- IBM OS/VS COBOL Version 1 Release 2 Modification 4

- IBM PL/I for MVS & VM Version 1 Release 1 Modification 1
- IBM OS PL/I Optimizing Compiler Version 2 Release 3 Modification 0
- IBM PL/I Optimizing Compiler Version 1 Release 5 Modification 1

Distillation is the process of reducing an input data set to a minimum size, without reducing coverage. Using a distilled data set may greatly reduce testing time.

While a program under test is running, the Distillation Assistant monitor records the logical keys (for data from a specified input data set) that cause new coverage. When you have finished testing, you can use Distillation Assistant to make a new input data set consisting of the input records that have those logical keys. The new input data set is the distilled data set.

2.5.1 Hardware Requirements

OS/390 system resource requirements for Distillation Assistant are described in "Resources Needed by Coverage Assistant and Distillation Assistant" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide*.

2.5.2 Software Requirements

- OS/390
- ISPF

2.5.3 Installation and Usage Hints

To use Distillation Assistant, you must install the Application Testing Collection. This involves performing a system installation for each OS/390 system, and a user installation for each user.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for system installation directions.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for user installation directions.

It is recommended that you test the installation using the sample programs and their JCL, as appropriate, depending on the versions of compiler you use. The sample programs and their JCL are shipped with the tool.

2.6 Enhanced SuperC

You can use the Enhanced SuperC tool to compare two files. It is supplied as a PTF for High Level Assembler for MVS & VM & VSE Toolkit Feature.

For Year 2000 testing, Enhanced SuperC is used to compare files on a record by record basis. When used in this way, it can compare files, reports and screen images.

Enhanced SuperC is an extended version of SuperC designed to handle Year 2000 testing. In particular, Enhanced SuperC is able to compare dates stored with two digits for the year (YY), and dates stored with four digits for the year (YYYY). Because Enhanced SuperC is able to compensate for differences in the representation of dates, any differences it finds between files should point to genuine problems.

Enhanced SuperC runs interactively or in batch mode under VM, and in batch mode under OS/390 and VSE/ESA.

2.6.1 Hardware Requirements

Enhanced SuperC runs on any processor supported by the operating systems listed below.

2.6.2 Software Requirements

Enhanced SuperC requires one of the following operating systems:

- VM
- OS/390
- VSE/ESA

2.6.3 Installation and Usage

For installation instructions, refer to the *High Level Assembler for MVS & VM & VSE Toolkit Feature Installation and Customization Guide Release 2*, GC26-8711-02. For usage instructions, refer to the *High Level Assembler for MVS & VM & VSE Toolkit Feature User′s Guide Release 2*, GC26-8710-03.

2.7 Data Facility Sort

IBM Data Facility Sort (DFSORT) is a program for OS/390 or VSE for sorting, merging, and copying small to very large amounts of data. You can also use DFSORT to analyze data and produce reports, and to transform data at the record, field, and bit level.

Year 2000 features of DFSORT allow you to sort, merge and transform a wide variety of dates with two-digit years, according to a specified sliding or fixed century window.

New formats and a run-time option allow you to:

- Set the appropriate century window for your applications
- Order two-digit character, zoned decimal, packed decimal, or decimal year dates according to the century window
- Transform two-digit character, zoned decimal, packed decimal, or decimal year dates to four-digit character (or zoned decimal) year dates according to the century window
- Order parts of packed decimal fields (such as month and day in date fields)
- Transform packed decimal and zoned decimal fields to character fields, and insert literals such as ′/′ in transformed fields

Year 2000 features of DFSORT are provided in DFSORT/VSE Version 3 Release 3, in PTF UN99635 for DFSORT/VSE Version 3 Release 2, and in PTF UN90139 and PTF UQ05520 for DFSORT Version 1 Release 13 (for MVS).

2.7.1 Hardware Requirements

DFSORT runs on any processor supported by the operating systems listed in Software Requirements below.

2.7.2 Software Requirements

DFSORT requires one of the following operating systems:

- OS/390
- VSE/ESA

2.7.3 Installation and Usage

You can install PTF UQ00530 for DFSORT Version 1 Release 13 to simplify the installation of DFSORT under OS/390. For other installation information for DFSORT under OS/390, refer to the *DFSORT Release 13 Installation and Customization Guide*, SC26-4109.

For usage instruction for DFSORT under OS/390, refer to *DFSORT Release 13 Getting Started with DFSORT*, SC26-4109.

For instructions for installing DFSORT/VSE, refer to *DFSORT/VSE Installation and Tuning Guide*, SC26-7041.

For usage instructions for DFSORT/VSE, refer to *DFSORT/VSE Getting Started with DFSORT/VSE*, SC26-7101.

For information about the Year 2000 features of DFSORT (for OS/390 and VSE), look under the **Sort Specific Papers** heading of:

`http://www.software.ibm.com/year2000/resource.html`

2.8 REXX

REXX is used to customize files, before the tutorials run.

If you are running under VSE, and you do not have REXX, you have to customize the files by hand. REXX is supplied with VSE/ESA Version 2. REXX was an optional product with VES/ESA 1.4.3.

2.9 Source Audit Assistant

Source Audit Assistant is a member of the IBM Application Testing Collection (ATC).

Source Audit Assistant compares two levels of source code, usually before and after modification. Source Audit Assistant currently supports OS/390 and the following programming languages:

- C/C++
- PL/I
- COBOL

Running Source Audit Assistant is a matter of providing details about the old and new source files. The Compare report shows which lines are deleted, inserted or changed. The Focus output report also uses seeds (variable names) to extract information from the Compare report. The Focus report identifies all specified variables that did not appear in any New line in the Compare report, and all lines in the Compare report that did not contain at least one seed.

The Compare report provides the vehicle for targeted coverage. Your test strategy must produce coverage for the lines that have changed.

2.9.1 Hardware Requirements

OS/390 system resource requirements for Source Audit Assistant are described in "Resources Needed by Source Audit Assistant" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide*.

2.9.2 Software Requirements

- OS/390
- ISPF

2.9.3 Installation and Usage Hints

To use Source Audit Assistant, you must install the Application Testing Collection. This involves performing a system installation for each OS/390 system, and a user installation for each user.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for system installation directions.

Refer to "Installation" in *IBM Application Testing Collection for MVS Version 1 Release 2 User's Guide* for user installation directions.

2.10 WITT Year2000 for Windows

WITT Year2000 for Windows is functionally equivalent to ATP (see 2.1, "Auto Test Performer" on page 9), except that it runs under Windows 95 and Windows NT, instead of OS/2.

Note that WITT Year2000 for Windows captures only graphical images of screens. It does not capture character versions, and so the images cannot be compared using SuperC.

2.10.1 Hardware Requirements

- Processor: i486 100MHz minimum (Pentium 90MHz or higher recommended)
- Display: VGA minimum (SVGA recommended)
- RAM:
  - Windows 95 - 24MB minimum
  - Windows NT - 32MB minimum

  Memory to run testing application is needed in addition to above requirements.
- Disk Space:
  - 30MB (excluding prerequisite software and swap area)

2.10.2 Software Requirements

- Microsoft Windows 95 with Service Pack 1, or Microsoft Windows NT4.0 with Service Pack 2
- To use 3270/5250 capability, at least one of the following:
  - For Windows 95, Personal Communications AS/400 and 3270 for Windows 95, Version 4.1 + CSD2
  - For Windows NT, Personal Communications AS/400 and 3270 for Windows NT, Version 4.1 plus fixes for following APARs:

    - IC16625
    - IC16624
    - IC17453
- To access the online User′s Guide and Tutorial, an HTML browser such as Netscape Navigator 3.0 or Internet Explorer 3.0 is needed.

2.10.3 Installation and Usage Hints

The file `hcsehlnt.exe`, which provides the fixes for the NT APARs listed above, is available from:

`ftp://ps.software.ibm.com/ps/products/pcom/fixes/v4.1x/winnt/pcsehlnt/`

For online information about WITT Year2000 for Windows, double-click the *User′s Guide* in the IBM WITT Year2000 for Windows group.

Chapter 3. Installation and Customization

The files required to run the tutorials in this book are on the CD-ROM provided with the book.

This chapter tells you how to install these files on your host system and workstation, and how to customize them to your installation′s requirements.

There are instructions for installing files to a host running under OS/390 and VSE, and from a workstation running Windows or OS/2. Select the instructions appropriate for your installation.

The instruction assume that the CD-ROM is in drive e. Change where necessary.

3.1 Prerequisite Software

See documentation provided with the following products for installation and customization information.

3.1.1 OS/390

The following software was used on the OS/390 system to produce the tutorials:

- Customer Information Control System (CICS), Version 5 Release 1
- Language Environment (LE), Version 1 Release 5
- Debug Tool (DT), Version 1 Release 2
- Enhanced SuperC (High Level Assembler Tool Kit Feature, Version 1 Release 2 + PTF UQ09985)
- DFSORT Version 1 Release 13 + PTF UQ00530
- Interactive System Productivity Facility (ISPF), Version 4 Release 4

It is recommended that your system have these products at these levels, or later.

3.1.2 VSE

The following software was used on the VSE system to produce the tutorials:

- VSE/ESA, Version 2 Release 2
- Customer Information Control System (CICS), Version 2 Release 3
- Language Environment (LE/VSE) Version 1 Release 4
- Debug Tool (DT/VSE), Version 1 Release 1
- Enhanced SuperC (High Level Assembler Tool Kit feature, Version 1 Release 2 + PTF UQ09948)
- DFSORT/VSE Version 3 Release 3

It is recommended that your system have these products at these levels, or later.

3.1.3 Workstation

The following software is required on the workstation to support the tutorials:

- VisualAge Test for OS/2, Version 1 (or later) **or**
- WITT Year2000 for OS/2, Version 3 (or later) **or**
- WITT Year2000 for Windows NT and Windows 95, Version 1

The following software is required on the workstation to view this book online:

- Library Reader/2
  - Provided on the CD-ROM accompanying this book—see README.TXT
- Adobe Acrobat Reader
  - Available for downloading from Adobe Systems:

    `http://www.adobe.com/prodindex/acrobat/main.html`

Host emulation software is also required. The following instructions are based on using *Personal Communications AS/400 and 3270 for OS/2 V4.1 (PC/3270)* for OS/2, or *Personal Communications AS/400 and 3270 for Windows V3.1*, or an equivalent.

3.2 Structure of CD-ROM

The CD-ROM has been created in a form which can be read by OS/2 and Windows. The directory structure is:

README.TXT Supplementary information; please read first.

DEMO Demonstration “slide-show,” that runs under Windows:

ABOUT.DMR	Information about the demos
ATP1.DMR	Auto Test Performer demo, part 1
ATP2.DMR	Auto Test Performer demo, part 2
CICSA.DMR	Auto Test Performer demo, part 3
CICSW.DMR	WITT Year2000 for Windows demo, part 3
COBT1.DMR	COBOL Tester demo
OVER1.DMR	Overview of all the tools
OVER2.DMR	Overview of the testing process
USING.DMR	Instructions on how to use the demos
VA2000T.DMR	The main menu
VA2000T.EXE	The demonstration player
WITT1.DMR	WITT Year2000 for Windows demo, part 1
WITT2.DMR	WITT Year2000 for Windows demo, part 2

For instructions on running the demonstration slide-show, see Appendix C, “The Slide-Show Demonstrations” on page 143.

DOC This book:

SG242230.BOO	For BookManager Read
SG242230.PDF	For Adobe Acrobat Reader

OS390	The data sets for uploading to OS/390:	
	ATC.PDS	Application Testing Collection (ATC) batch jobs and source code
	CLIST.PDS	CLISTs and REXX procedures
	CNTL.PDS	JCL
	COBOL.PDS	COBOL source of example programs
	DATA.PDS	Data required by programs or procedures
	DEBUG.PDS	Debug Tool command scripts
	DEPT.DAT	Department Master data file
	EMP.DAT	Employee Master data file
	LISTING.PDS	Debug Tool listings and ATC report samples
	LOAD.PDS	Load modules for example programs
	PLI.PDS	PL/I source of example programs
	SENDPDS.BAT	Windows BAT file to send PDS files to OS/390 platform
	TARSEND.CMD	REXX procedure to send PDS files to OS/390 platform
OS2	The data sets required on the workstation:	
	TARADJST.CMD	REXX procedure to convert and age ATP scripts
	TARDTE3.COB	COBOL program used as COBOL Tester example program
READ2	Library Reader/2	
VSE	The sample JCL, source and phases for uploading to VSE:	
	VSELOAD1.Z	Sample JCL, source, and data members
	VSELOAD2.Z	Phases
WINDOWS	The data set required on a Windows workstation:	
	TARADJST.TSC	LotusScript procedure to convert and age WITT Year2000 for Windows scripts

3.3 Installing Files on OS/2

If you intend to work through the COBOL Tester tutorial, or if you intend to use Auto Test Performer to create a script, copy the OS/2 files into directories on your workstation, using the following commands:

```
e:
cd \os2
d:
md va
cd va
copy e:taradjst.cmd
md vatest
cd vatest
copy e:tardte3.cob
```

This assumes a hard disk drive d.

3.4 Installing Files on Windows 95 or Windows NT

If you intend to use WITT Year2000 for Windows to create a script, copy the Windows file into a directory on your workstation, using the following commands:

```
e:
cd \windows
d:
md va
cd va
copy e:taradjst.tsc
```

This assumes a hard disk drive d.

3.5 Loading OS/390 Files

There are many ways of transferring and loading files to OS/390 systems. This section shows two ways; one from OS/2 and one from Windows. Each way assumes that you are logged onto TSO at the READY prompt. The examples assume that you are on TSO session f.

If you prefer to use an alternative method to transfer the files, please do so.

3.5.1 Partitioned Data Sets

Make sure you are logged on to TSO at the READY prompt.

3.5.1.1 Sending from OS/2

1. On the workstation enter:

   ```
   e:
   cd \os390
   tarsend
   ```

   The message `Please enter os390 session id or 999 to exit` appears.

2. Enter the ID of the session (for example f).

   The message

   ```
   Sending file send atc.pds f:atc
   Press any key when ready . . .
   ```

   appears.

3. Press a key.

   The message `The file transfer request is being processed.` appears.

   The number of bytes being transferred is displayed and updated as the file transfer takes place.

   The message `File transfer is complete.` appears once the transfer is finished.

   This process repeats for the remaining files:

- CLIST
- CNTL
- COBOL
- DATA
- DEBUG
- LISTING
- LOAD
- PLI

When all nine files have been sent, the message

```
File Transfer Completed
```

appears.

Go to 3.5.1.3, "Receiving the files in TSO" to receive the files into TSO.

3.5.1.2 Sending from Windows

1. Open up a command line or a DOS window.
2. Enter:

   ```
   e:
   cd \os390
   sendpds f
   ```

   where `f` is the OS/390 session id to send the files to.

   This procedure sends nine PDS files.

Go to 3.5.1.3, "Receiving the files in TSO" to receive the files.

3.5.1.3 Receiving the files in TSO

1. On the TSO command line enter:

   ```
   receive inds(clist)
   ```

   A message such as

   ```
   INMR901I Dataset VA2000TS.CLIST from ROSSB on PTHMVS4
   INMR906A Enter restore parameters or 'DELETE' or 'END' +
   ```

   appears.

2. You must now enter parameters to restore the PDS with the name and attributes that fit your installation's standards. If your high-level qualifier is the same as your login ID, press Enter. Otherwise, enter the full PDS. For example, to restore to a PDS using another chosen high-level qualifier:

   `dsn('`*`hlq`*`.va2000ts.clist')`

   For more information on the use of the `receive` command, in ISPF enter `TSO HELP RECEIVE`.

   You have now restored the CLISTs. You can now use the TARREC procedure to receive the other data sets.

3. On the TSO command line enter:

   ```
   altlib act app(clist) dsn('hlq.va2000ts.clist')
   tarrec
   ```

   (Note altlib includes the `clist` library in your procedure library concatenation—see page 25.)

   This procedure asks for the high level qualifier.

4. Press Enter if the user ID is the high level qualifier, or enter the high level qualifier.

   The proposed high level qualifier appears.

5. Enter 1 to accept.

   The remaining eight PDS files are restored. The end of each restore is flagged with File received. The restored files are:

   - *hlq*.VA2000TS.ATC
   - *hlq*.VA2000TS.CLIST
   - *hlq*.VA2000TS.CNTL
   - *hlq*.VA2000TS.COBOL
   - *hlq*.VA2000TS.DATA
   - *hlq*.VA2000TS.DEBUG
   - *hlq*.VA2000TS.LISTING
   - *hlq*.VA2000TS.LOAD
   - *hlq*.VA2000TS.PLI

3.5.2 Sequential Data Sets

Make sure you are still logged on to TSO at the READY prompt.

On the workstation (OS/2 or Windows) enter:

```
E:
cd \os390
send dept.dat   f:'hlq.dept.data'
send emp.dat    f:'hlq.emp.data' lrecl(200)
```

This creates data sets named *hlq*.dept.data, and *hlq*.emp.data. These provide the data for the initial VSAM data sets used by the tutorials.

3.5.3 Customizing Files for OS/390

Every installation has its own standards for data set names, job cards and CICS transactions.

The TARCNTL procedure applies a base level of customization to jobs that are run during the tutorials. Before running this procedure, make sure that you know:

- The high level qualifier
- The location of Enhanced SuperC

1. On the TSO command line enter:

   ```
   altlib act app(clist) dsn('hlq.va2000ts.clist')
   tarcntl
   ```

   The procedure asks for four values.

2. For the file to be edited, the high level qualifier, and the job name prefix, the procedure displays the default value which is used if you press Enter. Either accept the default value by pressing Enter, or enter a different value.

3. For the Enhanced SuperC Job library and whether COBOL or PL/I, you must type a value.

4. If you wish to terminate the procedure, you can enter 999 at any step.

5. After the details you have entered appear, enter 1 to confirm.

   The procedure lists each member as it is customized.

The message `ALL MEMBERS HAVE BEEN UPDATED` indicates that the procedure has finished.

The partitioned data sets which you installed in 3.5, "Loading OS/390 Files" on page 22 assume certain things:

- Unchanged second and subsequent data set name qualifiers
- Simple job cards without installation-specific information
- DFSORT in a linklist library (if not, JOBLIB or STEPLIB will need to be added)
- Program names of the form `TARMUx`
- CICS transactions of the form `NBxx` (for COBOL programs) or `MBxx` (for PL/I programs)

You will need to make changes to these if they do not fit your installation's standards.

3.5.4 Initializing for OS/390

Before you can run the tutorials you need to:

- Allocate and initialize the VSAM data sets required
- Allocate the report data sets required
- Define transactions, programs and files to CICS
- Add *hlq*.VA2000TS.LOAD to the CICS DFHRPL DD statement

  Note: SCEECICS and SCEERUN Language Environment libraries also need to be added if they are not there already.

You may find it convenient to add the *hlq*.VA2000TS.CLIST to your SYSPROC or SYSUPROC concatenation in your TSO logon procedure. If you prefer to not do this, then at the start of each TSO session you should, on the TSO command line, enter:

```
altlib act app(clist) dsn('hlq.va2000ts.clist')
```

When you use one of these methods, then you can invoke CLISTs and REXX procedures by entering the member name only, that is `rexxname` rather than `ex 'hlq.va2000ts.clist(rexxname)'`. The rest of this book assumes the entry of only the member name.

3.5.4.1 Allocating and Initializing VSAM Data Sets

The initial VSAM data sets for the tutorials are quite small. The Employee Master file has 13 records (200 bytes each) and the Department Master file has 11 records (80 bytes each).

1. On the option line, running under ISPF, enter:

   ```
   tardvsam
   ```

   The message `Please enter data set name high-level qualifier` appears.

2. Enter your chosen data set name high-level qualifier. .

   The procedure then creates VSAM data sets:

   ```
   hlq.DEPT.MASTER.ONLINE
   hlq.EMP.MASTER.START
   ```

3.5.4.2 Allocating Non-VSAM Data Sets

This step allocates the small sequential data sets to hold reports and log files generated by, and used in, the tutorials.

1. On the option line, running under ISPF, enter:

   ```
   tardnvsm
   ```

   The message:

   ```
   Current data set name high-level qualifier is: hlq
   Press Enter (with no data) to accept this,
   or enter new high-level qualifier and then press Enter
   ```

   appears.

2. Press Enter, or enter a different high-level qualifier.

   The procedure then creates sequential data sets:

   ```
   hlq.TARRPA1.REPORT
   hlq.TARRPA2.REPORT
   hlq.TARRPA3.REPORT
   hlq.TARRPA4.REPORT
   hlq.TARMU3.LOG
   hlq.TARMU3.LOG2
   hlq.TARMU3X.LOG
   hlq.PL1MU3.LOG
   hlq.PL1MU3X.LOG
   ```

3.5.4.3 CICS Definitions

Sample CICS definitions (for programs, files, transactions) are supplied in *hlq*.VA2000TS.DATA(CICSDEF); sample JCL to apply these is in *hlq*.VA2000TS.CNTL(CICSDEF).

Note: These will need to be tailored to meet your installation's standards.

3.5.4.4 CICS Startup JCL

The following is an example of the statements that need to be added to the started procedure for the CICS system running VisualAge 2000 Test Solution:

```
⋮
//DFHRPL   DD DSN=...
⋮
//         DD DSN=...SCEECICS,DISP=SHR            <== LE CICS
//         DD DSN=...SCEERUN,DISP=SHR             <== LE
//         DD DSN=...EQAWV120.SEQAMOD,DISP=SHR    <== DEBUG TOOL
//         DD DSN=hlq.VA2000TS.LOAD,DISP=SHR        <== VA2000TS
```

Note: This will need to be tailored to meet your installation's standards.

3.5.5 Setting Up for Debug Tool

Debug Tool script files use compiler listings and log files. These members must be adjusted, to use the appropriate high level qualifier. The REXX procedure **tardbg** does the adjusting. The procedure asks for two items of information: the name of the PDS that needs to be adjusted, and the high level qualifier pointing to where the compiler listings and log files are stored. Make sure you know these two items before you run the procedure.

1. On the TSO command line, enter:

   ```
   tardbg
   ```

An explanatory message appears. Then the message:

```
Optionally enter name of file containing the
members to be globally edited by this EXEC. Enter 999 to exit at any time.
Default file =  hlq.VA2000TS.DEBUG
```

appears.

2. Press Enter to accept the default name, or enter the full name of the file to be adjusted.

   The response:

   ```
   Default value used
   ```

   appears. Then the message:

   ```
   Optionally enter high-level qualifier or 999 to exit
   Default hlq =  hlq
   ```

   appears.

3. Press Enter to accept the default high level qualifier, or enter a different high level qualifier.

   The response:

   ```
   Updating members in hlq.VA2000TS.DEBUG
     High level qualifier =  hlq
   ```

   appears. Then the confirmatory message appears.

4. Enter 1 to accept the details.

   The list of updated members appears.

The Debug Tool script files are now customized.

3.5.6 Conventions in This Redbook for OS/390

- When the text says “Execute job TARABC,” then you should submit member TARABC from *hlq*.VA2000OTS.CNTL, where *hlq* is the high level qualifier you are using.
- When the text says “Execute Rexx procedure TARABC,” then you should run REXX procedure TARABC from *hlq*.VA2000OTS.CLIST.
- When the text says “Check the report from the job, by browsing TARABC.REPORT,” then you should browse the report *hlq*.TARABC.REPORT, through ISPF or an equivalent facility.
- When reports are mentioned, prefix their name with *hlq*.
- The high level qualifier, *hlq*, takes the value you provide at installation.

3.6 Loading VSE Files

The following steps show how to:

- Upload the files to VSE.
- Process the files to produce individual members in the selected library.

3.6.1 Uploading Files from a Windows Platform

1. Make sure that:

   - Windows NT or Windows 95 is up and running.
   - A host emulator session is started.
   - The host emulator is configured for file transfer (for example, *IBM Personal Communication/3270 for Windows V3.1*):

     – From the **File** menu, select **API settings**. Ensure that the **DOS-Mode EHLLAPI** is ON.
     – From the **Transfer** menu, select the transfer type of your host operating system.
     – From the **Transfer** menu select **Setup - Miscellaneous Settings**. Check that the correct PC code page is specified (in most cases the international code page 850 is satisfactory).

2. Logon to the VSE/ESA host system to place the session into VSE file transfer mode.

3. Select fast path 596 if you have programmer authority (an administrator would select fast path 386).

   This CICS session should be defined as a DFT terminal—the upload does not work if it is defined as a CUT terminal. If you are unsure, run the *INWQ* transaction which indicates what type of terminal it is. If you do not know how to select a DFT terminal, see your systems programmer.

4. Open a command-line window.

5. Enter:

```
e:
cd \vse
send vseload1.z <S>:vseload1 Z
     (file=lib l=<LIB> s=<SUBLIB> binary nocrlf lrecl=80
send vseload2.z <S>:vseload2 Z
     (file=lib l=<LIB> s=<SUBLIB> binary nocrlf lrecl=80
```

   where:

e	is the CD-ROM drive.
<S>	is the emulator session you are uploading to.
<LIB>	is the name of the library where the two uploaded files are to go.
<SUBLIB>	is the name of the sublibrary where the two uploaded files are to go.

   For example: To upload two files to host session *A* into *PRD2.PROD* enter:

```
SEND VSELOAD1.Z A:VSELOAD1 Z
   (FILE=LIB L=PRD2 S=PROD BINARY NOCRLF LRECL=80

SEND VSELOAD2.Z A:VSELOAD2 Z
   (FILE=LIB L=PRD2 S=PROD BINARY NOCRLF LRECL=80
```

Go to 3.6.3, “Processing Uploaded Files” on page 30 to process these files on the host.

3.6.2 Uploading Files from an OS/2 Platform

1. Make sure that:

   - A host emulator session is started. It must be configured via attachment IEEE802.2, not TCP/IP.
   - The host emulator is configured for file transfer. The following setup refers to *IBM Personal Communications AS/400 and 3270 for OS/2 V4.1 (PC/3270)*:

     – From the **File** menu, select **API settings**. Ensure that the **DOS-Mode EHLLAPI** is ON.

     – From the **Transfer** menu, select the transfer type of your host operating system.

     – From the **Transfer** menu, select **Setup - Miscellaneous Settings**. Check that the correct PC code page is specified (in most cases the international code page 850 is satisfactory).

2. Logon to the VSE/ESA host system to place the session into VSE file transfer mode.

3. Select fast path 596 if you have programmer authority (an administrator would select fast path 386)

   This CICS session should be defined as a DFT terminal—the upload will not work if it is defined as a CUT terminal. If you are unsure, run the *INWQ* transaction which will indicate what type of terminal it is. If you do not know how to select a DFT terminal, see your systems programmer.

4. Open a command-line window (an OS/2 prompt window).

5. Enter:

```
 e:
 cd \vse
 send vseload1.z <S>:vseload1 Z
      (file=lib l=<LIB> s=<SUBLIB> binary nocrlf lrecl=80
 send vseload2.z <S>:vseload2 Z
      (file=lib l=<LIB> s=<SUBLIB> binary nocrlf lrecl=80
```

   where:

e	is the CD-ROM drive.
<S>	is the emulator session you are uploading to.
<LIB>	is the name of the library where the two uploaded files are to go.
<SUBLIB>	is the name of the sublibrary where the two uploaded files are to go.

   For example: To upload two files to host session *A* into *PRD2.PROD* enter:

```
SEND VSELOAD1.Z A:VSELOAD1 Z
   (FILE=LIB L=PRD2 S=PROD BINARY NOCRLF LRECL=80

SEND VSELOAD2.Z A:VSELOAD2 Z
   (FILE=LIB L=PRD2 S=PROD BINARY NOCRLF LRECL=80
```

Go to 3.6.3, “Processing Uploaded Files” on page 30 to process these files on the host.

3.6.3 Processing Uploaded Files

Each of the uploaded files must be processed to create the individual members contained within them. To do this:

1. Punch VSELOAD1.Z from the library in which it is installed. This can be achieved by:
    a. Librarian PUNCH command from a batch LIBR job. Add the FORMAT=NOHEADER operand to the PUNCH statement to suppress the punching of CATALOG and EOD commands. This could be used to place the file in a CMS environment for editing.
    b. Using the ICCF LIBRP macro to place the file in an ICCF library or punch area for editing.
2. Edit the file to:
    - Add VSE/POWER statements, if required.
    - Change the ACCESS statement to point to the required library and sublibrary.
3. Submit the job for processing. This will create all the JCL, source and data members.
4. Punch VSELOAD2.Z from the library in which it is installed. This can be achieved by:
    a. Librarian PUNCH command from a batch LIBR job. Add the FORMAT=NOHEADER operand to the PUNCH statement to suppress the punching of CATALOG and EOD commands. This could be used to place the file in a CMS environment for editing.
    b. Using the ICCF LIBRP macro to place the file in an ICCF library or punch area for editing.
5. Edit the file to:
    - Add VSE/POWER statements, if required.
    - Change the SETPARM statement to point to the required library and sublibrary.
6. Submit the job for processing. This will linkedit the phases into the indicated sublibrary.

3.6.4 Customizing Batch Jobs

The tutorials include the running of batch jobs. These jobs must be customized to reflect the correct user libraries, disk volumes, and starting track numbers. Also, you have to specify if you are running COBOL or PL/I.

Member RTARCHG.Z is a sample job stream that customizes the JCL run as part of the tutorials. This batch job executes two REXX procedures:

TARCHG	Uses parameters read via sysipt to update specific batch jobs. User library, disk volumes and starting track numbers are set up here.
TARCHG2	Updates specific batch jobs to execute either COBOL or PL/I programs

1. Punch the member RTARCHG.Z from the installation sublibrary and edit it so that it reflects how you want the batch jobs set up:
    - Replace ′COMMON.VA2000′ userlib on // SETPARM statement.

   - Update the ′replacement value′ portion of the TARCHG parameters (such as COMMON.VA2000, SYSWK1, SYSWK2, 3000, 3005).
   - Set parameter 2 of TARCHG2 to either COBOL or PL1.

2. Submit RTARCHG.Z for processing.

   Members that are updated are listed. RTARCGG.Z updates members in the installation sublibrary itself.

3. Check that all values have been correctly updated.

3.6.5 Initializing for VSE

Before you can run the tutorials you need to:

- Allocate and initialize the VSAM data sets required
- Define transactions, programs and files to CICS

3.6.5.1 Allocating and Initializing VSAM Files

1. Punch TARDVSAM.Z from the installation sublibrary and submit.

   This defines and initializes three files.

   The default LE/VSE library is PRD2.SCEEBASE. Change this, if necessary, before submitting.

3.6.5.2 CICS Definitions

1. Punch TARCSD.Z from the installation sublibrary and submit.

   This job defines transactions and programs for CICS processing.

2. You need to add five files to your FCT (File Control Table). FCT definitions for these files are supplied in TARFCT.A. Include this member in your FCT.

3. If necessary, install the group VA2000, with the command

   ```
   CEDA I G(VA2000)
   ```

3.6.5.3 CICS Startup JCL

You need to add DLBL statements for the five files to your CICS startup JCL. These are:

```
// DLBL IJSYSUC,′VSESP.USER.CATALOG′,,VSAM
// DLBL EMPMAST,′VA2000.EMP.MASTER.ONLINE′,,VSAM
// DLBL EMPMST2,′VA2000.EMP.MASTER.ONLINE2′,,VSAM
// DLBL EMPMST3,′VA2000.EMP.MASTER.ONLINE3′,,VSAM
// DLBL EMPMST4,′VA2000.EMP.MASTER.ONLINE4′,,VSAM
// DLBL DEPMAST,′VA2000.DEPT.MASTER.ONLINE′,,VSAM
```

Multiple versions of the Employee Master file (EMP.MASTER.ONLINE) are created during the tutorials. This eases the flow through the tutorials.

You should perform a CICS cold start to activate VA2000.

3.6.6 Conventions in This Redbook for VSE

- When the text says “Execute job TARABC,” then you should punch TARABC.Z from the installation sublibrary, and submit.
- When the text says “Execute Rexx procedure TARABC,” then you should punch TARABC.Z from the installation sublibrary, and submit.

- When the text says “Check the report from the job, by browsing TARABC.REPORT,” then you should browse the report *hlq*.TARABC.REPORT.
- When reports are mentioned, prefix their name with *hlq*.
- The high level qualifier, *hlq*, takes the value VA2000.

Part 2. The Test Tutorials

Chapter 4. Introducing The Tutorials

The following tutorials work through the steps in the testing methodology as outlined in Figure 1 on page 4.

The tutorials are based on a common application (described later).

4.1 An Outline of the Tutorials

The first tutorial, Chapter 5, “Developing Path Coverage and Distilling the Testbed” on page 37, is a stand alone tutorial that you can work through at any time.

The second and subsequent tutorials, from Chapter 6, “Creating the Baseline Results” on page 69, to Chapter 15, “Comparing the 19xx Results with the 20xx Results” on page 123, form a series. You must start with the first tutorial in the series, and work through the tutorials in order. Part of the output of a tutorial is often used in one or more of the following tutorials.

Backup checkpoints are built into the tutorials so that if a process fails, it is possible for you to restore from backup, and then continue, without needing to start the process from the second tutorial.

Each tutorial starts with a summary, followed by a detailed outline of the procedures followed in the tutorial process. After this introductory material, the tutorial lists each step in the procedure.

There may be more than one procedure in a tutorial. For example, when Auto Test Performer is used for screen capture and script creation, the tutorial also includes instructions on backing up the distilled master file, starting a CICS session, starting ATP, entering the CICS transactions, and so on.

To work through a procedure, the user should be competent in the standard applications. For example, the user who enters CICS transactions should know how to invoke a CICS program, and how to enter information.

The interactive tutorials have a lot of steps. Some of the other tutorials require merely the submission of a few jobs. However, you may benefit by taking time to look at the jobs. From them you may learn approaches that help you with your testing.

Each tutorial assumes that the tools have been installed on the workstation or host, as required. For an introduction to each tool, and pointers on how to install them and the example programs, see Chapter 2, “The Tools Used by VisualAge 2000 Test Solution” on page 9.

For script recording and playback, and screen image capture and comparison, there are two alternatives: Auto Test Performer and WITT Year2000 for Windows. The following tutorials perform these functions using Auto Test Performer. The equivalent actions, as carried out using WITT Year2000 for Windows, are provided in Appendix B, “Using WITT Year2000 for Windows” on page 133.

4.2 An Outline of the Sample Application

The application around which the tutorials are built is a simple master file update.

The main file is the Employee Master file. This file holds basic information about employees:

- Employee name
- Department in which the employee works
- Employee address details
- Dates:
  - When the employee joined the organization
  - When the employee left the organization (was terminated)
  - When the employee was born (birth date)
  - When the employee′s security pass expires (expiry date)

This information is maintained through a CICS interactive program.

As well, the system includes three batch programs:

- A report program that lists all employee details.
- A report program that lists in expiry date order the security pass expiry dates for employees still employed. This particular program does not need to be converted to handle Year 2000 dates. The required bridging can be handled by adjusting the JCL.
- A batch program that updates the security pass expiry date.

The application also includes a Department Master file, used for reference and not updated.

A set of data files is provided to start the process. A job copies the Employee Master file from a source file, so that the tutorials can be worked through as often as you want, without needing to reload any files.

This sample application is fairly simple, by design. The intention is to allow you to focus on the testing steps and procedures, and so understand what is involved in Year 2000 testing.

The COBOL and PL/I source provided are not required for running the tutorials. Once again, you may find that the source provides useful techniques to apply to your applications.

Chapter 5. Developing Path Coverage and Distilling the Testbed

The aim of this process is to develop a testbed of transactions and master file records that tests every sub-path in the program, but contains no superfluous test transactions.

Input	Original program code, original testbed (or cut-down production data)
Output	Distilled testbed

You may not have set aside any data specifically for testing. In this case, the easiest way to create a starting point is to extract data from a production system. The first step in the testing process refines and expands this data to overcome its inadequacies as a proper testbed.

An alternative approach may be to start with no data, and use a path coverage tool to build up a testbed.

The purpose of this process is to create a distilled testbed. A *distilled* testbed is a small testbed, where the master files are as small as possible, and the number of transactions has been reduced to a minimum (commensurate with the high-risk or low-risk testing strategy that you are adopting).

The reason for producing a distilled testbed is that the testbed and variations of the testbed are used at least four times during Year 2000 testing. This means that the testbed should be as small as possible.

For *high-risk* (critical) applications, you must include test cases to cover every path in the code, whether the path includes a Year 2000 conversion or not. The resulting testbed is substantially larger, because of the comprehensive testing required. If you are running out of time or resources, and so are unable to develop 100% coverage, you may prefer to develop “targeted coverage” using a tool such as Source Audit Assistant to focus on the lines of code that are changed during conversion.

For *low-risk* applications, you may seek to cover only those paths in the code that include a Year 2000 conversion. This limits the size of the testbed to what is practical and necessary. Targeted coverage will identify these paths.

5.1 Paths, Nodes and Sub-paths

To understand path coverage, you must know about paths, nodes and sub-paths.

A path in code is the lines of code that are used to process an individual data record. Each path is made up of *sub-paths* and *nodes*.

A node is a decision point, followed by two or more sub-paths. A particular record takes a particular sub-path, depending on the values in the record.

A sub-path is the executable code between two nodes. Figure 3 on page 38 shows code containing a node and two sub-paths.

```
Node           IF MESSAGE = SPACES THEN
Sub-path-1A       MOVE DATE-YYMMDD TO TERM-DATE
               ELSE
Sub-path-1B       MOVE 6 TO WORK-ERROR-CODE
Sub-path-1B       SET ERRORS TO TRUE
               END-IF
```

Figure 3. Nodes and Sub-paths

The first sub-path is “Sub-path-1A,” and is one line of code. The second sub-path is “Sub-path-1B,” and is two lines of code. The only node in the code is the first line. The “ELSE” and “END-IF” lines are not executable code, and so are not part of the sub-paths. These lines delimit the sub-paths following the node.

When nodes are nested, they produce further sub-paths (see Figure 4).

```
Node 1         IF MESSAGE = SPACES THEN
Sub-path-1A       MOVE DATE-YYDDMM TO TERM-DATE
Node 11           IF  TERM-DATE(1:2) = 00
Sub-path-11A          MOVE 1 TO CENTURY-IND
                  ELSE
Sub-path-11B          MOVE 0 TO CENTURY-IND
                  END-IF
               ELSE
Sub-path-1B       MOVE 6 TO WORK-ERROR-CODE
Sub-path-1B       SET ERRORS TO TRUE
               END-IF
```

Figure 4. Nested Nodes and Sub-paths

There are potentially many different paths within a program. For example, if the program has ten independent two-way decision points (nodes), then there are 1024 different paths through the program. A complete test would thus require over a thousand test transactions. And if the program has 20 two-way nodes, then the complete test requires more than a million test transactions. As a program grows in complexity, it becomes impossible to provide complete testing.

However, to cover all the sub-paths, both programs (one with ten nodes, and one with 20) need only two test transactions, if the data is set up so that each choice is taken at a node. And even when nested nodes provide more choices, three or four test transactions may still be enough.

5.2 The Path Coverage Process

Figure 5 on page 39 shows the cyclical nature of the coverage process.

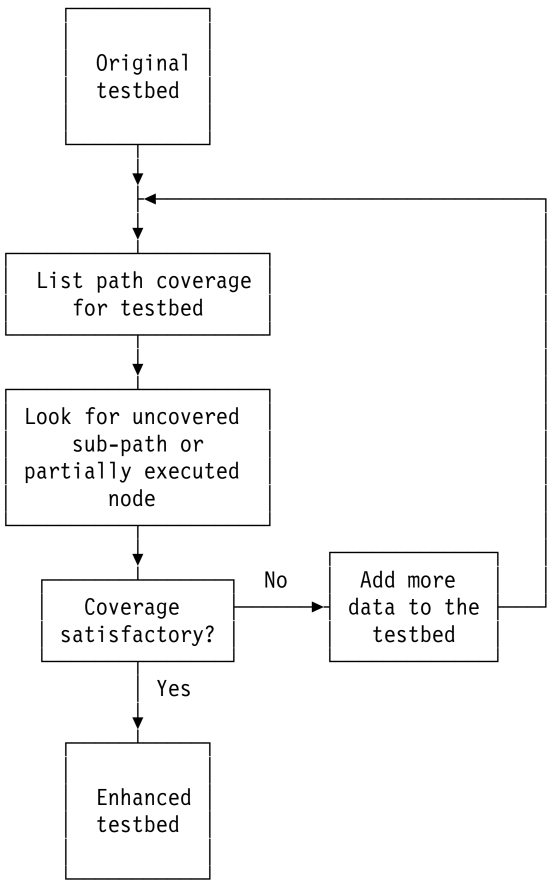

Original testbed
List path coverage for testbed
Look for uncovered sub-path or partially executed node
Coverage satisfactory?
No
Add more data to the testbed
Yes
Enhanced testbed

Figure 5. The Coverage Process

Starting with the original testbed, you list the coverage (using the appropriate tool). You then examine this coverage. If it is satisfactory, the process is finished, and the testbed is the enhanced testbed.

If the coverage is not satisfactory, then you must add to the testbed. By looking at the nodes preceding the sub-paths that are not covered, you should be able to develop test data to cover the sub-paths. After you have added to the testbed, you then repeat the process.

If you are testing a high-risk application, then the aim is 100% coverage. Note that you may find that you cannot gain 100% coverage. This may be because the node is checking for conditions that are very difficult or impossible to force using normal test data. For example, you may be testing for failure to open a file, when the file can be opened except for extraordinary conditions. In this case, you can do nothing but accept that it is not possible to cover the code. Also, you may also find that code is not covered because it is logically impossible to cover it—maybe the code is a procedure that is never called. In this case you should remove the code. Leaving it could later lead to confusion and waste time, especially if someone puts in effort applying a Year 2000 conversion to it.

If you are testing a low-risk application, the aim is to cover every sub-path that includes Year 2000 code, and every node that tests a Year 2000 condition. The coverage may be only 30%, but you may consider this to be satisfactory.

5.2.1 Targeted coverage

In theory, the maximum coverage should be 100%. However, resource constraints may make this impracticable. You may have the choice of maximum coverage for some programs and no coverage for others, or some coverage for every program.

The second choice is the best, if you can make the coverage as significant as possible. In particular, you want to provide coverage for all of the paths through code that handle dates, and are being (or have been) converted.

A quick way of identifying this code is to use a tool such as Source Audit Assistant. Source Audit Assistant is a part of the Application Testing Collection. It compares two program sources, and reports on lines that have been inserted, deleted or changed; precisely the lines that need coverage.

Source Audit Assistant is also able to identify date fields that were not part of a change to the source code. This provides a means of checking that all required changes have been made.

5.3 The Distillation Process

After you have completed path coverage, you can be confident that your code is being adequately exercised. However, the resultant testbed might be quite large.

It is desirable to use the smallest possible testbed, since such a testbed requires less storage, uses less computer time for processing, generates fewer records for comparisons, and requires less effort for converting and aging.

The aim is to produce a testbed that includes no redundant records. This process, called distillation, is shown in Figure 6 on page 41.

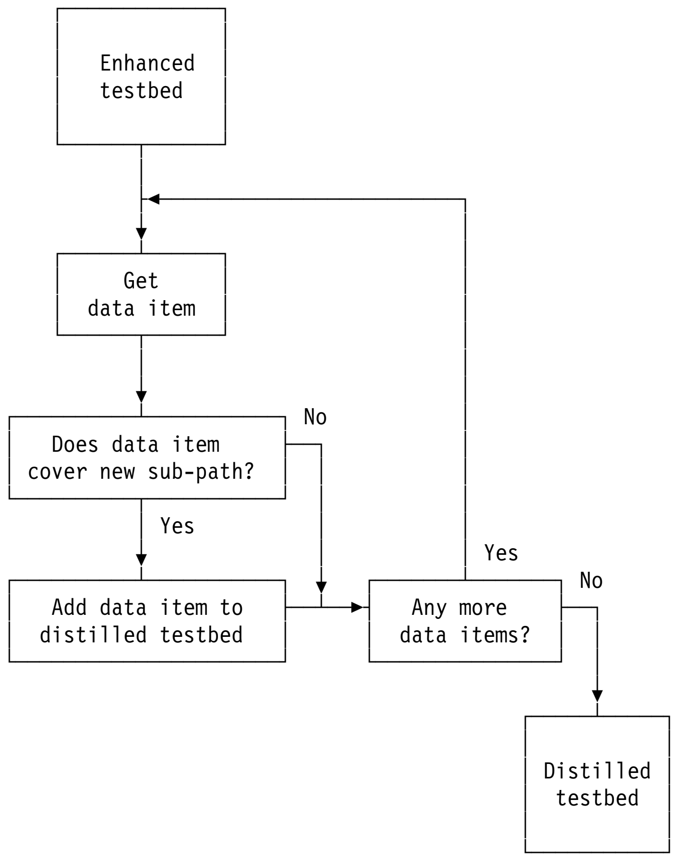


Figure 6. The Distillation Process

This figure is different to Figure 5 on page 39 in that the decisions are made within the distillation process. You do not have to make any decisions during distillation; you just run the distillation utility.

In effect, the distillation utility checks the path coverage after each data item has been processed, and discards all data items which do not add to the coverage.

The distillation process may, under unusual circumstances, remove data items that are required for complete coverage. To eliminate this possibility, you may consider running the distilled testbed through the coverage utility, and adding any data items needed to ensure that coverage is satisfactory.

If you are doing performance testing, you may wish to develop two testbeds. One is the distilled version, which uses the smallest number of transactions, for speed of testing. The other is the performance testing version, which includes a lot of redundant data, but puts the program under a measurable load.

5.4 Path Coverage and Data Distillation - COBOL Tester

This tutorial shows you how to use COBOL Tester to develop path coverage in a sample program. VisualAge for COBOL, Test for OS/2, COBOL Tester (COBOL Tester) works by building up coverage from nothing. Provided each transaction you add increases the coverage, the final set of transactions is a distilled set of transactions.

The sample program used in the tutorial is "TARDTE3," a utility program that converts a date from one format to another. This program accepts the input date, the input date format, and the output date format as parameters. If the parameters are valid, the program returns the date in the requested output format. If the parameters are not valid, the program returns an error message.

COBOL Tester is appropriate to use on original code. It may not be suitable for use on converted code, since it does not support testing of items such as object-oriented features, or Millennium Language Extensions.

5.4.1 Starting COBOL Tester

1. Double-click the VisualAge Test folder icon.
2. Double-click the COBOL Tester folder icon.
3. Double-click the COBOL Tester icon.

   The **COBOL Tester: Testcase** window appears.

5.4.2 Creating a Testcase

A testcase links a COBOL source program with a series of scripts. Each script is one pass of the program, providing input data, and producing output data. The output data can be checked against expected output data.

One script is unlikely to cover the entire program. You can create a number of scripts, so that among them they cover the program.

Before you create scripts, you have to create a testcase.

1. From the **Testcase** menu, select **New...**

   The **New** dialog box appears.
2. Click **File list...**

   The **COBOL Source File List** dialog box appears.
3. Search through the directories and find the TARDTE3.COB program. (If the files were installed as suggested in the previous section, this file should be in the `vavatest` directory.)
4. Click on TARDTE3.COB to highlight it.
5. Click **OK**.
6. Click **New**.

   The **.Untitled - Source Analysis** box appears. This shows messages about COBOL source analysis, logical path analysis, and so on.
7. When the message "The analysis process is successful" appears, click **OK**.

5.4.3 Saving the Testcase

The testcase file is not yet saved. Save it as `TARDTE3.CTA`.

1. From the **Testcase** menu, select **Save**.

   The **Save as filename** dialog box appears.

   The default filename of `TARDTE3.cta` is displayed.

   Click **Save As**.

The **COBOL Tester** window appears. It now has a filename at the top (`D:vavatesttardte3.cta`).

5.4.4 Temporarily Suspending the Tutorial

If you want to suspend the tutorial, you can save the testcase, then return to it (and the tutorial) by opening the testcase and continuing the tutorial. Just select Testcase and **Save** before you leave COBOL Tester (which you do by double-clicking the control box at the top left corner of the **COBOL Tester** window).

To re-open the testcase:

1. Start COBOL Tester.
2. From the **Testcase** menu, select **Open...**
3. Type in the file name `TARDTE3.CTA`, or click **File list...** and browse through your directories till you find this file.
4. Click **Open**.

5.4.5 Creating the First Script

Now that you have created the testcase, you must add scripts to it. Each script consists of a unique set of input data which is used to generate output data. Each script covers part of the code of the program, with as few duplications of the same code as possible:

1. From the **Edit** menu, select **Create script**.

   The **Script** window appears; it displays the Procedure Division of the TARDTE3 COBOL program.

2. Browse the **Script** window.

   a. The left side of the window shows line numbers. If a number is greyed, then the line cannot be executed (it is a node, a delimiter or a comment).

   b. The right side of the window shows the COBOL lines, much as you would see them in a listing.

   c. Between the line numbers and the lines of code there are vertical lines, with short horizontal lines at each end. These lines are called "nested brackets." The nested brackets link nodes and delimiters. The lines between the ends of a bracket are sub-paths.

      For example, if you scroll down to lines 33 (000033) through lines 38, you will notice that they are joined by the ends of a bracket. Line 33 holds an IF statement. Line 36 holds an intermediate ELSE statement, and line 38 holds the concluding END-IF statement.

   d. Scroll to the top of the file. The four input data items are displayed on the top four lines. There is a pink box to the right of each input item. The pink color indicates that the item is not yet defined. The "IN:" tag indicates that these are input values. Whenever you run a script, you must provide input values.

   e. Scroll downwards. There are four more pink boxes near the end of the file. The tag "EX:" indicates that these are expected output values. You do not need to provide expected output values, but if you do, COBOL Tester is able to compare the expected results with the actual results.

3. Enter the first input value.

   a. Scroll to the top of the program.

   b. Double-click the variable name INPUT-DATE in the rounded box.

      The **Set Data** dialog box appears.

      Normally, the Data Available box lists all the data items that you have defined for this data item. The list is empty, because none are yet defined.

   c. Define a new data item.

      1) Click **New**.

         The **Edit Data** window appears.

         This window shows a data item (INPUT-DATE) and its attribute (X(8)), so that you enter a value that makes sense.

      2) Type the value `03/17/98` and press Enter.

         Dates are entered in the format “MM/DD/YY.”

         The value is displayed beside the data item.

      3) From the **Data** menu, select **Save**.

         The **Save As** dialog box appears.

         Name the data item and its associated value. After the item is named, you can use it with another script.

      4) Enter `MM/DD/YY-OK` and click **Save as**.

      5) Close the **Edit Data** box (by double-clicking the control box).

      The **Set Data** box appears. Now MM/DD/YY-OK appears in the list of Data Available.

   d. Assign this data value as input, by clicking < < **Input**.

   e. Close the **Set Data** window, by clicking **Close**.

      The **Script** window reappears. Note that the first input value box is blue. This shows that the variable now has an input value.

4. Repeat the process to set values for INPUT-FORMAT, OUTPUT-FORMAT, and MSG.

   a. Double-click the variable name INPUT-FORMAT in the rounded box.

   b. Click **New**.

   c. Type the value `MM/DD/YY` and press Enter. From the **Data** menu select the **Save** option. Save the data to `INF-MM/DD/YY`. Close the **Edit Data** window.

   d. Click < < **Input**, to assign this value.

   e. Click **Next**.

   f. Repeat the procedure, to create a new value and assign it to OUTPUT-FORMAT. Assign this the value `YYMMDD` and the name `OUTF-YYMMDD`. Then repeat the procedure, to create a new value and assign it to MSG. Do not assign it a value, but assign it the name `OK`.

   g. Click **Close** to close the **Set Data** window. This returns to the **Script** window. All of the input boxes are now blue.

The entry of the script is complete. You now need to save it and run it.

5.4.6 Saving the Script

1. From the **Script** menu, select **Save**.

   The **Save As** window appears.

2. Enter the value SCRIPT-01, and click **Save as**.

   When you come to creating scripts for your programs, you may prefer to give them descriptive names.

3. Close the **Script** window by double-clicking the control box.

The **Testcase** window appears, and shows one script. To the right of the script name, the Result column shows Not yet run. It is time to run the script.

5.4.7 Running the Script

1. Select the script.
2. From the **Selected** menu, select **Run**. (A shortcut is to press the F4 key.)

   The **Run** window appears, and shows messages as the program is compiled, and the test module is built and run.

3. When the Script run ended message appears, click **Close**.

The **Testcase** window displays the result of running this script. The result is OK, and the coverage is 41%. The reason that the result is OK is that no output values were entered. Since there was nothing to compare, COBOL Tester assumes that no comparisons are wanted, and so the run is marked as successful.

5.4.8 Checking Coverage

The test run shows 41% coverage. It is possible to look at what is covered by this script, and to increase the coverage (looking for 100% coverage).

1. Double-click on the script SCRIPT-01.

   The **Script** window appears.

2. Look at the line numbers. The numbers for the first nine lines should have a light gray background. If they do not, from the **View** menu, select **Paths run by script**. The gray background shows that the line has been covered.
3. Scroll down the code. There is grey shading from lines 1-10, 31, 40, 43-44, 47-48, 54, 58, 60, 77-78, 81-84, 86-89, 92, 95-96.

Before extending the coverage, the tutorial shows what happens if you put in an expected value.

5.4.9 Setting an Output Value and Running the Script

1. Scroll to line 81.
2. Double-click INPUT-DATE in the rounded box.

   The **Set Data** dialog box appears.

3. Create a new **Data Available** item. Assign it the value 980318. Remember to press Enter after you type the value. Assign it the name OUT-YYMMDD. Select OUT-YYMMDD, and assign it as an expected output, by clicking << **Expected**.

This date has the right format, but the wrong value. It should be 980317, if it is to match with the input date.

4. Click **Close** to close the **Set Data** window.

   The **Script** window appears.

5. Press F4, to run the script. (You can run the script from the **Script** window, or from the **Testcase** window.)
6. Respond to the Run message about the script being modified by clicking **Yes**.
7. When the script run ends, click **Close**.
8. From the **Script** menu, select **Show report**.

   The script report appears. The coverage is displayed as 0 %, because the Result is an Unexpected result.

9. Click the **Result** tab.

   The results appear.

10. If necessary, widen the window by dragging the right hand side, so that you can see the text in the **Result** column.
11. Look under the **Result** column. The fifth line has a Differ result.
12. Double-click the fifth line (line number 81).

    The detailed result for INPUT-DATE appears.

13. Look at the bottom of the window. The expected output is displayed as 980318 and the output is displayed as 980317.
14. Close the **Detailed Result** and the **Report** windows.

    The **Script** window reappears.

15. Close the **Script** window.

    The **Tester** window reappears. The red cross through the script icon indicates that it has been run, but the result was unexpected.

    It is possible to change a data value from the **Testcase** window, instead of through a script.

16. From the **Testcase** menu, select **Manage data**.

    INPUT-DATE is displayed in the Parameter field. This is the parameter that needs changing. You can access other parameters from this field, by clicking the down arrow at the right edge of the field—but do not do that now.

17. Double-click OUT-YYMMDD, in the Data Available box.

    The **Edit Data** window appears.

18. Change the value to 980317 and press Enter.
19. Close the **Edit Data** window by double-clicking the control box.
20. Respond to the **Close** message by clicking **Save**.
21. Respond to the **Save** message by clicking **Yes**.
22. Close the **Manage Data** window by double-clicking the control box.

    You have now changed the expected output for one data item of the script. In the **Testcase** window, the Result has changed to Not yet run, since the change in data invalidated the previous test run.

23. Run the script.

After the script has finished running, return to the **Testcase** window. Notice that the red cross has been replaced by a blue √, indicating that the result was OK. The coverage is again shown as 41%.

5.4.10 Extending Coverage

You extend coverage by changing the values of the input variables. To identify suitable new values, search the script for black line numbers that are not yet backed by grey.

1. Double-click the script.

   The **Script** window appears.

2. Look down the script.

   Lines 11 and 16 are not covered, but the line numbers are greyed. This shows that they are nodes, and therefore cannot be covered.

   The first line needing coverage is line 12.

3. Take a blank sheet of paper, and on it jot down INPUT-FORMAT, which is the input variable, and YY/DD/MM, which is the value for this variable. Skip to line 30.

   Lines 12 to 30 are covered by other values of INPUT-FORMAT.

   Line 31 is an EVALUATE clause. You are going to set INPUT-FORMAT to "YY/DD/MM," so the code bracketed by lines 32 to 38 cannot be covered by this test script.

4. The next line that can be covered is line 41, which is covered when WRK-DTE-YY < "00" OR > "99." To find out where WRK-DTE-YY comes from, scroll to lines 8–10. These lines show that WRK-DTE-YY is derived from INPUT-DATE.

5. On the sheet of paper, jot down INPUT-DATE and AA/03/03. This date needs to be given the format set by INPUT-FORMAT. The leading "AA" should force the error condition.

6. Scroll back to line 41.

7. Look beyond the bracket lines covering line 41. The next line that can be covered is line 55.

   In fact, by placing an error value in the year of INPUT-DATE, this line of code is covered. Furthermore, the error condition forces a jump in the COBOL program to TARDATE-END.

   For the next script, you have only written down two variables. There is no need to create a new value for OUTPUT-FORMAT, because no code using OUTPUT-FORMAT is going to be executed by this script.

To cater for these input variable values, create a new script.

1. Close the script window.

   The **Testcase** window appears.

2. From the **Edit** menu, select **Create script**.

3. Assign the new data item called YY/MM/DD-BAD-YY with the value AA/03/03 to INPUT-DATE.

4. Assign the new data item called INF-YY/DD/MM with the value YY/DD/MM to INPUT-FORMAT.

5. Assign OUTF-YYMMDD to OUTPUT-FORMAT. Make sure you highlight the correct item before you click << **Input**. Assign OK to MSG.

6. Save the script as SCRIPT-02.

7. Run the script.

8. When the script is finished, close it, and look at the testcase window.

   The script coverage is 25%, and testcase coverage has increased to 50%.

To increase coverage, you now have to repeat the process, by creating new scripts. Inspect the path coverage to date, jot down new values for the INPUT-FORMAT, INPUT-DATE and OUTPUT-FORMAT, then create a script that uses the new values.

Make sure that when you are looking at the script, you set the coverage to **Paths covered by testcase** (set from the **View** menu).

The following sections suggest values for each variable, and the coverage after each new script is added.

Script 3

- INPUT-DATE = “ABCDE” name = “YYDDD-BAD”
- INPUT-FORMAT = “YYDDD” name = “INF-YYDDD”
- OUTPUT-FORMAT = existing OUTF-YYMMDD
- MSG = existing OK
- Script name = “SCRIPT-03”

Run the script. After it is finished, close it, and check the coverage. The script coverage is 27% and the testcase coverage is now at 58%.

Script 4

- INPUT-DATE = “930015” name = “YYMMDD-BAD-MM”
- INPUT-FORMAT = “YYMMDD” name = “INF-YYMMDD”
- OUTPUT-FORMAT = existing OUTF-YYMMDD
- MSG = existing OK
- Script name = “SCRIPT-04”

Run the script. After it is finished, close, and check the coverage. The script coverage is 45%, and the testcase coverage is now at 63%. The testcase coverage is increasing only slowly, but this is because the scripts are testing error conditions.

Script 5

You can create this script quickly by right-clicking on Script 4 (SCRIPT-04), and selecting **Copy** from the menu, then typing in the name of the new script (SCRIPT-05), and pressing **Copy**. Then open the new script and edit it to change just the INPUT-FORMAT.

- INPUT-DATE = existing YYMMDD-BAD-MM
- INPUT-FORMAT = “BBAADD” name = “INF-BAD”
- OUTPUT-FORMAT = existing OUTF-YYMMDD
- MSG = existing OK
- Script name = “SCRIPT-05”

Run the script. After it is finished, close, and check the coverage. The script coverage is 21% and the testcase coverage is now at 67%.

Script 6

- INPUT-DATE = “96060” name = “YYDDD-LEAP-YR”
- INPUT-FORMAT = existing INF-YYDDD
- OUTPUT-FORMAT = “YYDDD” name = “OUTF-YYDDD”
- MSG = existing OK
- Script name = “SCRIPT-06”

Run the script. After it is finished, close, and check the coverage. The script coverage is 60% and the testcase coverage has jumped to 92%.

Script 7

- INPUT-DATE = “02/29/97” name = “MM/DD/YY-BAD-DD”
- INPUT-FORMAT = existing INF-MM/DD/YY
- OUTPUT-FORMAT = existing OUTF-YYDDD
- MSG = existing OK
- Script name = “SCRIPT-07”

Run the script. After it is finished, close, and check the coverage. The script coverage is 43%, and the testcase coverage has increased to 94%

Script 8

- INPUT-DATE = “98371” name = “YYDDD-TOO-MANY”
- INPUT-FORMAT = existing INF-YYDDD
- OUTPUT-FORMAT = “MM/DD/YY” name = “OUTF-MM/DD/YY”
- MSG = existing OK
- Script name = “SCRIPT-08”

Run the script. After it is finished, close, and check the coverage. The script coverage is 54%, and the testcase coverage is up to 96%.

Script 9

- INPUT-DATE = “98355” name = “YYDDD-OK”
- INPUT-FORMAT = existing INF-YYDDD
- OUTPUT-FORMAT = existing OUTF-MM/DD/YY
- MSG = existing OK
- Script name = “SCRIPT-09”

Run the script. After it is finished, close, and check the coverage. The script coverage is 54%, and the testcase coverage is up to 98%.

Script 10

- INPUT-DATE = existing YYDDD-OK
- INPUT-FORMAT = existing INF-YYDDD
- OUTPUT-FORMAT = “BBAADD” name = “OUTF-BAD”
- MSG = existing OK
- Script name = “SCRIPT-10”

Run the script. After it is finished, close, and check the coverage. The script coverage is 56%, and the testcase coverage has finally reached 100%.

5.4.11 Completing the Tutorial

This completes the tutorial using COBOL Tester.

You can experiment further, if you wish, to find out more about COBOL Tester′s capabilities. In particular, you may wish to add expected output values to the sample scripts, and run them again. If you wish to test the program thoroughly, you should do this.

As well, you may need more scripts. Although the sample scripts provide 100% coverage, they do not exercise the code completely. For example, the PERFORM UNTIL statement (line 99) is really 12 nodes compressed into one. To test the code thoroughly, you would want to check it against a date from each month. Likewise, the check on the number of days in each month (line 40) also needs checking 12 times, for each month (maybe 13, if you include leap years).

Even though COBOL Tester can make it much easier for you to create test records, you still need to apply commonsense to what it provides.

You also need to transfer the input values into a proper set of transactions, to create the distilled testbed. A quick way to list the values for input records is to run a report. You can do this by selecting all the scripts (from the **Testcase** window), and choosing **Print report** from the **Selected** menu.

When you finish experimenting, from the **Testcase** menu choose **Save**, then close COBOL Tester and close the associated folders.

The tutorial demonstrated the development of coverage for a legacy program. COBOL Tester may be unable to develop coverage for converted programs, because at the moment it is unable to handle Language Environment (LE) calls.

5.5 Introducing Application Testing Collection

The Application Testing Collection (ATC) is a set of tools that focus on path coverage and distillation. The three tools in ATC are:

- Coverage Assistant

  Coverage Assistant measures code coverage in application programs written in the COBOL and PL/I languages and compiled by specific IBM COBOL and PL/I compilers.

- Distillation Assistant

  Distillation Assistant monitors file reads for a specified file that contains logical keys and determines which keys increase code coverage. It then creates a ″distilled″ copy of the input file, containing only the records containing these keys.

  You use the smaller distilled file during program testing to obtain equivalent code coverage, using less time and resources. Distillation Assistant distills files for applications written in the COBOL language and compiled by specific IBM COBOL compilers.

- Source Audit Assistant

  Source Audit Assistant compares two levels of source code and places the results in a comparison report. Source Audit Assistant helps locate differences, which makes it easy for you to verify if changes are valid, or examine items that may need attention. In particular, using Source Audit Assistant allows you to develop targeted coverage. Instead of attempting to develop path coverage for an entire program, you provide data such that the coverage includes all the lines which Source Audit Assistant has indicated are different. Final coverage may be less than 100%, but it will be targeted at the lines that have changed, and thus are most critical in testing the conversion.

Application Testing Collection supplies interactive screens to make it easy for you to set up the batch jobs needed to run each process. Here are some examples:

```
------------------------ ATC Primary Option Menu V1R1M0 ---------------
Option ===> 1

0  Defaults      Manipulate ATC defaults
1  CA/DA         Coverage and Distillation Assistant
2  SAA           Source Audit Assistant

Enter X to Terminate
```

Figure 7. Application Testing Collection Primary Option Menu

The Primary Option menu gives you access to all of the tools in Application Testing Collection, and also gives you access to Application Testing Collection defaults, which apply to all the tools.

Coverage Assistant and Distillation Assistant share many processes. This is reflected in the structure of their common menu:

```
----------------- Coverage and Distillation Assistant -----------------
Option ===>

1 CntlFile      Work with the CA/DA Control File
2 Setup         Create JCL for Setup
3 StartMon      Create JCL to Start the Monitor
4 CA            Coverage Reports
5 DA            Distillation
6 Monitor       Control the CA/DA Monitor

Enter END to Terminate
```

Figure 8. Coverage Assistant and Distillation Assistant Menu

5.6 Building Path Coverage with Coverage Assistant

Coverage Assistant provides a series of steps which process a program. The outcome of this processing is an annotated program listing, showing the path coverage that arises from the supplied testbed.

Before a program can be tested for coverage, it must first be instrumented by:

- Editing a control file to specify setup information. The control file identifies compiler listing and object code locations.

- Executing the setup job. This creates control breaks in the object code to produce a “zapped” object module.
- Linking the object module. This object module is now ready to check for coverage.

The monitor must be running while the test jobstream is executing, as the monitor picks up information based on the control breaks and writes it to files used by Coverage Assistant. The monitor is stopped once the test jobstream has finished executing.

Coverage Assistant uses the output files created during the monitoring process to produce its coverage reports and annotated listings. Coverage Assistant produces output for COBOL or PL/I.

The sample program run through the Coverage Assistant processing is a program that checks the master file, and updates by one year any SECURITY CARD EXPIRY date that is expired.

5.6.1 The Coverage Process Step By Step

This section does not include a detailed, step-by-step, tutorial that you can use to run Coverage Assistant—if you do not have ATC, then you can′t run a tutorial, and if you do have ATC, the package already provides a detailed tutorial.

The only difference between developing path coverage as a general testing technique, and developing path coverage for Year 2000 testing is that when you are developing path coverage for Year 2000 testing, you may be content with coverage of all date functions, and not worry about the coverage of other pieces of code.

For those without Coverage Assistant, the following information outlines the steps to produce output using Coverage Assistant, plus a sample output.

Each step also shows the sample jobs that have been installed from the CD-ROM (*hlq*.VA2000TS.ATC). The COBOL sample is listed first, followed by the PL/I sample:

1. Compile the program

   CTARMU6X or CPL1MU6X

2. Run the setup JCL

   This JCL puts breakpoints into the compiled object modules—STARMU6X or SPL1MU6X

3. Run the link JCL

   This JCL links the object modules with the inserted breakpoints—LTARMU6X or LPL1MU6X

4. Start the monitor

   The monitor handles breakpoint processing when the program is run—XTARMU6X or XPL1MU6X

5. Run the program

   GTARMU6X or GPL1MU6X

6. Stop the monitor

The monitor is controlled through the ATC online facility

7. Generate Coverage Assistant reports

   RTARMU6X (for report RTARMU6X) or RPL1MU6X (for report RPLMU6X)

8. Inspect the reports. If necessary, restore the databases, add more data, and repeat the coverage process, by returning to step 4 on page 52.

5.6.2 A Sample Coverage Assistant Report

The sample Coverage Assistant report is found at *hlq*.VA2000TS.LISTING(RTARMU6X) for COBOL, and *hlq*.VA2000TS.LISTING(RPL1MU6X) for PL/I.

The first few pages of this report show summary information.

The bulk of the report is the coverage listing. This is a listing of the program, annotated by symbols indicating the type of coverage. The following segment of the program, Figure 9 on page 54, shows the main control logic of the program, with annotations.

The annotations (under column **A**) have the following meanings:

&	A conditional branch instruction that has executed both ways
>	A conditional branch instruction that has branched but not fallen through
V	A conditional branch instruction that has fallen through but not branched
:	Non-branch instruction that has executed
¬	Instruction that has not executed

```
       A
000275               200-MAIN-PROCESS.

000276
000277 :                 READ EMPLOYEE-MASTER-FILE

000278                       AT END
000279 :    1                GO TO 200-EXIT
000280                   END-READ.
000281
000282 >                 IF  EMPLOYEE-ID = "999999"  THEN    B
000283 ¬    1                GO TO 200-MAIN-PROCESS.         C
000284
000285 &                 IF  EMPLOYEE-DATE-TERMINATED = ZEROS THEN     D
000286
000287 :    1                MOVE EMPLOYEE-SECURITY-EXP TO TARDATE-DATE-YYDDD
000288 :    1                MOVE "YYDDD"               TO TARDATE-INPUT-FORMAT
000289 :    1                MOVE "YYYYDDD"             TO TARDATE-OUTPUT-FORMAT
000290 :    1                CALL  TARDTE3X     USING TARDATE-DATE
000291      1                                         TARDATE-INPUT-FORMAT
000292      1                                         TARDATE-OUTPUT-FORMAT
000293      1                                         TARDATE-MESSAGE
000294 :    1                MOVE TARDATE-DATE-YYYYDDD  TO WORK-SEC-YYYYDDD
000295
000296 &    1                IF  WORK-SEC-YYYYDDD <= WORK-DATE-YYYYDDD THEN
000297
000298 :    2                    PERFORM 300-PRINT-REPORT THRU 300-EXIT
000299
000300      1                END-IF
000301
000302                   END-IF.
000303
000304 :                 GO TO 200-MAIN-PROCESS.
000305
000306               200-EXIT.
000307 :                 EXIT.
```

Figure 9. Segment of Coverage Assistant Report

For example, line **B** is a conditional branch that has been branched, but not fallen through—no employee has the ID of "999999." Consequently line **C** is an instruction that has never been executed. In contrast, line **D** is a conditional branch that has been executed both ways, so the following instructions have also been executed.

Once you have reviewed the coverage report, you may then add more data to the testbed, and rerun the coverage process (as illustrated in Figure 5 on page 39).

5.6.3 Further Comments on Coverage Assistant

Path coverage applies to one program. When you are using Coverage Assistant, you can zap many programs (or modules), and run the path coverage process over all the programs in one monitored run. Coverage Assistant is able to provide coverage statistics for each module, or the run as a whole.

When you are undertaking system testing, where information is processed by a series of programs, you may find that the information that provides full path coverage for the first program in the series does not provide full path coverage for the last program in the series.

Here are some possible approaches to this problem:

- Check the path coverage of the last program in the series, and then work towards the start of the series, making sure that at each stage, the output of the previous program provides adequate coverage for the following program.
- Focus on the program that seems to be the most complex. Build maximum path coverage for this, and don′t worry about the other programs in the series.

When building path coverage, you are seeking to maximize it. You may find that it is not possible to build the coverage to 100%. There could easily be error conditions that are difficult to test. Remember that this is Year 2000 testing, and make sure that the path coverage covers all date-related code.

5.7 Developing Targeted Coverage with Source Audit Assistant

Source Audit Assistant provides a comparison of two levels of a program source code, normally before and after conversion. The result is a comparison report, which highlights differences between the programs.

This comparison report provides the means for developing targeted coverage. You must make sure that any path coverage covers at least the lines mentioned in the comparison report. These are the lines of code that have changed with conversion, and therefore are lines that must be tested.

5.7.1 Running Source Audit Assistant

This section provides an outline of the running of Source Audit Assistant.

Source Audit Assistant looks at two static items—two source code listings. It does not need to look at the dynamic execution of a program. Thus the process is simpler than running Coverage Assistant or Distillation Assistant.

Running Source Audit Assistant takes one step. The Source Audit Assistant control panel, shown in Figure 10 on page 56, provides the run options and the parameters controlling the run.

```
------------------------ Execute Source Audit Assistant ---------------
Option ===>

1  Run          Run Source Audit Assistant
2  ViewCmp      View Source Audit Assistant Compare File
3  ViewLog      View Source Audit Assistant Log File

Enter END to Terminate

New Source File:
  Data Set Name . . . . 'ATC.VA2000.COBOL.JCL.CNTL(TARMU6X)'

Old Source File:
  Data Set Name . . . . 'ATC.VA2000.COBOL.JCL.CNTL(TARMU6)'

SAA Output Compare File:
  Data Set Name . . . . 'hlq.TEST.SAA.CMP'

SAA Output Log File:
  Data Set Name . . . . 'hlq.TEST.SAA.LOG'

Programming Language:
  Source Language . . . COBOL     (C|C++|PL/I|COBOL|LCOB)

Select Line Audit Filters:
  Comments  NO (Yes|No) Declares  NO (Yes|No) Reformatted  NO (Yes|No)
```

Figure 10. The Source Audit Assistant Control Panel

By selecting filters, you can control the information displayed on the report. For example, you can tellSource Audit Assistant to disregard any differences that occur in comment lines, or lines that have been reformatted, but not otherwise changed. By selecting filters carefully, you can remove from the report information that you do not want.

With the parameters provided, Source Audit Assistant is able to proceed, and compare the two files.

5.7.2 A Sample Source Audit Assistant Report

The sample Source Audit Assistant report is found at *hlq*.VA2000TS.LISTING(SAAREPT), for COBOL. It is about 360 lines long. The following segments of the report illustrate the contents. Figure 11 shows the redefinition of a few items in Working Storage. In particular, note item TARDATE-DATE-YYYY, on line **A**.

```
THE FOLLOWING LINE PAIR(S) HAVE BEEN REPLACED

   174   New              05  TARDATE-DATE-YYYY       PIC 9(4).   A
   153   Old              05  TARDATE-DATE-YY         PIC 9(2).

   175   New          03  TARDATE-INPUT-FORMAT        PIC X(10).
   154   Old          03  TARDATE-INPUT-FORMAT        PIC X(8).

   176   New          03  TARDATE-OUTPUT-FORMAT       PIC X(10).
   155   Old          03  TARDATE-OUTPUT-FORMAT       PIC X(8).
```

Figure 11. Item Redefinitions in the Source Audit Assistant Report

The second segment, Figure 12 on page 57, shows this particular item used in the Procedure Division, line **B**.

```
THE FOLLOWING LINE PAIR(S) HAVE BEEN REPLACED

   333   New            MOVE "YYYYDDD"               TO TARDATE-INPUT-FORMAT.
   265   Old            MOVE "YYDDD"                 TO TARDATE-INPUT-FORMAT.

   334   New            MOVE "MM/DD/YYYY"            TO TARDATE-OUTPUT-FORMAT.
   266   Old            MOVE "MM/DD/YY"              TO TARDATE-OUTPUT-FORMAT.

   335   New            CALL TARDTE3X             USING TARDATE-DATE
   267   Old            CALL "TARDTE3"            USING TARDATE-DATE

   343   New            MOVE "MM/DD/YYYY"            TO TARDATE-OUTPUT-FORMAT.
   275   Old            MOVE "MM/DD/YY"              TO TARDATE-OUTPUT-FORMAT.

   344   New            CALL TARDTE3X             USING TARDATE-DATE
   276   Old            CALL "TARDTE3"            USING TARDATE-DATE

   348   New            ADD 1 TO  TARDATE-DATE-YYYY.   B
   280   Old            ADD 1 TO  TARDATE-DATE-YY.
```

Figure 12. Replaced Lines in the Procedure Division, from the Source Audit Assistant Report

When you develop coverage of this program, you want to be sure that this particular line (line 280 in the old program, or line 348 in the new program) is covered.

5.7.3 Further Comments on Source Audit Assistant

This example has focused on using Source Audit Assistant to provide targeted coverage.

Source Audit Assistant can also provide a form of audit, to check the completeness of code conversion. As well as the Compare report shown above, you can create a Focus report, by providing a list of seeds, or variable names. The Focus report is produced by scanning the Compare report, and reporting on seeds that are not shown in any changed lines, and changed lines that do not contain at least one seed reference.

The Focus report cannot tell you if lines of code have not been converted when they ought to be. However, by the judicious selection of seeds, the Focus report will point out potential problems that require further investigation.

5.8 Using Debug Tool to Check Coverage

Debug Tool is a general purpose tool that allows you to check the running of C, COBOL and PL/I programs.

Debug Tool allows you to debug a program interactively. It also allows you to use script files. By this means you can provide the same process to Debug Tool repeatedly without needing to type in all the Debug Tool commands.

This process shows how Debug Tool can create a frequency count for each line of code in a program. The frequency count can then be used to check coverage.

In this tutorial, the method is applied to the original CICS transaction, and the original Employee Master file.

You cannot run this tutorial until you have completed the steps in Chapter 6, “Creating the Baseline Results” on page 69. The current tutorial assumes the existence of the first ATP (or WITT Year2000 for Windows) script. It is strongly suggested that you complete the testing tutorials before you run this tutorial. Note that the results from this tutorial are not used as input into any other tutorial.

5.8.1 Recreating the Original Testbed

The current testbed must be returned to the original testbed.

1. Run the REXX procedure `TARDVSM1` from *hlq*`.VA2000TS.CLIST`.

   This recreates EMP.MASTER.ONLINE.

5.8.2 Setting up the Auto Test Performer Transactions

If you are using WITT Year2000 for Windows to enter transactions, you can skip this procedure. The method of naming WITT Year2000 for Windows scripts is different from that used to name ATP scripts.

1. Open an OS/2 session.
2. Change to the directory holding the ATP transactions (`X:vaatptartranscicsone`).
3. Type

   ```
   rename MAIN.TSF AGED.TSF
   copy DISTILL.TSF MAIN.TSF
   ```

4. Close the OS/2 session.

5.8.3 Starting the CICS Job, and Setting up for Debug Tool

1. Start a CICS session.

   You must now make sure that the Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMAST)`.
4. Look along this line to the second status word, which comes after `Vsa`.
5. If the second status word is `Ope`, go to step 8.
6. Move the cursor to the second status word, `Clo`.
7. Type `O`, then press Enter.

   The status word changes to `Ope` (for Open).

8. Press `PF3` to exit the CEMT display.
9. Clear the screen.
10. For VSE users, if necessary, install the group EQA to make Debug Tool available, with the command:

    ```
    CEDA I G(EQA)
    ```

11. Enter the transaction `DTCN`

The Debug Tool CICS Interactive Facility appears.

12. COBOL Against the Transaction Id (entry line 2) type NB03 and against the Command File (entry line 9) type *hlq*.VA2000TS.DEBUG(COBDB0) for OS/390 or userlib.(COBDB0.CMD) for VSE COBOL

    PL/I Against the Transaction Id (entry line 2) type MB03 and against the Command File (entry line 9) type *hlq*.VA2000TS.DEBUG(PL1DB0) for OS/390 or userlib(PL1DB0.CMD) for VSE PL/I

13. Press PF4 (Add).

    If the add fails because the profile exists, press PF5 (Replace).

14. Press PF3 (Exit).

15. COBOL Start program NB03 COBOL

    PL/I Start program MB03 PL/I

    There is a delay, while Debug Tool is initialized, then the update screen appears.

16. Make sure that Caps Lock is on.

When the screen displays the Employee Update window, your CICS session is ready for you to enter data.

5.8.4 Using Auto Test Performer to Enter Transactions

The equivalent WITT Year2000 for Windows procedure is found at B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Start Auto Test Performer Manager.

   ATP starts, and the Manager window appears.

2. Click on the + sign at the left of the TARTRANS group icon.

   The testcases are listed under the group.

3. Double-click on the CICSONE testcase entry.

   The right of the screen lists two scripts and other items.

4. Right-click on the MAIN.TSF entry.

5. Select **Playback** from the pop-up menu.

   The log window appears and is currently empty. As well, the **Playback Control** window appears. This contains four buttons: **Stop playback**, **Start/Resume playback**, **Pause playback**, and **Step**.

6. If the **Playback Controls** window is not at the bottom of the screen, drag it to the bottom, where the buttons are not obscured by the CICS session.

7. If the CICS window is minimized, restore it to its normal size.

8. From the Playback Controls, click **Start/Resume playback**, the arrow pointing to the right.

   ATP enters the recorded transactions into the CICS session.

   Sit back and watch the transactions flow!

When the playback has finished, ATP displays the message that the script playback is completed.

9. Click **OK**.

5.8.5 Looking at the Frequency Counts

1. Click the CICS window, and wait.

   After a few seconds, the Debug Tool window appears.

   The bottom window shows the program frequency counts.

2. Press PF11 (Zoom Log).

   The log now fills the entire window.

3. Enter `TOP`, to scroll to the top of the window.
4. Scroll down the lines in this window. The line numbers relate to the listing of the program, found at *hlq*.`TARMU3.LIST`.
5. Where there is a "0" against a line, the line has not been covered. By inspection of the program, you can work out what data needs to be added to cover this line.
6. When you finish your inspection, press PF3, and respond `Y` to the question about terminating the session.
7. Close DTCN, by entering DTCN and pressing F10.
8. Close the CICS session.
9. Close ATP Manager.

There is no point in looking at screen images. ATP was used in this tutorial merely to provide the means of entering the transactions.

If you use this technique to check coverage, you may find that it is useful to add transactions to the ATP script by using a text editor. Using ATP in this way guarantees repeatability of results.

5.8.6 An Explanation of the Debug Tool Script

Figure 13 on page 61 shows the working portion of the Debug Tool script (for VSE users, the format of the SETL LOG and SET SOURCE Debug Tool differ slightly from this OS/390 example).

```
STEP;
SET LOG ON FILE <hlq>.TARMU3.LOG;
CLEAR LOG;
SET SOURCE ON ("TARMU3") <hlq>.VA2000TS.LISTING(TARMU3);

SET FREQUENCY ON (TARMU3);                          A

AT APPEARANCE TARDTE3
   PERFORM
    SET SOURCE ON ("TARDTE3") <hlq>.VA2000TS.LISTING(TARDTE3);

    AT ENTRY TARDTE3
       PERFORM
         SET FREQUENCY ON (TARDTE3);                B
         GO;
       END-PERFORM;
    GO;
   END-PERFORM;


AT OCCURRENCE CEE067
   GO;

AT TERMINATION
   LIST FREQUENCY *;                                C
GO;
```

Figure 13. Debug Tool Script for Path Coverage

In this script, the critical command is SET FREQUENCY ON, which is executed at **A** and **B**. It is needed twice because it counts the frequency for a program and a subprogram.

The LIST FREQUENCY * command, at **C**,lists all line frequency counts to the log file.

5.9 Distilling a File with Distillation Assistant

Distillation Assistant is another component of Application Testing Collection (ATC).

It uses the same approach as Coverage Assistant—a series of steps processing a program. For Distillation Assistant, the outcome is a list of distilled keys and optionally a distilled file. Distillation Assistant also modifies object code to insert breakpoints, and then monitors the running of the modified code. Distillation Assistant works with COBOL and PL/I. COBOL samples are provided with this redbook.

The sample program run through the Distillation Assistant processing is a program that checks the master file, and updates by one year any SECURITY CARD EXPIRY date that has expired; the same program as for Coverage Assistant.

5.9.1 The Distillation Process Step By Step

There is no difference in distillation between general testing and Year 2000 testing.

For those without Distillation Assistant, the following information outlines the steps to produce output using Distillation Assistant, plus a sample distilled key listing.

Each step also shows the sample jobs that have been installed from the CD-ROM (*hlq*.VA2000TS.ATC). The samples are all for COBOL:

1. Compile the program

   CTARMU6X

2. Run the setup JCL

   This JCL puts breakpoints into the compiled object modules—STARMU6X

3. Run the link JCL

   This JCL links the object modules with the inserted breakpoints—LTARMU6X

4. Start the monitor

   The monitor handles breakpoint processing when the program is run—XTARMU6X

5. Run the program

   GTARMU6X

6. Stop the monitor

   The monitor is controlled through the ATC online facility

7. Create the distilled file

   DISTCOPY

8. Generate Distillation Assistant reports

   DTARMU6X (for report DTARMU6X)

The first six steps in this process are the same as for Coverage Assistant. Since the first three steps are setting-up steps, you don′t need to repeat these if you have already applied Coverage Assistant to the program.

Developing path coverage is a cyclic process, where you check the path coverage after each cycle, and then repeat the cycle with more information, if you want to increase path coverage.

Distillation, on the other hand, is not cyclic. You only go through the distillation process once for each file.

5.9.2 A Sample Distillation Assistant Report

The sample Distillation Assistant report is found at *hlq*.VA2000TS.LISTING(DTARMU6X).

The bottom part of the listing shows the list of distilled keys. The keys were reduced from 14 (for the master file after updating) to 5. The next figure shows the original and distilled key lists.

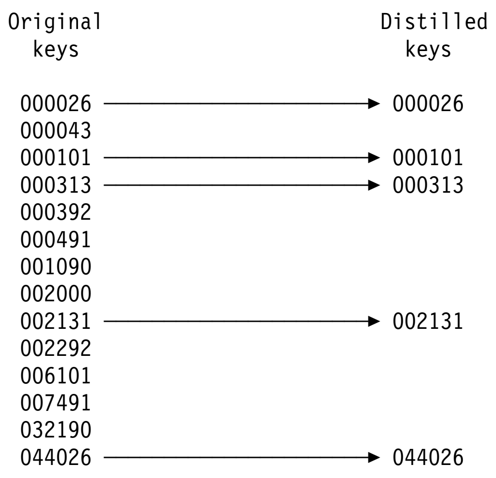


Figure 14. Distillation Assistant Key List

Distillation Assistant offers a means of converting this list into a file. However, the file must be a sequential file, and you may prefer to use your own means to create an indexed file.

5.9.3 Further Comments on Distillation Assistant

Each distillation pass applies to only one file. If you want to distill more than one file (for example, a master file and a transaction file), then you have to repeat the process, adjusting the control file, to indicate the file that you are currently distilling.

Before you repeat a distillation against a different file you must restore all files to their original state.

A distilled file will not necessarily produce 100% path coverage of a program. Even if you are distilling a large production file, the data on this file may not exercise all the paths in the program. The more data on the file, the more likely that coverage is greater.

Also, the process of distilling a file may remove records that are necessary for more complete coverage. For example, imagine you have a program that is accumulating a value found on individual records. If you reduce the number of records, the accumulated value may never exceed a threshold value, so that some code may not be executed.

For these reasons, you may find it necessary to recheck the path coverage after you have distilled files.

5.10 Using Debug Tool to Build a Distilled Key List

This procedure uses Debug Tool to build a list of record keys. You can then use this list to build a distilled file.

This example applies only to COBOL.

The procedure uses a CICS session to enter transactions, but it could be readily modified to work for a batch program.

During this procedure, you enter information for four employees. The log file produced during the data entry shows that entering information for two employees will suffice.

You cannot run this tutorial until you have completed the steps in Chapter 6, “Creating the Baseline Results” on page 69. The current tutorial is run against the distilled testbed, although it does not update the testbed. In practice, you run distillation against an original testbed.

5.10.1 Recreating the Distilled Testbed

There is no need to recreate this testbed, since the procedure does not look at records that have been updated.

5.10.2 Starting the CICS Job and Opening Files

1. Start a CICS session.

   You must now make sure that the distilled Employee Master files is open for the exclusive use of CICS.

2. Enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMAST)`.
4. Look along this line to the second status word, which comes after `Vsa`.
5. If the second status word is `Ope`, go to step 8.
6. Move the cursor to the second status word, `Clo`.
7. Type `O`, then press Enter.

   The status word changes to `Ope` (for Open).

8. Press `PF3` to exit the CEMT display.
9. Clear the screen.

The file is now open.

5.10.3 Entering Transactions

1. Enter the transaction `DTCN`

   The Debug Tool CICS Interactive Facility appears.

2. Against the Transaction Id (entry line 2) type `NB03` and against the Command File (entry line 9) type *hlq*`.VA2000TS.DEBUG(COBDB2)`
3. Press PF4 (Add).

   If the add fails because the profile exists, press PF5 (Replace).

4. Press PF3 (Exit).
5. Start program `NB03`

   There is a delay of a few seconds while Debug Tool is initialized.

   The Employee Maintenance window appears.

6. For EMPLOYEE NO type `000101` and press Enter.

The information about Peter Seller appears, along with the message "I-ENTER CHANGE DETAILS."

The cursor moves to the DEPT CODE field.

7. Tab to the ZIP CODE.
8. For ZIP CODE type `XXXXX` and press Enter.

   The message "E-ZIP CODE NOT NUMERIC" appears.
9. Tab back to the EMPLOYEE NO, type `001090` and press Enter.

   The information about Julie Milner-Winter appears, along with the message "I-ENTER CHANGE DETAILS."

   The cursor moves to the DEPT CODE field.
10. Tab to the ZIP CODE.
11. For ZIP CODE type `XXXXX` and press Enter.

    The message "E-ZIP CODE NOT NUMERIC" appears.
12. Tab back to the EMPLOYEE NO, type `006101` and press Enter.

    The information about Penny Marker appears, along with the message "I-ENTER CHANGE DETAILS."

    The cursor moves to the DEPT CODE field.
13. Tab to the ZIP CODE.
14. For ZIP CODE type `XXXXX` and press Enter.

    The message "E-ZIP CODE NOT NUMERIC" appears.
15. Tab back to the EMPLOYEE NO, type `044026` and press Enter.

    The information about Alison Clarke appears, along with the message "I-ENTER CHANGE DETAILS."

    The cursor moves to the DEPT CODE field.
16. Tab to the ZIP CODE.
17. For ZIP CODE type `XXXXX` and press Enter.

    The message "E-ZIP CODE NOT NUMERIC" appears.
18. Press F3.

    The CICS transaction is terminated.
19. Close DTCN, by entering DTCN and pressing F10.
20. Close the CICS session.

5.10.4 Looking at the Distillation Key List

You will find the distillation list at *hlq*.`TARMU3.LOG2` for OS/390, or `userlib(TARMU3.LOG2)` for VSE.

The top of the list repeats the Debug Tool script. This is followed by lines that look like this:

```
* 'DISTILL 575.1 000101                              '
* 'DISTILL 576.1 000101                              '
* 'DISTILL 577.1 000101                              '
```

Figure 15. Lines from the Distillation Key List

The important item in each line is the third item, “000101.” This is the value of the key that you entered, which for these three lines is the key for Peter Seller′s record.

Scan down the list. There are pages and pages for key 000101. However, near the bottom there are about 15 lines for key 001090. This is the key for Julie Milner-Winter′s record.

There are no entries for key 6101 or key 44026.

This list shows that entering the third and fourth key provided no further coverage of the code in the program. The number of transactions has been distilled from four to two.

You may be wondering why the list includes two keys, and not just one. The reason is that the Peter Seller record does not have a terminated date, whereas the Julie Milner-Winter record one does, and thus exercises new lines of code.

5.10.5 An Explanation of the Debug Tool Script

Figure 16 shows the working portion of the Debug Tool script.

```
STEP;
SET LOG ON FILE <hlq>.TARMU3.LOG2;
CLEAR LOG;
SET SOURCE ON ("TARMU3") <hlq>.VA2000TS.LISTING(TARMU3);
SET ECHO OFF;
01  WORK-EMP   PIC X(6);
01  OUT-REC    PIC X(50);
MOVE SPACES TO WORK-EMP;
MOVE SPACES TO EMP-ID;

AT LINE (489 - 897)                                 A
  PERFORM
   MOVE EMP-ID TO WORK-EMP;
   IF  WORK-EMP NOT = SPACES
       MOVE SPACES    TO OUT-REC;
       MOVE "DISTILL" TO OUT-REC(1:8);
       MOVE %LINE     TO OUT-REC(9:6);
       MOVE WORK-EMP  TO OUT-REC(15:6);
       LIST UNTITLED(OUT-REC);
       CLEAR AT LINE %LINE;                         B
   END-IF;
   GO;
  END-PERFORM;


AT OCCURRENCE CEE067
   QUIT;

GO;
```

Figure 16. Debug Tool Script for Distillation

In this script, the `AT LINE (489 - 897)` command at **A** sets a breakpoint for each line in the COBOL procedure division. Each time a breakpoint is encountered, the details of the employee key are written to the log file. Also, the breakpoint is cleared (**B**). This means that the execution of a line is logged only the first time, regardless of how many times the line is executed.

5.10.6 Extending This Example

This example has focussed on the essentials of using Debug Tool to help in distillation.

There is no need to restrict yourself to online programs. The same technique can be applied to batch programs.

The current log file has key information for each line of the program that was executed. You could readily process this list so that it provides a list of the unique keys, in this case condensing the list to two lines.

Once you have the list of unique keys, you must then create the distilled transactions. For online entry, you distill the transactions by entering just those in the key list. For batch processing, you have to remove transactions from the batch files.

Chapter 6. Creating the Baseline Results

This process creates a set of baseline results, which are compared with the post-fix results (from Chapter 8, “Creating the Post-Fix Results” on page 85). The baseline results reflect the state of the system now. They do not include any Year 2000 dates, as the code has not been changed to handle Year 2000 dates.

Input Original program code, distilled testbed

Output Baseline master files, reports, and screen images

This process is shown in Figure 17.

The basic process is to run distilled transactions through the original code, updating the distilled master files to produce the baseline master files and baseline reports.

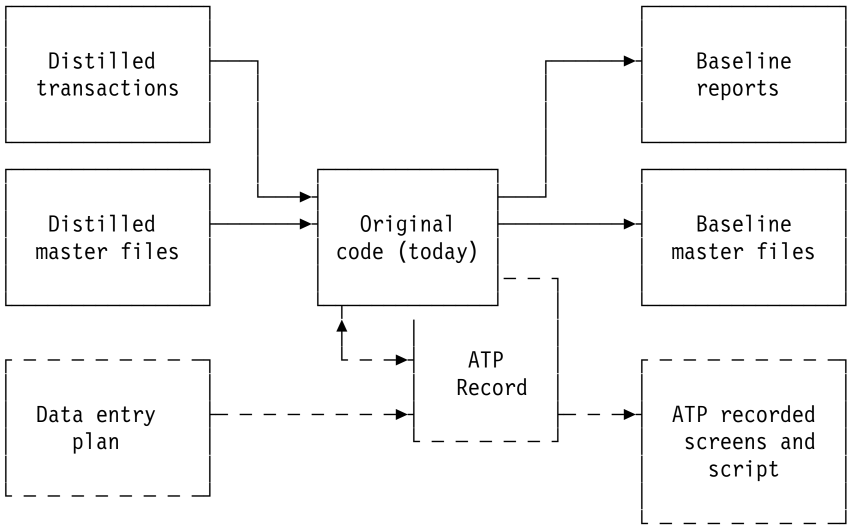


Figure 17. Creating the Baseline Results

If the code involves online (interactive) programs, where information is entered on a screen, then there is a second process. This second process requires the entry of this online information while Auto Test Performer (ATP) is running. ATP then captures the entry to produce a script automatically, and records screen images when requested. In this figure, and following figures, WITT Year2000 for Windows performs a function identical to ATP.

The diagram Figure 17 refers to “original code.” This is because the process is not necessarily tied to one program. The original code can be a suite of programs used to process the transactions. So this process applies equally to unit (one program) testing and system testing.

6.1 Screen Capture and Script Creation - Auto Test Performer

This tutorial works through the process of capturing screens and an online transaction entry script using Auto Test Performer (ATP).

Before starting this tutorial, make sure that:

- ATP is installed on your workstation
- All the sample data and program files are installed on your mainframe

There are data files supplied to start the process. These are:

- *hlq*.DEPT.MASTER.ONLINE

  The Department Master file (not updated, but used for reference)

- *hlq*.EMP.MASTER.START

  The initial Employee Master file. This file is updated as part of the processing. It is copied to produce *hlq*.EMP.MASTER.ONLINE, which is the file that is processed.

6.1.1 Creating a Sample Distilled Master File

This procedure creates the file *hlq*.EMP.MASTER.ONLINE by copying the file *hlq*.EMP.MASTER.START. No JCL is provided to update or delete *hlq*.EMP.MASTER.START. Provided this file is not deleted or changed, you can always come back to this point and work through the chain of tutorials. This means that the tutorials can be run many times. If the file is damaged, reload it using the procedure in Chapter 3, “Installation and Customization” on page 19.

1. Execute REXX procedure TARDVSM1.

   (See 3.5.6, “Conventions in This Redbook for OS/390” on page 27, and 3.6.6, “Conventions in This Redbook for VSE” on page 31, for detailed instructions.)

6.1.2 Capturing the Screens and Scripts - Auto Test Performer

Auto Test Performer is a workstation product. It is able to capture screens for comparison, and also to capture keystrokes as they are entered. This part of the tutorial tells you how to start ATP, and then what to enter as transactions.

If you have to enter online transactions, it is important that you create a Data Entry Plan. This is where you map out the details of the transactions that you are going to enter. The plan might take the form of hand-written notes, or entries in a spreadsheet; the form does not really matter. However, you must know what you want to enter before you start the session. This means that the script you create is fairly clean, which means that the script is easier to convert and age.

This tutorial includes the data for you to enter. The transactions correspond to a distilled data set, though they have been created manually, rather than using a coverage or distillation tool.

There are only two transactions to enter. One transaction adds a new employee, the other changes the detail of an existing employee.

Because this is an online program, most of the error checking can be done by entering an invalid value into a field, and then correcting it immediately.

6.1.3 Starting the CICS Session

1. Start a CICS session.

   You must now make sure that the Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMAST)`

4. Check the high level qualifier of the file. If it has the value you want, go to step 5. If it is not the high level qualifier you want to use, you must change it by the following procedure:

   - For OS/390:

     a. Look along the Fil(EMPMAST) line to the second status word, which comes after `Vsa`. If the second status word is `Clo`, go to step 4d.

     b. Move the cursor to the second status word, `Ope`.

     c. Type `C`, then press Enter.

        The status word changes to `Clo` (for Closed).

     d. Move the cursor over the high level qualifier, and change it to the high level qualifier you want.

   - For VSE:

     Modify your CICS startup JCL to point to the correct files, and restart CICS.

5. Look along the Fil(EMPMAST) line to the second status word, which comes after `Vsa`. If the second status word is `Ope`, go to step 8.

6. Move the cursor to the second status word, `Clo`.

7. Type `O`, then press Enter.

   The status word changes to `Ope` (for Open).

8. Press `PF3` to exit the CEMT display.

9. Clear the screen.

   You must now make sure that the Department Master file is open for the exclusive use of CICS.

10. Enter the transaction

    `CEMT I FI (DEP*)`

    The list of files starting with DEP appears.

11. Find the line starting `Fil(DEPMAST)`

12. Look along the Fil(DEPMAST) line to the second status word, which comes after `Vsa`. If the second status word is `Ope`, go to step 15.

13. Move the cursor to the second status word, `Clo`.

14. Type `O`, then press Enter.

    The status word changes to `Ope` (for Open).

15. Press `PF3` to exit the CEMT display.

16. Clear the screen.

17. COBOL ▶ Start program NB03 ◀ COBOL

    PL/I ▶ Start program MB03 ◀ PL/I

18. Make sure that Caps Lock is on.

When the screen is displaying the Employee Update window, your CICS session is ready for you to enter data.

6.1.4 Starting Auto Test Performer

The equivalent WITT Year2000 for Windows procedure is found at B.1, "Starting WITT Year2000 for Windows" on page 133.

If you are using VisualAge COBOL, Test for OS/2, then:

1. Double-click the VisualAge Test folder icon.
2. Double-click the Manager icon.

If you are using WITT Year2000 for OS/2, then:

1. Double-click the WITT Year2000 folder icon.
2. Double-click the Manager icon.

ATP is now started, and the Manager window appears.

Having started ATP, you now have to create a group and testcase, to hold the information that is captured as you enter the CICS details.

6.1.4.1 Entering the Group Path

The group path indicates where information is to be held.

1. From the **Root** menu, select **Path...**

   The **Group Path** dialog box appears.

2. Enter `X:vaatp`, where "X" is the drive you are using (normally C or D), and click **OK**. If you are asked about creating the path, click **Yes**.

6.1.4.2 Creating a Group and a Testcase

A group is a convenient way of holding in one place all the scripts to be applied against one program.

1. From the **Edit** menu, select **Create group...**

   The **Create Group** dialog box appears.

2. Enter `TARTRANS` and click **Create**.

   The TARTRANS group is listed on the left of the screen.

3. Right-click on the TARTRANS group.
4. Select **Create testcase...** from the pop-up menu.

   The **Create Testcase: TARTRANS** dialog box appears.

5. Enter `CICSONE` and click **Create**.
6. Click on the + sign at the left of the TARTRANS group icon.

   The CICSONE testcase entry appears.

7. Double-click on the CICSONE testcase entry.

   The MAIN.TSF entry appears on the right of the screen.

8. Right-click on the MAIN.TSF entry.

9. Select **Record** from the pop-up menu.

   The **MAIN.TSF - Record** window appears.

   The large area holds the recorded script, although it shows nothing yet. The window in the top right corner is the **Record Controls** window. It includes the Stop record, Start record, Select session and Save images buttons.

10. Drag the **Record Controls** window to the bottom of the screen. Position it so that you can see the buttons, and the buttons are not obscured when your CICS session is running.

    The buttons are used to control the recording of the script, and the capture of screen images.

11. From the Record Controls, click **Select session**.

    The **MAIN.TSF - Select session** dialog box appears.

12. Click `Host session` under target.

13. Select the Host session ID (for example, `G`).

14. Click **Select**.

The EMPLOYEE MAINTENANCE window in CICS appears, ready for you to enter transactions.

6.1.5 Enter the Transactions

OS/2 In this section, steps between the OS/2 tags are undertaken using Auto Test Performer. OS/2

Windows In this section, steps between the Windows tags are undertaken using WITT Year2000 for Windows. Windows

ATP is now ready to capture the transaction details as you enter them.

Attention: Before you start entering transactions, a word of warning. There are quite a few dates on the CICS screen. When you are entering a date, you may be tempted to use the cursor keys to jump over values that have not changed, or not bother to enter the year if the default year value has not changed. When you enter a date, make sure that you type the complete date. Be sure not to use the cursor keys to skip values, and be sure to type the year. The reason is that later in the testing process, you will convert the ATP script. This conversion will not be correct if you do not enter complete dates.

This tutorial focuses on exercising date fields. This means that sometimes a bad date is entered, to make sure that the program checks the date. For complete coverage, you could enter bad data in other fields. This is not done in the current tutorial, to minimize the number of tutorial steps.

As you enter information into the EMPLOYEE MAINTENANCE window, ATP (or WITT Year2000 for Windows) records the information.

Here are the details to type. Note that after typing the data into a field, you press either Tab or Enter. The instructions tell you which to press:

1. For EMPLOYEE NO type 002000 and press Enter.

   The message "I-ENTER EMPLOYEE DETAILS" appears in the message line, which starts "MSG==>" and is near the bottom of the screen.

   This shows that you have entered an employee number that is not in the Employee Master file, and hence is a new employee.

   The cursor moves to the DEPT CODE field.

2. For DEPT CODE type 0002 and press Tab.
3. For NAME type CHARLES ROGERS and press Tab.
4. For ADDRESS 1 type 34 MILES WAY and press Tab.
5. For ADDRESS 2 type WAYVILLE and press Tab.
6. For ADDRESS 3 type SOUTH CAROLINA and press Tab.
7. For ZIP CODE type 34567 and press Tab.
8. For JOINED type 08/14/97 and press Tab.
9. For BIRTHDATE type 31/02/77. This time, press Enter.

   A default SECURITY CARD EXPIRY date is entered (one year from the JOINED date).

   The CICS transaction now validates the information typed so far.

   The message "E-INVALID DATE" appears, and the cursor is still on the BIRTHDATE.

   The month and day are in the wrong order for the BIRTHDATE.

10. To correct BIRTHDATE type 02/31/77 and press Enter.

    The message "E-INVALID DATE" is still displayed.

    Too many days in February for the BIRTHDATE.

11. To correct BIRTHDATE type 02/28/77 and press Enter.

    The message "I-SCR VALID PF10 TO UPDATE" appears.

    This message indicates that the data on the screen is now valid. The record can be added to the file. However, to finish, enter the TERMINATED date and replace the default SECURITY CARD EXPIRY date.

12. Tab to the TERMINATED field.
13. For TERMINATED type 08/14/96 and press Enter.

    The message "E-JOINED > TERMINATED DATE" appears.

    This message indicates that joined date is less than the terminated date. This does not make sense. How can you leave a job before you start it?

    The cursor is on the JOINED date. This date is OK. It is the TERMINATED date that is at fault.

14. Tab to the TERMINATED field.
15. To correct TERMINATED type 08/14/98 and press Tab.

    The cursor moves to the SECURITY CARD EXPIRY field.

16. For SECURITY CARD EXPIRY type 08/14/98 and press Enter.

All the information is now entered and correct. Now it is time to take a picture of the screen.

17. OS/2 Click the **Save images** button on the Record Controls. This button looks like a camera. The window blinks as ATP takes its picture. OS/2

    Windows Press <Ctrl+Alt+F10>, to save an image. Windows

18. Press F10, to add the record.

    The message "I-RECORD ADDED" appears.

    The record for the new employee has been successfully added.

    To complete the transaction entry, update the record of an existing employee.

19. Press Tab.

    The cursor moves to the EMPLOYEE NO field.

20. For EMPLOYEE NO type 000043. and press Enter.

    The information about Mary Gardener appears, along with the message "I-ENTER CHANGE DETAILS."

21. Tab to the TERMINATED field.

22. For TERMINATED type 06/07/96 and press Enter.

    The message "I-SCR VALID PF10 TO UPDATE" appears.

    The record can now be updated on the file.

23. OS/2 Click the **Save images** button on the Record Controls to save another image. OS/2

    Windows Press <Ctrl+Alt+F10>, to save another image. Windows

24. Press F10.

    The message "I-RECORD CHANGED" shows that the record has been successfully updated.

25. Press F3.

    The CICS transaction is terminated.

To finish recording, you need to stop ATP Record or WITT Year2000 for Windows recording.

OS/2 To stop ATP:

1. Click the **Stop record** button on the Record Controls.

   The ATP recording session is now over. The **MAIN.TSF - Record** window appears.

2. Close the **Record** window, and click **Yes** when asked if you want to save the file. OS/2

Windows To stop WITT Year2000 for Windows recording:

1. Press <Ctrl+Alt+F12>. Windows

6.1.6 Ending the CICS Session

The Employee Master file must be closed, so that other jobs can use it.

1. In your CICS session, clear the window, then enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

2. Find the line starting `Fil(EMPMAST)`.
3. Look along this line to the second status word, which comes after `Vsa`.
4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed).

5. Press `PF3` to exit the CEMT display.
6. Clear the screen.
7. Close the CICS session.

The recording session is now finished.

6.1.7 A Comment on the Script

This script includes the entry of values that are in error, and are subsequently changed. When you are creating your own scripts, you should not include such values. Either make sure that you do not enter values in error, or if you do, edit the script to remove them.

There is one time when you want to deliberately enter values that are in error. This is when you want to check that the error has been detected. To do this, enter the error value, and immediately capture a screen image which includes an error message.

6.1.8 Looking at the Script

The equivalent WITT Year2000 for Windows procedure is found at B.2, “Looking at the Script” on page 134.

The script that you have created in ATP is a text file which you can edit:

1. In the ATP window, right-click on the MAIN.TSF file.
2. Select **Open** from the pop-up menu.

   The script is displayed in a text editor.

3. Look at the script.

   You should be able to see where you have pressed the Tab key or Enter key, or typed a date. With a little more effort, you should also be able to see where you entered a field. The two “Window_Compare” lines come from the screen captures.

   When you entered the script, it is possible that you may have typed a wrong key, and then pressed Backspace or Left Arrow to move the cursor to replace the wrong keystroke. If you did, the routine that converts the script will not convert everything, as it is expecting dates to be in “MM/DD/YY” format.

Check your script, and make sure that all your dates are properly entered. If not, edit the script to make sure that they are. (This normally involves deleting a few lines, to "close up" the date information.)

The following sample shows the script when the start of the JOINED date is entered as "08/41." The mistake was noticed and corrected by back spacing and typing the day "14," before adding "/97."

```
'Enter_Text "08/41"'
'Enter_Key <BCKSP>, 2'
'Enter_Text "14/97"'
```

This mistake is corrected by deleting from (and including) '41' on the first line up to (and including) the " before '14' on the third line, to consolidate the date entry into one line:

```
'Enter_Text "08/14/97"'
```

If you edit the script file, make sure you save it before closing the **Editor** window.

4. Close the **Editor** window.
5. Close the **Manager** window, and the ATP groups.

In the process you have just completed, you achieved three things:

- You used ATP to capture data entry, and thus create a script which you can play back through ATP.
- You used ATP to capture screen images, which you can use for comparison.
- You updated the Employee Master file, and along the way exercised the update program.

The updated Employee Master file is the baseline result that you will later compare with the post-fix results.

The script is going to be converted, so that it can be used to provide data entry to the converted code.

6.2 Creating a Baseline Report

This job creates the report TARRPA1.REPORT, which is the Security Card Expiry Dates report. The report lists the dates on which security cards will expire, in date order, and also lists the associated employee's ID and name.

The program that prints the report is basic. However, the JCL that runs the print job selects only employees that are still employed (Terminated Date has the value "000000"), and creates an input file for the report program, selecting only the relevant fields from the Employee Master file.

1. Execute job TARRPA1. For VSE users: The default Language Environment library is PRD2.SCEEBASE. Change in the job if necessary.
2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report from the job, by browsing TARRPA1.REPORT.

   The report should list six employees, starting with Drummond Rapper and ending with Vi Sproika.

The report is catalogued, so that it can be used later when the baseline results and post-fix results are compared.

Chapter 7. Building the Converted Testbed

This process creates a set of files adjusted to reflect the Year 2000 changes put into the programs. The converted testbed does not include Year 2000 dates.

Input Distilled testbed

Output Converted testbed

This process is shown in Figure 18.

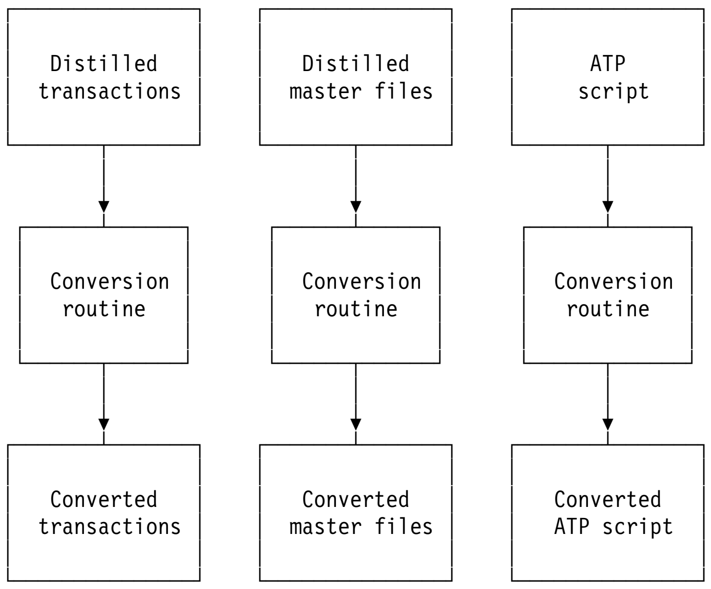


Figure 18. Converting the Distilled Testbed to Produce the Converted Testbed

After you have worked out the techniques you are going to apply to handle Year 2000 dates, you may have to modify the data, so that it can be handled by the new techniques.

If you are using a windowing technique, then there is no need to adjust any date fields in your records (though you may need to remove some records if they fall outside the bounds of your window). It is possible that you are using a windowing technique, but you are also changing the way in which you store your dates. For example, you might want to change their format from “MMDDYY” to “YYMMDD” In this case, even though you are using a windowing technique, you will have to run the master files and transactions through a conversion routine.

For this tutorial, you do not need to convert distilled transactions, because there are none. In reality, transactions are converted using techniques similar to converting master files.

7.1 Converting the Master File

This procedure uses DFSORT to convert date fields to Year 2000 ready formats. Most of the dates are converted by expanding them, by changing the year from YY to YYYY. During conversion, three of the fields are converted to packed (COMP-3) format. The security expiry date is being held as a windowed date, and thus is not converted.

1. Execute REXX procedure TARDVSM2.

   This job creates EMP.MASTER.ONLINE2 (before conversion) from EMP.MASTER.START.

2. Execute job TARCNV2.

   This job converts EMP.MASTER.ONLINE2, leaving the results in the same file.

3. Check the output of the job.

4. If the job failed, determine the reason, correct, and resubmit.

The job prints out the converted master file, so that you can check the dates.

This job illustrates the use of the DFSORT Year 2000 features. The parameters for the first DFSORT pass are shown in Figure 19. Compare these with the legacy COBOL file description Figure 20 on page 81. The PL/I file description has the same structure.

```
SORT FIELDS=COPY
OPTION Y2PAST=1900
OUTFIL OUTREC=(1,135,136,2,Y2C,138,3,141,2,Y2C,143,4,147,2,Y2C,
             149,3,152,2,Y2C,154,39)
```

Figure 19. Using DFSORT to Expand Dates

- The first OUTREC pair, “1,135,” selects the text information.
- The next OUTREC triplet, “136,2,Y2C,” selects “YY” from the EMP-DATE-JOINED field, and expands it to “YYYY.”
- The next OUTREC pair, “138,3,” selects “DDD” from the EMP-DATE-JOINED field.
- Likewise, the EMP-DATE-TERMINATED, EMP-DATE-MAINTAINED, and EMP-BIRTH-DATE fields are expanded.
- The last date, the EMP-SECURITY-DATE, is not expanded.
- The “Y2PAST=1900” option sets the base year for the 100-year window (1900-1999) applied to all two-digit dates.

```
 01  EMPLOYEE-MASTER-RECORD.
*        ** key field
     03  EMP-ID                       PIC X(6).
     03  EMP-DEPT-CODE                PIC X(4).
     03  EMP-NAME                     PIC X(30).
     03  EMP-ADDR-1                   PIC X(30).
     03  EMP-ADDR-2                   PIC X(30).
     03  EMP-ADDR-3                   PIC X(30).
     03  EMP-ZIP-CODE                 PIC X(5).
*        ** format (yyddd)
     03  EMP-DATE-JOINED              PIC 9(5).
*        ** format (yymmdd)
     03  EMP-DATE-TERMINATED          PIC 9(6).
*        ** format (yyddd)
     03  EMP-DATE-MAINTAINED          PIC 9(5).
*        ** format (yyddd)
     03  EMP-BIRTH-DATE               PIC 9(5).
*        ** format (yyddd)
     03  EMP-SECURITY-EXP             PIC 9(5) COMP-3.
     03  FILLER                       PIC X(41).
```

Figure 20. The Legacy Employee Master Record

The OS/390 job uses the IEBGENER utility to convert some date fields to packed (COMP-3) format. The VSE job uses the COBOL program TARCONV to achieve the same results. The parameters in the IEBGENER step are shown in Figure 21.

```
GENERATE MAXFLDS=5,
         MAXLITS=6
RECORD FIELD=(150,1,,1),
         FIELD=(7,151,ZP,151),
         FIELD=(7,158,ZP,155),
         FIELD=(36,165,,159),
         FIELD=(6,'      ',,195)
```

Figure 21. Using IEBGENER to Convert Dates to COMP-3

Here IEBGENER is used to convert the expanded EMP-DATE-MAINTAINED and EMP-BIRTH-DATE fields into COMP-3 fields, as shown in Figure 22 on page 82.

```
 01  EMPLOYEE-MASTER-RECORD.
*        ** key field
     03  EMP-ID                     PIC X(6).
     03  EMP-DEPT-CODE              PIC X(4).
     03  EMP-NAME                   PIC X(30).
     03  EMP-ADDR-1                 PIC X(30).
     03  EMP-ADDR-2                 PIC X(30).
     03  EMP-ADDR-3                 PIC X(30).
     03  EMP-ZIP-CODE               PIC X(5).
*        ** format (yyyyddd)  date expanded
     03  EMP-DATE-JOINED            PIC 9(7).
*        ** format (yyyymmdd) date expanded
     03  EMP-DATE-TERMINATED        PIC 9(8).
*        ** format (yyyyddd) packed date compressed
     03  EMP-DATE-MAINTAINED        PIC 9(7) COMP-3.
*        ** format (yyyyddd) packed date compressed
     03  EMP-BIRTH-DATE             PIC 9(7) COMP-3.
*        ** format (yyddd) packed date uses sliding window
     03  EMP-SECURITY-EXP           PIC 9(5) COMP-3.
     03  FILLER                     PIC X(39).
```

Figure 22. The Converted Employee Master Record

This job has two sorts and a merge as intermediate steps.

These steps are needed because the EMP-DATE-TERMINATED field is not mandatory. The file is split in two, depending on whether this field is present or not, processed, and then merged to provide a consolidated, converted file.

More information about the Year 2000 features of DFSORT is available from `http://www.storage.ibm.com/software/sort`.

7.2 Backing up the Converted Master File

This job backs up the converted master file to a file called *hlq*.EMP.MASTON2.BACK.

1. Execute REXX procedure TARBON2.

7.3 Converting the Auto Test Performer Script File

The equivalent WITT Year2000 for Windows procedure is found at B.3, "Converting the WITT Year2000 for Windows Script File" on page 135.

The Auto Test Performer (ATP) script file is a text file. The data it holds includes dates in the "MM/DD/YY" format. The converted CICS screen has dates entered in "MM/DD/YYYY" format. If you run the ATP script file against the converted CICS screen, then you will experience problems.

The REXX routine TARADJST converts dates, and can also age dates.

To run this routine:

1. Open an OS/2 window.
2. Change to the ATP directory that holds the script file (for example, `d:vaatptartranscicsone`).
3. Rename the MAIN.TSF file to DISTILL.TSF; type `rename main.tsf distill.tsf` and press Enter.

4. Start the TARADJST Rexx procedure; type `vataradjst` and press Enter.
5. Enter the FULL source file name: `DISTILL.TSF`
6. Enter the source date format: `MM/DD/YY`
7. Enter the FULL target file name: `MAIN.TSF`
8. Enter the target date format: `MM/DD/YYYY`
9. Enter the number of years to age dates: `0`

   For this procedure, the transactions are not aged.

   The routine creates the file MAIN.TSF. This is identical to the original file, except that the years in dates are now four digits long.

   The message indicates that it has updated 8 records, which are the records containing dates.

   The message asking for the source file name appears again.
10. Press Enter without entering any data.

    The routine finishes.
11. Close the OS/2 window.

This script file is used in Chapter 8, “Creating the Post-Fix Results” on page 85.

The script is structured so that it can also be used to convert a batch of files. The batch file is made up of sets of five lines, where each set of lines corresponds to responses to the five requests mentioned above. Any line starting with an asterisk (*) is a comment line, and is disregarded. Comment lines must be placed above each set of five lines.

To invoke batch processing, use the batch file as the standard input, like so:

```
\va\taradjst < batch.txt
```

Here is an example of a batch file, which converts three files in one pass:

```
SCRIPT1.TSF
MM/DD/YY
SCRIPT1C.TSF
MM/DD/YYYY
0
SCRIPT2.TSF
YY/MM/DD
SCRIPT2C.TSF
YYYY/MM/DD
0
SCRIPT1.TSF
MM/DD/YY
SCRIPT1C.TSF
MM/DD/YYYY
0
```

Chapter 8. Creating the Post-Fix Results

This process creates a set of post-fix results, which can be compared with the baseline results (from Chapter 6, “Creating the Baseline Results” on page 69), and with the 19xx results (from Chapter 12, “Creating the 19xx Results” on page 107). The post-fix results reflect the state of the system after code has been changed to handle Year 2000 dates. The post-fix results do not include data with Year 2000 dates.

Input	Converted program code, converted testbed, converted ATP scripts
Output	Post-fix master files, reports, and screen images

This process, shown in Figure 23, is similar to that of creating the baseline results.

The files used to drive the process are the converted files.

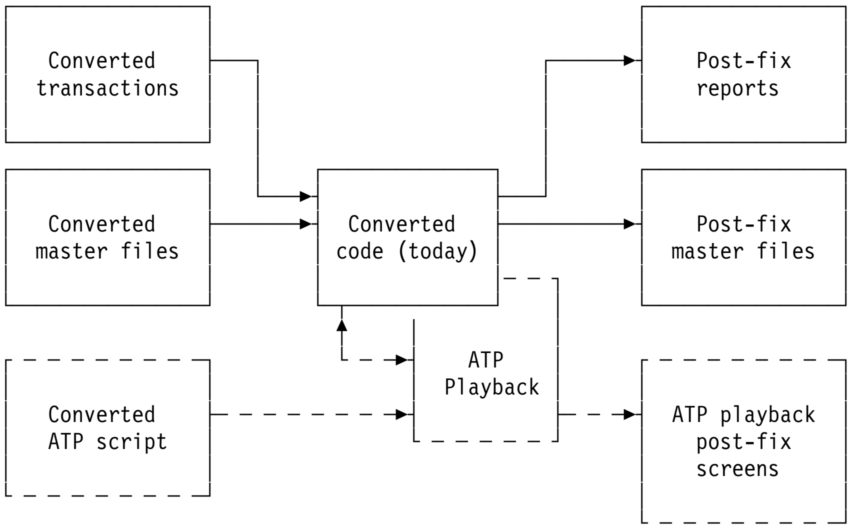


Figure 23. Creating the Post-Fix Results

A more important difference, if the process uses online programs, is that the online entries are not entered by hand. Instead, Auto Test Performer (ATP) is able to read the converted script and so provide data entry. And the output from ATP is not a script (since this has already been captured), but a log file and a set of screen images, for comparison.

8.1 Playback and Screen Capture - Auto Test Performer

This process is similar to the process by which Auto Test Performer captured screen images and the script.

This time Auto Test Performer (ATP) does the hard work. And, of course, the CICS program that is running is the converted program.

8.1.1 Starting the CICS Session

1. Start a CICS session.

   You must now make sure that the converted Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMST2)`.

   This time, you are using a converted CICS transaction, which uses a different version of the Employee Master file.

4. Look along this line to the second status word, which comes after `Vsa`.
5. If the second status word is `Ope`, go to step8.
6. Move the cursor to the second status word, `Clo`
7. Type `O`, then press Enter.

   The status word changes to `Ope` (for Open).

8. Press `PF3` to exit the CEMT display.
9. Clear the screen.
10. COBOL ▶ Start program `NB3E` (not NB03) ◀ COBOL

    PL/I ▶ Start program `MB3E` (not MB03) ◀ PL/I

11. Make sure that Caps Lock is on.

When the screen is displaying the Employee Update window, your CICS session is ready to receive data.

8.1.2 Using Auto Test Performer to Enter Transactions

The equivalent WITT Year2000 for Windows procedure is found at B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Start Auto Test Performer Manager.

   ATP starts, and the Manager window appears.

2. Click on the + sign at the left of the TARTRANS group icon.

   The testcases are listed under the group.

3. Double-click on the CICSONE testcase entry.

   The right of the screen lists two scripts and other items.

4. Right-click on the MAIN.TSF entry.
5. Select **Playback** from the pop-up menu.

   The log window appears and is currently empty. As well, the **Playback Controls** window appears. This contains four buttons: **Stop playback**, **Start/Resume playback**, **Pause playback**, and **Step**.

6. If the **Playback Controls** window is not at the bottom of the screen, drag it to the bottom, where the buttons are not obscured by the CICS session.
7. If the CICS window is minimized, restore it to its normal size.

8. From the **Playback Controls**, click **Start/Resume playback**, the arrow pointing to the right.

   Your converted transactions are now entered into the CICS session.

   Sit back and watch the transactions flow!

   When the playback has finished, ATP displays the message that the script playback is completed.

9. Click **OK**.

8.1.3 Ending the CICS Session

1. In your CICS session, clear the window, then enter the transaction

   `CEMT I FI (EMP*)`

   The list of files starting with EMP appears.

2. Find the line starting `Fil(EMPMST2)`.

3. Look along this line to the second status word, which comes after `Vsa`.

4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed). The file is now available for processing by other jobs.

5. Press `PF3` to exit the CEMT display.

6. Clear the screen.

7. Close the CICS session.

8.1.4 Viewing the Log File

The equivalent WITT Year2000 for Windows procedure is Steps 8 to 10 of B.4, "Using WITT Year2000 for Windows to Enter Transactions" on page 137.

1. Return to the **MAIN.TSF - Playback** window, and browse the log. Near the bottom, ATP has indicated that for the Window_Compares, a compare mismatch occurred. This is as expected, since the new CICS transaction displays dates with YYYY instead of YY.

2. Close the **MAIN.TSF - Playback** window. Click **Yes** to save the current log.

3. Close ATP Manager.

   The next tutorial shows you how to use ATP to visually compare the screens.

You have now updated the Employee Master file, creating the post-fix results, ready for comparison with the baseline results.

8.2 Creating a Post-Fix Report

This job creates the report `TARRPA2.REPORT`, which is the Security Card Expiry Dates report, for converted data.

The JCL is essentially the same as for TARRPA1, except that it uses a different version of the master file, and the DFSORT parameters have been adjusted to take into account the conversion of the data, and hence the conversion and repositioning of fields.

1. Execute job `TARRPA2`.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report `TARRPA2.REPORT`.

   The report should list six employees, starting with Drummond Rapper and ending with Vi Sproika.

Chapter 9. Comparing the Baseline Results with the Post-Fix Results

This process compares the baseline results with the post-fix test results, to check that the Year 2000 code changes are not causing any problems with pre-Year 2000 (current) dates.

Input	Baseline and post-fix files, reports, and screen images
Output	Comparisons of files, reports, and screen images

Figure 24 shows the process of comparing the baseline and post-fix master files. SuperC compares sequential files, which is why the master files are first converted from VSAM to sequential non-VSAM files.

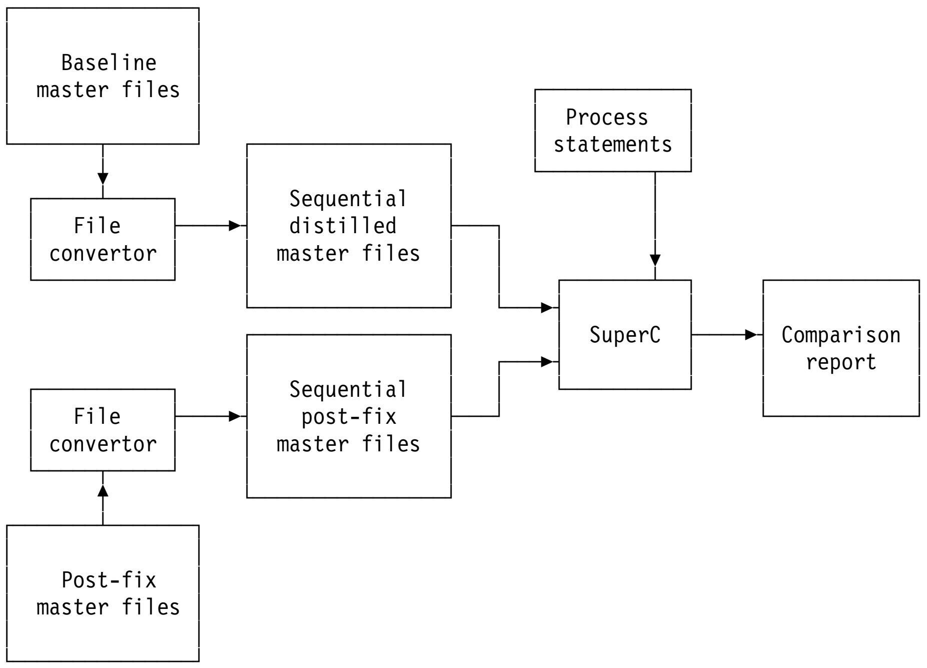


Figure 24. Comparing Baseline and Post-Fix Master Files

The process to compare reports is simpler, because the report files are already sequential non-VSAM files. Figure 25 on page 90 shows this process.

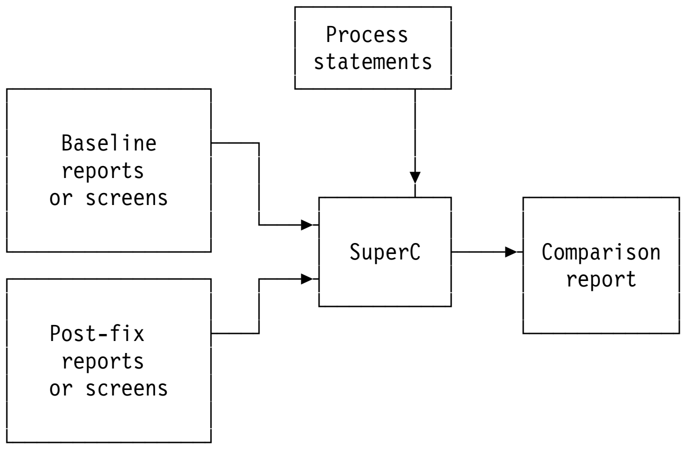


Figure 25. Comparing Baseline and Post-Fix Reports

You compare screen images captured by Auto Test Performer (ATP) by observation. Figure 26 shows this process.

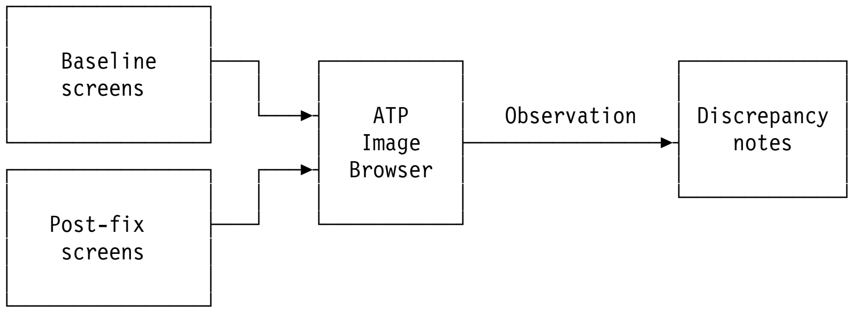


Figure 26. Comparing Baseline and Post-Fix Screen Images

9.1 Screen Comparisons - Auto Test Performer

This tutorial uses Auto Test Performer (ATP) to compare the baseline and post-fix screen images.

The equivalent WITT Year2000 for Windows procedure is found at B.5, “Screen Comparisons - WITT Year2000 for Windows” on page 138.

1. Open ATP Manager, and display the contents of the TARTRANS testcase.
2. Double-click the CICSONE group.
3. Double-click the IMAGES icon, on the right side of the window.

   The **CICSONE - Images** window appears.
4. Select all the lines in the table (click on the first line, then hold down Shift and click on the last line).
5. Right-click on the highlighted lines.
6. Select **Compare**.
7. If the **Screen image - Compare options** dialog box appears, click **OK**.

ATP compares Benchmark and Current images and displays the type of difference (“TEXT”) in the Difference column.

8. Right-click on the first line (Screen CICS0000).

9. Select **Browse**.

   ATP displays the **CICS0000 - Browse** window, and a screen image. The screen image contains the details of the employee added to the master file (Charles Rogers), just before updating.

10. Adjust the size of the window, so that it shows all of the image.

11. From the icon bar at the top of the window, click the **Differences** button.

    ATP highlights the years for the dates. The current screen image differs from the benchmark screen image at the highlighted areas. This difference is to be expected, since the dates have been converted to include YYYY instead of YY.

12. From the icon bar, click the **Benchmark image** button.

    ATP shows the original screen image, where the years are only two digit.

13. From the icon bar, click the **Current image** button.

    ATP shows the most recent screen image, where the years are four digit.

14. From the icon bar, click the **Next image** button.

    ATP displays the screen image for the employee whose details were changed (Mary Gardener).

15. Click the **Differences** button.

    Once again, ATP highlights the differences. This time, ATP highlights more than the years. It also highlights the second digit in the month and day, for the SECURITY CARD EXPIRY date.

16. Click the **Benchmark image** button.

17. Note that the SECURITY CARD EXPIRY date is “06/07/96,” the same as the TERMINATED date.

18. Click the **Current image** button.

19. Note that the SECURITY CARD EXPIRY date is “01/03/1998,” whereas the TERMINATED date is “06/07/1996.”

    Looks like there has been a problem in converting the code—the comparisons have more than the expected differences.

    Before attempting to track down the problem, complete looking at the other comparisons.

20. Repeat the process of looking at differences, the benchmark, and the current image, for the remaining images. Since the text images are essentially the same as the graphic images, they provide the same answers.

21. Close the **Browser** window.

    The CICSONE - Images window reappears.

    The screen image comparison is complete.

    At this point in the procedure, you would normally archive the images, which moves the current images into the benchmark images, destroying the benchmark images.

However, the comparisons have shown a problem. The archive step is delayed until this problem is resolved, and the comparisons provide the desired result.

22. Close the **CICSONE - Images** window.

23. Close the **ATP Manager** window.

9.2 Data Comparisons - SuperC

1. Execute job SUPERC1A.

   This job compares the master files.

2. Check the job output.

3. If the job failed, determine the reason, correct, and resubmit. Do not reject the job if the "SUPERC1A RUN" line shows an error code of 01.

The job produces the SuperC comparison report which includes the summary:

```
                 LINE COMPARE SUMMARY AND STATISTICS

13 NUMBER OF LINE MATCHES                1  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)    1
 0 REFORMATTED LINES                     1  PAIRED CHANGES (REFM+PAIRED INS/DEL)
 1 NEW FILE LINE INSERTIONS              0  NON-PAIRED INSERTS
 1 OLD FILE LINE DELETIONS               0  NON-PAIRED DELETES
14 NEW FILE LINES PROCESSED      5
14 OLD FILE LINES PROCESSED      6
```

Figure 27. SuperC Summary Output Comparing Distilled and Post-Fix Master Files

Line **1** shows that 13 lines matched, and there was 1 paired change (1 set of corresponding lines that did not match). Lines **5** and **6** show that there were 14 lines compared from each file.

The top of the SuperC report shows the records that did not match. This is the information entered for Mary Gardener:

```
I - 0000430002MARY GARDENER                 13/4 SILVER STREET           BROKEN HILL
INFO   Date cols 136:142  char 1990003        Comp=(1990003   )
INFO   Date cols 143:150  char 19960607       Comp=(1996 06 07)
INFO   Date cols 155:158  packed 1964323      Comp=(1964323   )
INFO   Date cols 159:161  packed 98003        Comp=(1998003   )  5T

D - 0000430002MARY GARDENER                 13/4 SILVER STREET           BROKEN HILL
INFO   Date cols 136:140  char 90003          Comp=(1990003   )
INFO   Date cols 141:146  char 960607         Comp=(1996 06 07)
INFO   Date cols 152:156  char 64323          Comp=(1964323   )
INFO   Date cols 157:159  packed 96159        Comp=(1996159   )  5B
```

Figure 28. Details from the SuperC Record Comparison

Line **5T** (date cols 159:161) of the top record is different from line **5B** of the bottom record.

These dates are the SECURITY CARD EXPIRY dates. The Julian date of 1998003 corresponds to 01/03/1998, and 1996159 corresponds to 06/07/1996. So SuperC is reporting the same discrepancy as the ATP comparison.

The bottom of the SuperC report, shows the SuperC process statements:

```
OY2C 136:140 YYDDD                   1
NY2C 136:142 YYYYDDD                 2
OY2C 141:146 YYMMDD,EMPTY            3
NY2C 143:150 YYYYMMDD,EMPTY          4
OEXCLUDE COLS 147:151                5
NEXCLUDE COLS 151:154                6
OY2C 152:156 YYDDD                   7
NY2P 155:158 YYYYDDD                 8
OY2P 157:159 YYDDD                   9
NY2P 159:161 YYDDD                   10
Y2Past 1900                          11
```

Figure 29. SuperC Process Statements, Baseline to Post-Fix Data Comparison

These statements tell SuperC how to interpret dates.

- An “O” at the start of a statement (for example, lines 1 and 3) indicates the statement applies to the “old” file, and an “N” (for example, lines 2 and 4) that the statement applies to the “new” file.
- The “OY2C” and “NY2C” lines come in pairs, for example, the pair of lines 1 and 2, and the pair of lines 3 and 4. They describe the column locations of dates. For example, the first pair (1 and 2) tell SuperC to compare a date of “YYDDD” starting in column 136 of the baseline master file to a date of “YYYYDDD” starting in column 136 of the post-fix master file. This is the “EMP-DATE-JOINED” field. See Appendix H, “Employee Master File Descriptions” on page 165 for the legacy and converted Employee Master file descriptions.
- The “EMPTY” keyword at the end of lines 3 and 4 indicates that the field might be empty—this is the “EMP-DATE-TERMINATED” field.
- The OEXCLUDE and NEXCLUDE statements (lines 5 and 6) tell SuperC to exclude columns from the comparison. They exclude the MAINTENANCE date. This is date when the records were last maintained. They are excluded because they depend on when the updating tutorials were run. The MAINTENANCE date is not used in any way in the sample system. If this date were included, comparisons may fail.
- The last line (11) sets the window that SuperC uses to work out 2-digit year dates. This line sets the window from 1900 to 1999.

9.3 Report Comparisons - SuperC

1. Execute job SUPERC1B.

   This job compares the two report files.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report output. This should show that 12 lines have matched. The reports have produced the same results.

   This SuperC job includes parameters to exclude the first two lines of each report because the top two lines include the date and time that the report was printed. This will be different each time the report is produced.

The report lists only six records, but there are blank lines between each report line, and SuperC compares all the lines in the report, even the blank ones.

This comparison produced a perfect match, even though the comparison of the master files did not. This comparison was only for employees still working. Mary Gardener was no longer working (her record had a TERMINATED date). Her record was not included in the comparison.

This job has only two process statements:

```
OEXCLUDE ROWS 1:2
NEXCLUDE ROWS 1:2
```

Figure 30. SuperC Process Statements, Baseline to Post-Fix Report Comparison

These two statements tell SuperC to exclude the first two rows, which are the header rows for the reports. These are excluded because they include the run date for the report.

9.4 Tracking Down the Bug

There are many methods that can be used to find the bug that has caused the compares to fail.

These include:

- Using SuperC to compare the legacy and converted COBOL source programs.
- Using Source Audit Assistant to compare the legacy and converted COBOL source programs.
- Using Debug Tool to step through the program while it is running.
- More traditional methods, such as a code walk-through, or working from listings and cross-references.

Just as detailed discussion of the methods of converting programs is outside the scope of this Redbook, so is a detailed discussion of the methods used to debug programs.

Using the methods suggested above, you will be able to find that the bug comes from one line of code which is in the legacy program, but not in the converted program. When a TERMINATED date is supplied, the line of code moves it to the SECURITY CARD EXPIRY date.

The error occurs at line 414. Here is the segment of code that is in error:

```
IF  TARDATE-MESSAGE NOT = SPACES THEN
    MOVE DFHBMASB  TO  TARM3XMTDATEA
    IF  NOT ERRORS THEN
        SET ERRORS TO TRUE
        MOVE -1    TO  TARM3XMTDATEL
        MOVE 5     TO WORK-MSG-CODE
    END-IF
ELSE
    MOVE TARDATE-DATE-YYYYDDD TO WORK-TERMINATED-YYYYDDD
    IF  WORK-TERMINATED-YYYYDDD < WORK-JOINED-YYYYDDD
        MOVE DFHBMASB  TO  TARM3XMJDATEA
                           TARM3XMTDATEA
        IF  NOT ERRORS THEN
            SET ERRORS TO TRUE
            MOVE -1    TO  TARM3XMJDATEL
            MOVE 12    TO WORK-MSG-CODE
        END-IF
    END-IF
END-IF
```

Figure 31. Segment of Code in Error in the Converted Program

Here is the segment of code with the line correctly inserted:

```
IF  TARDATE-MESSAGE NOT = SPACES THEN
    MOVE DFHBMASB  TO  TARM3XMTDATEA
    IF  NOT ERRORS THEN
        SET ERRORS TO TRUE
        MOVE -1    TO  TARM3XMTDATEL
        MOVE 5     TO WORK-MSG-CODE
    END-IF
ELSE
    MOVE TARM3XMTDATEO      TO TARM3XMSDATEO
    MOVE TARDATE-DATE-YYYYDDD TO WORK-TERMINATED-YYYYDDD
    IF  WORK-TERMINATED-YYYYDDD < WORK-JOINED-YYYYDDD
        MOVE DFHBMASB  TO  TARM3XMJDATEA
                           TARM3XMTDATEA
        IF  NOT ERRORS THEN
            SET ERRORS TO TRUE
            MOVE -1    TO  TARM3XMJDATEL
            MOVE 12    TO WORK-MSG-CODE
        END-IF
    END-IF
END-IF
```

Figure 32. The Correct Code

For the purpose of the testing tutorials, the bug has been successfully isolated, and you are ready to continue testing. The next step shows how to restart testing.

Chapter 10. Restarting Testing

This process repeats the procedures from Chapter 8, “Creating the Post-Fix Results” on page 85 and Chapter 9, “Comparing the Baseline Results with the Post-Fix Results” on page 89. A different version of the CICS program is used, and so the outcomes should be different.

Input	Correctly converted program code, converted testbed, converted ATP scripts
Output	Post-fix master files, reports, and screen images, and comparisons of these files, reports, and screen images

10.1 Restoring the Baseline

The building of the converted testbed has updated the master file. The file must be restored so the data used gives the same result.

1. Execute REXX procedure `TARRON2`.

   This job restores EMP.MASTER.ONLINE2 to the converted file before update.

10.2 Playback and Screen Capture - Auto Test Performer

In this procedure Auto Test Performer (ATP) plays back the converted transactions into a correctly converted CICS program.

10.2.1 Starting the CICS Session

1. Start a CICS session.

   You must now make sure that the Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMST2)`.

   You are still using the second version of the Employee Master file.

4. Move the cursor to the second status word for this file, and type `O`, and then press Enter.

   The status word changes to `Ope` (for Open).

5. Press `PF3` to exit the CEMT display.
6. Clear the screen.
7. COBOL Start program `NB3X` (not NB03 or NB3E) COBOL

   PL/I Start program `MB3X` (not MB03 or MB3E) PL/I

8. Make sure that Caps Lock is on.

When the screen is displaying the Employee Update window, your CICS session is ready for you to enter data.

10.2.2 Using Auto Test Performer to Enter Transactions

The equivalent WITT Year2000 for Windows procedure is found at B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Start Auto Test Performer Manager.

   ATP starts, and the Manager window appears.

2. Click on the + sign at the left of the TARTRANS group icon.

   The testcases are listed under the group.

3. Double-click on the CICSONE testcase entry.

   The right of the screen lists two scripts and other items.

4. Right-click on the MAIN.TSF entry.
5. Select **Playback** from the pop-up menu.

   The log window appears.

6. If the **Playback Controls** window is not at the bottom of the screen, drag it to the bottom, where the buttons are not obscured by the CICS session.
7. If the CICS window is minimized, restore it to its normal size.
8. From the Playback Controls, click **Start/Resume playback**, the arrow pointing to the right.

   Your aged transactions are now entered into the CICS session.

   When the playback has finished, ATP displays the message that the script playback is completed.

9. Click **OK**.

10.2.3 Ending the CICS Session

1. In your CICS session, clear the window, then enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

2. Find the line starting `Fil(EMPMST2)`.
3. Look along this line to the second status word, which comes after `Vsa`.
4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed). The file is now available for processing by other jobs.

5. Press `PF3` to exit the CEMT display.
6. Clear the screen.
7. Close the CICS session.

10.2.4 Viewing the Log File

The equivalent WITT Year2000 for Windows procedure is Steps 8 to 10 of B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Return to the **MAIN.TSF - Playback** window, and browse the log. Near the bottom, ATP has indicated that for the Window_Compares, a compare mismatch occurred.

The current screen images are being compared with the baseline screen images, so the screens are different because instead of YY, dates now include YYYY.

2. Close the **MAIN.TSF - Playback** window. Click **Yes** to save the current log.
3. Close ATP Manager.

You have now updated the Employee Master file, creating the post-fix results, ready for comparison with the baseline results.

10.3 Creating a Post-Fix Report

The report created last time compared correctly. However, for completeness, this procedure is included. You can skip it if you wish, but in proper retesting, you should redo every procedure.

1. Execute job TARRPA2.
2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report TARRPA2.REPORT.

   The report should list six employees, starting with Drummond Rapper and ending with Vi Sproika.

10.4 Screen Comparisons - Auto Test Performer

This tutorial uses Auto Test Performer (ATP) to compare the baseline and post-fix screen images.

The equivalent WITT Year2000 for Windows procedure is found at B.5, “Screen Comparisons - WITT Year2000 for Windows” on page 138.

1. Open ATP Manager, and display the contents of the TARTRANS testcase.
2. Double-click the CICSONE group.
3. Double-click the IMAGES icon, on the right side of the window.

   The **CICSONE - Images** window appears.
4. Select all the lines in the table.
5. Right-click on the highlighted lines.
6. Select **Compare**.
7. If the **Screen image - Compare options** dialog box appears, click **OK**.

   ATP compares Benchmark and Current images and displays the type of difference (“TEXT”) in the Difference column.
8. Right-click on the first line (Screen CICS0000).
9. Select **Browse**.

   ATP displays the **CICS0000 - Browse** window, and a screen image. The screen image contains the details of the employee added to the master file (Charles Rogers), just before updating.
10. Adjust the size of the window, so that it shows all of the image.
11. From the icon bar at the top of the window, click the **Differences** button.

ATP highlights the years for the dates. The current screen image differs from the benchmark screen image at the highlighted areas. This difference is to be expected, since the dates have been converted to include YYYY instead of YY.

12. From the icon bar, click the **Benchmark image** button.

    ATP shows the original screen image, where the years are only two digit. For example, the JOINED date is 08/14/97.

13. From the icon bar, click the **Current image** button.

    ATP shows the most recent screen image, where the years are four digit. For example, the JOINED date is now 08/14/1997.

14. From the icon bar, click the **Next image** button.

    ATP displays the screen image for the employee whose details were changed (Mary Gardener).

15. Click the **Differences** button.

    Once again, ATP highlights the differences.

    This time, ATP highlights only the years. The bug has been removed successfully from the CICS program.

16. Look at the benchmark and current images, and the other screen images.

17. Close the **Browser** window.

    The **CICSONE - Images** window reappears.

    At this point, the screen image comparison is complete.

    However, screen comparisons are always the current image against the benchmark image. The benchmark images are baseline images. The next screen image comparison is post-fix results against 19xx results. The benchmark images must be removed:

18. Select all the lines in the table.

19. Right-click on the highlighted lines.

20. Select **Archive**.

21. When the reassurance question appears, click **Yes**.

    The Current and Difference columns are cleared. The “Benchmark” images are now the Post-fix images. If you wish, you can verify this by browsing the images.

22. Close the **Image** window, and close ATP.

That completes the comparison of the screens.

10.5 Data Comparisons - SuperC

1. Execute job SUPERC1A.

   This job compares the master files.

2. Check the output of the job.

3. If the job failed, determine the reason, correct, and resubmit. Do not reject the job if the “SUPERC1A RUN” line shows an error code of 01.

The job produces the SuperC comparison report which includes the summary:

```
                 LINE COMPARE SUMMARY AND STATISTICS

14 NUMBER OF LINE MATCHES               0  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
 0 REFORMATTED LINES                    0  PAIRED CHANGES (REFM+PAIRED INS/DEL)
 0 NEW FILE LINE INSERTIONS             0  NON-PAIRED INSERTS
 0 OLD FILE LINE DELETIONS              0  NON-PAIRED DELETES
14 NEW FILE LINES PROCESSED
14 OLD FILE LINES PROCESSED
```

Figure 33. SuperC Summary Output for the Re-comparison

The top line shows that 14 lines matched—the comparison was completely successful.

10.6 Report Comparisons - SuperC

1. Execute job SUPERC1B.

   This job compares the two report files.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report output. This should show that 12 lines have matched. The reports have produced the same results.

The test results have now been successfully recreated to take into account the correctly converted program. The next process builds the aged testbed.

Chapter 11. Building the Aged Testbed

This process creates a set of files adjusted to reflect the Year 2000 changes made to the programs, and to include dates after 1999.

Input	Converted testbed
Output	Aged testbed

This process is similar to converting the distilled testbed to create the converted testbed (Chapter 7, "Building the Converted Testbed" on page 79), replacing the conversion routine with an aging routine.

The process is shown in Figure 34.

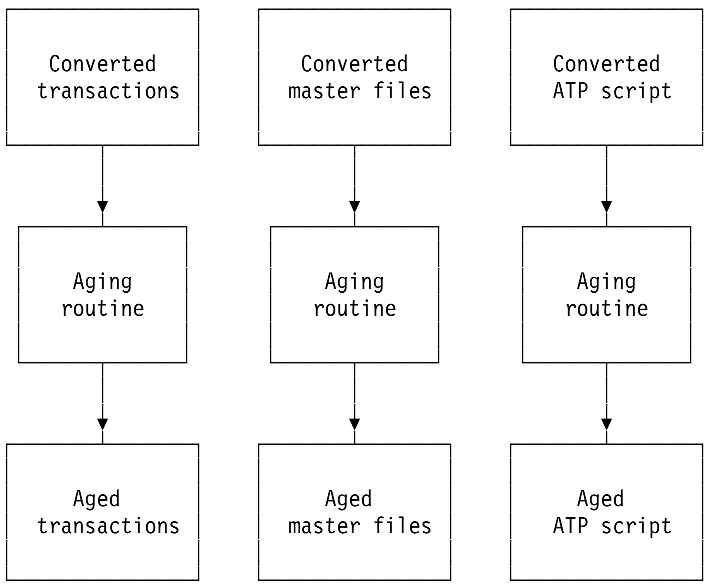


Figure 34. Aging the Converted Testbed to Produce the Aged Testbed

Since no transactions are used in the sample system, none need to be updated.

In this tutorial the ATP script and the master file are aged. This means that all information is properly synchronized.

You may prefer to only age transactions. To do so may produce unwanted side-effects, especially in the next stage, when you are running with a system clock set to a date after 1999. For example, you may find that items are flagged as "OVERDUE," when really they are "CURRENT." It is a matter of weighing the inconvenience of aging master files against the inconvenience of having an application produce unwanted results.

11.1 Creating a Testbed for Aging

Before you can age the converted testbed, you have to create a file for aging.

1. Execute REXX procedure TARDVSM3.

   This job creates EMP.MASTER.ONLINE3 from the backup of EMP.MASTER.ONLINE2.

11.2 Aging the Master File

1. Execute job TARAGE3T.

   This job ages EMP.MASTER.ONLINE3, leaving the result in the same file.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.

The job prints the aged master file, so that you can check the dates.

The JCL runs the program TARAGE3 to age all dates by the specified amount of years, which is the first parameter—in this case “03” specified on line **A** of the JCL (TARAGE3T) in Figure 35.

```
000012 //*    RUN TARAGE3  PROGRAM
000013 //TARAGE3   EXEC PGM=TARAGE3
000014 //EMPMAST  DD DSN=hlq.EMP.MASTER.ONLINE3,DISP=SHR
000015 //AGEPARM  DD *
000016 03  0228                       A
```

Figure 35. The Parameters to Control Aging

TARAGE3T ages a date by converting it from its storage format to the display format of “MM/DD/YYYY,” then adding the specified number to the years, then converting the date back to its storage format.

This approach runs into a problem when aging 29 February in a leap year to a non-leap year, for example aging 02/29/96 by five years. The second parameter on line **A** indicates how to handle this. This parameter has the value “0228,” which means age 29 February to 28 February in a non-leap year. The parameter could also be set to “0301,” which means to age the date to 1 March. If the parameter is left blank, then an error occurs if there is an attempt to age 29 February to a non-leap year.

An internal parameter sets the location of the window for dealing with windowed dates. For this system, the only windowed date is the security expiry date. The window used is a sliding window, which covers 30 years to the past, and 70 years to the future.

There are some security expiry dates which fall before the 30 year period. However, these dates apply only to employees from a long time ago. The report program that uses the security expiry date only looks at current employees, and for these employees, the security expiry date falls comfortably within the window. Strictly speaking, the information about long-gone employees should be removed from the current Employee Master file, and archived.

11.3 Backing up the Aged Master File

This job backs up the aged master file into a file called *hlq*.EMP.MASTON3.BACK

1. Execute REXX procedure TARBON3.

11.4 Aging the Auto Test Performer Script File

The equivalent WITT Year2000 for Windows procedure is found at B.6, "Aging the WITT Year2000 for Windows Script File" on page 140.

The ATP script file also needs aging, so that the transactions that are entered online are synchronized with the data from the master file.

The REXX routine TARADJST ages the dates in the script:

1. Open an OS/2 window.
2. Change to the ATP directory that holds the script file (for example, d:vaatptartranscicsone).
3. Rename the MAIN.TSF file to CONV.TSF; type rename main.tsf conv.tsf and press Enter.
4. Start the TARADJST Rexx procedure; type vataradjst and press Enter.
5. Enter the FULL source file name: CONV.TSF
6. Enter the source date format: MM/DD/YYYY
7. Enter the FULL target file name: MAIN.TSF
8. Enter the target date format: MM/DD/YYYY
9. Enter the number of years to age dates: 3

   This ages the transactions three years, the same number of years as the master file data is aged.

   The routine creates the file MAIN.TSF. This is identical to the original file, except that the years in dates are now three years "older." For example, the date "02/28/1997" becomes "02/28/2000."

   The message indicates that it has aged 8 records, which are the records containing dates.
10. Close the OS/2 window.

This script file is used in Chapter 12, "Creating the 19xx Results" on page 107, and in Chapter 14, "Creating the 20xx Results" on page 117.

Chapter 12. Creating the 19xx Results

This process creates a set of 19xx results, which are compared with the post-fix results (from Chapter 8, "Creating the Post-Fix Results" on page 85) and with the 20xx results (from Chapter 14, "Creating the 20xx Results" on page 117). The 19xx results reflect the state of the system after code has been changed to handle Year 2000 dates, and dates beyond 1999 have been included in the data. The results are produced by running programs "today."

Input Converted program code, aged testbed, aged ATP scripts

Output 19xx master files, reports, and screen images

This process is shown in Figure 36, and is the same as that for creating the post-fix results. The files used to drive the process are the aged files, in which at least some dates are now in 20xx.

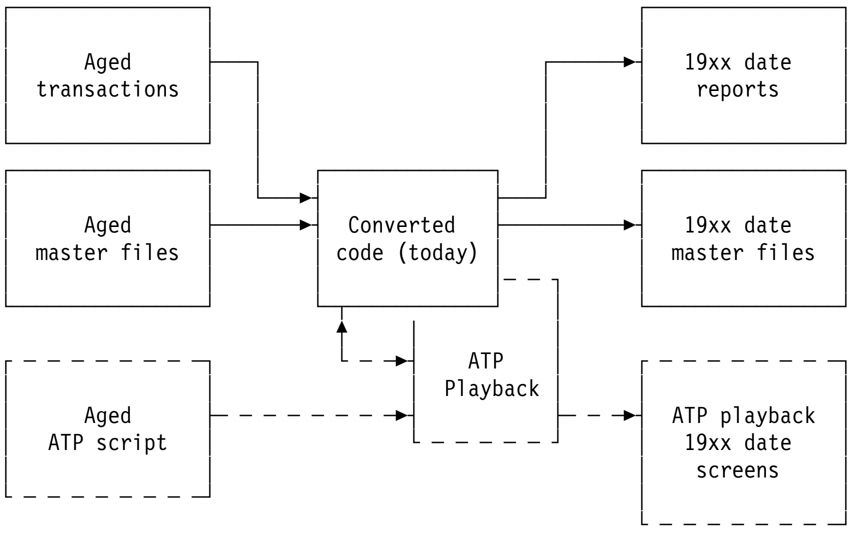


Figure 36. Creating the 19xx Results

12.1 Playback and Screen Capture - Auto Test Performer

This process is identical to the process used to create the post-fix results. The output is different only because the input is different.

This time Auto Test Performer (ATP) does the hard work. And, of course, the CICS program that is running is the converted program.

12.1.1 Starting the CICS Session

1. Start a CICS session.

   You must now make sure that the Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

The list of files starting with EMP appears.

For OS/390:

a. Find the line starting `Fil(EMPMST2)`.

   The file is pointing to the second version of the Employee Master file, and not the third. This must be changed.

b. Move the cursor to next line, and overtype the final digit, so that the DSN now reads *hlq*.EMP.MASTER.ONLINE3

c. Press Enter.

d. Move the cursor to the second status word for this file, and type `O`, and then press Enter.

   The status word changes to `Ope` (for Open).

   Figure 37 shows the CEMT file list at the end of this procedure.

```
I FI (EMP*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Fil(EMPMAST ) Vsa Clo Une Rea Upd Add Bro Del     Sha
        Dsn( hlq.EMP.MASTER.ONLINE                     )
 Fil(EMPMST2 ) Vsa Ope Ena Rea Upd Add Bro Del     Sha      NORMAL
        Dsn( hlq.EMP.MASTER.ONLINE3                    )
```

Figure 37. The CEMT File List for OS/390

For VSE:

a. Find the line starting `Fil(EMPMST3)`.

b. Move the cursor to the second status word for this file, and type `O`, and then press Enter.

   The status word changes to `Ope` (for Open).

c. Check the status of the other EMPMSTX files, and make sure that they are all closed. Close them if you need to.

   Figure 38 shows the CEMT file list at the end of this procedure.

```
I FI (EMP*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Fil(EMPMAST ) Vsa Clo Ena Rea Upd Add Bro Del
 Fil(EMPMST2 ) Vsa Clo Ena Rea Upd Add Bro Del
 Fil(EMPMST3 ) Vsa Ope Ena Rea Upd Add Bro Del
 Fil(EMPMST4 ) Vsa Clo Ena Rea Upd Add Bro Del
```

Figure 38. The CEMT File List for VSE

3. Press `PF3` to exit the CEMT display.
4. Clear the screen.
5. COBOL Start program `NB3X` COBOL

   PL/I Start program `MB3X` PL/I
6. Make sure that Caps Lock is on.

When the screen is displaying the Employee Update window, your CICS session is ready for you to enter data.

12.1.2 Using Auto Test Performer to Enter Transactions

The equivalent WITT Year2000 for Windows procedure is found at B.4, "Using WITT Year2000 for Windows to Enter Transactions" on page 137.

1. Start Auto Test Performer Manager.

   ATP starts, and the Manager window appears.

2. Click on the + sign at the left of the TARTRANS group icon.

   The testcases are listed under the group.

3. Double-click on the CICSONE testcase entry.

   The right of the screen lists three scripts and other items.

4. Right-click on the MAIN.TSF entry.
5. Select **Playback** from the pop-up menu.
6. If the **Playback Controls** window is not at the bottom of the screen, drag it to the bottom, where the buttons are not obscured by the CICS session.
7. If the CICS window is minimized, restore it to its normal size.
8. From the Playback Controls, click **Start/Resume playback**.

   ATP enters the recorded transactions into the CICS session.

   When the playback has finished, ATP displays the message that the script playback is completed.

9. Click **OK**.

12.1.3 Ending the CICS Session

1. In your CICS session, clear the window, then enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

2. For OS/390:

   - Find the line starting `Fil(EMPMST2)`.

   For VSE:

   - Find the line starting `Fil(EMPMST3)`.

3. Look along this line to the second status word, which comes after `Vsa`.
4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed). The file is now available for processing by other jobs.

5. Press `PF3` to exit the CEMT display.
6. Clear the screen.
7. Close the CICS session.

12.1.4 Viewing the Log File

The equivalent WITT Year2000 for Windows procedure is Steps 8 to 10 of B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Return to the **MAIN.TSF - Playback** window, and browse the log. Near the bottom, ATP has indicated that for the Window_Compares, a compare mismatch occurred. This is as expected, since the dates have all had 3 added to the year.
2. Close the **MAIN.TSF - Playback** window. Click **Yes** to save the current log.
3. Close ATP Manager.

   The next tutorial shows you how to use ATP to visually compare the screens.

You have now updated the Employee Master file, creating the 19xx results, ready for comparison with the post-fix results.

12.2 Creating a 19xx Report

This job creates the report TARRPA3.REPORT, which is the Security Card Expiry Dates report, for aged data.

The JCL is the same as for TARRPA2, except that it uses a different version of the master file.

1. Execute job TARRPA3.
2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report TARRPA3.REPORT.

   The report should list six employees, starting with Drummond Rapper and ending with Vi Sproika. Note that the dates now show a year of “01” and “02,” compared with previous reports showing “98” and “99.”

Chapter 13. Comparing the Post-Fix Results with the 19xx Results

This process compares the post-fix results with the 19xx results, to check that running Year 2000 dates through the system is not causing any problems.

Input Post-fix and 19xx files, reports, and screen images

Output Comparisons of files, reports, and screen images

Figure 39 shows the process of comparing the post-fix results and the 19xx results.

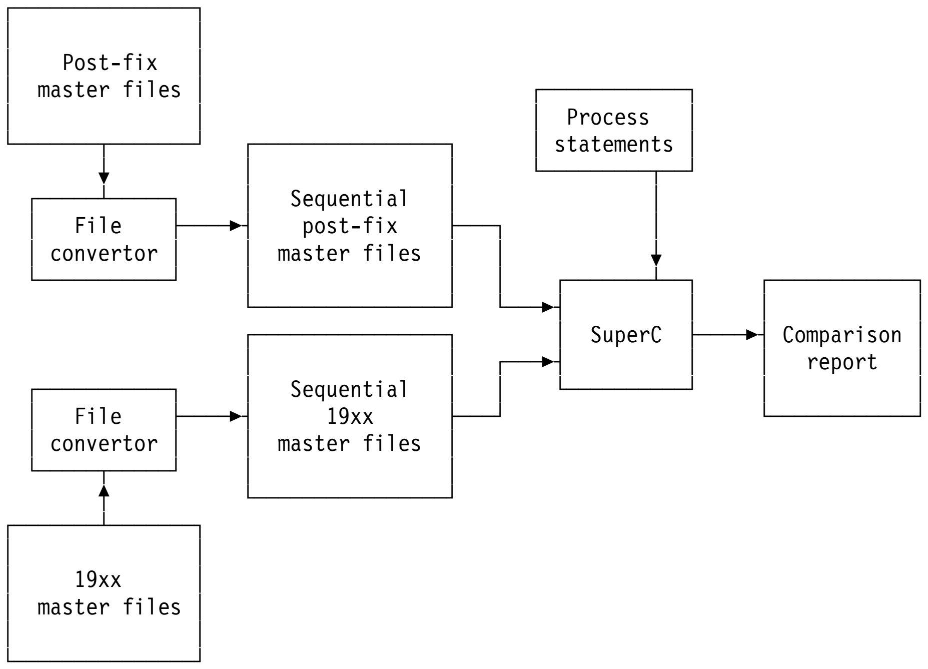


Figure 39. Comparing Post-Fix and 19xx Master Files

13.1 Screen Comparisons - Auto Test Performer

This tutorial uses Auto Test Performer (ATP) to compare the post-fix and 19xx screen images.

The equivalent WITT Year2000 for Windows procedure is found at B.5, “Screen Comparisons - WITT Year2000 for Windows” on page 138.

1. Open ATP Manager, and display the contents of the CICSONE testcase.
2. Double-click the IMAGES icon.

   The **CICSONE - Images** window appears.
3. Select all the lines in the table.
4. Right-click on the highlighted lines.
5. Select **Compare**.
6. If the **Screen image - Compare options** dialog box appears, click **OK**.

ATP compares Benchmark and Current images and displays the type of difference (“TEXT”) in the Difference column.

7. Right-click on the first line (Screen CICS0000).
8. Select **Browse**.

   ATP displays the **CICS0000 - Browse** window, and a screen image. The screen image contains the details of the employee added to the master file (Charles Rogers), just before updating.

9. Adjust the size of the window, so that it shows all of the image.
10. From the icon bar at the top of the window, click the **Differences** button.

    ATP highlights the years for the dates. The current screen image differs from the benchmark screen image at the highlighted areas.

    For three dates, the four digits of the year are highlighted, since the century has changed from “19” to “20.” For the fourth date, the birthdate, only the last two digits are highlighted, since the century has not changed.

11. From the icon bar, click the **Benchmark image** button.

    ATP shows the original screen image, where the years are not aged.

12. From the icon bar, click the **Current image** button.

    ATP shows the most recent screen image, where the years are aged.

13. From the icon bar, click the **Next image** button.

    ATP displays the screen image for the employee whose details were changed (Mary Gardener).

14. Repeat the process of looking at differences between the benchmark and the current image.

    In this case, only the last digit of the year is highlighted, because this is the only digit which changed when the information was aged.

15. Close the **Browser** window.

    The **CICSONE - Images** window reappears.

    The screen image comparison is complete, but the images have to be archived, so that the next ATP comparison will use the correct benchmark images.

16. Select all the lines in the table.
17. Right-click on the highlighted lines.
18. Select **Archive**.
19. When the reassurance question appears, click **Yes**.

    The Current and Difference columns are cleared. The “Benchmark” images are now the Post-fix images.

20. Close the **Image** window, and close ATP.

That completes the comparison of the screens.

13.2 Data Comparisons - SuperC

1. Execute job SUPERC2A.

   This job compares the master files. The job is similar to SUPERC1A except that the parameters have been changed to reflect the different structure of the files, and the OY2AGE parameter ages the dates in the old file by 3 years.

2. Check the output of the job.

3. If the job failed, determine the reason, correct, and resubmit. Do not reject the job if the “SUPERC2A RUN” line shows an error code of 01.

The job produces the SuperC comparison report, which includes the summary: table (Figure 40).

```
                 LINE COMPARE SUMMARY AND STATISTICS

 0 NUMBER OF LINE MATCHES             14  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
 0 REFORMATTED LINES                  14  PAIRED CHANGES (REFM+PAIRED INS/DEL)
14 NEW FILE LINE INSERTIONS            0  NON-PAIRED INSERTS
14 OLD FILE LINE DELETIONS             0  NON-PAIRED DELETES
14 NEW FILE LINES PROCESSED
14 OLD FILE LINES PROCESSED
```

Figure 40. SuperC Summary Output Comparing Post-Fix and 19xx Master Files

Looks like something has gone badly wrong. This output shows that there were no matches. There were problems with all 14 records!

Look carefully at the detailed output—you may find that it is easier to examine if you print the output report.

The output pairs records. Lines **1** to **6** pertain to the new record for Henry Stacker, and lines **7** to **12** lines pertain to the old record for Henry Stacker.

Figure 41 shows the first set of lines, the information for Henry Stacker.

```
I - 0000260001HENRY STACKER                14 CARGO COURT       1
INFO   Date cols 136:142  char 1973015       Comp=(1973015   )  2
INFO   Date cols 143:150  char 19990130      Comp=(1999 01 30)  3
INFO   Date cols 151:154  packed 2000218     Comp=(2000218   )  4
INFO   Date cols 155:158  packed 1951059     Comp=(1951059   )  5
INFO   Date cols 159:161  packed 01015       Comp=(2001015   )  6

D - 0000260001HENRY STACKER                14 CARGO COURT       7
INFO   Date cols 136:142  char 1970015       Comp=(1973015   )  8
INFO   Date cols 143:150  char 19960130      Comp=(1999 01 30)  9
INFO   Date cols 151:154  packed 1997217     Comp=(2000217   )  10
INFO   Date cols 155:158  packed 1948060     Comp=(1951060   )  11
INFO   Date cols 159:161  packed 98015       Comp=(2001015   )  11
```

Figure 41. The SuperC Details for Henry Stacker

The “Comp=” at the end of each line tells you how SuperC is interpreting the date, taking into account the description of each field, and any aging that has to be applied.

The first date "1973015" (lines **1** and **7**) is the same for both records, and the second date, "1999 01 30" (lines **2** and **8**), is also the same for both records. But the third date is "2000218" for the new record (line **3**), and "2000217" for the old record (line **9**). Likewise, the fourth date (lines **4** and **10**) differs by 1.

The explanation of this behavior lies in the way that the aging routine works. It ages all dates by converting them to a YYYY-MM-DD format, then adding three (the supplied parameter) to the year, then converting back to the storage format of the date. The problem arises when either the old or the new date fell in a leap year after 29 February, and the date is stored in a Julian format, "YYYYDDD." The number of days then differ by one, the leap "day," which is missing from one of the years but not the other.

If you work through all the records, you will find that the second date always compares correctly, because it is stored as "YYYYMMDD." You will also find that the third date is wrong by three years for Mary Gardener and Charles Rogers. This is because this is the maintenance date. In this comparison, the maintenance date has not been excluded from the comparison, as it was for the baseline to post-fix comparison.

For more discussion about dates, see Appendix A, "Other Matters to Consider Before You Test" on page 129.

The SuperC process statements are:

```
OY2C 136:142 YYYYDDD
NY2C 136:142 YYYYDDD
OY2C 143:150 YYYYMMDD,EMPTY
NY2C 143:150 YYYYMMDD,EMPTY
OY2P 151:154 YYYYDDD
NY2P 151:154 YYYYDDD
OY2P 155:158 YYYYDDD
NY2P 155:158 YYYYDDD
OY2P 159:161 YYDDD
NY2P 159:161 YYDDD
Y2PAST 1903                        11
OY2AGE 3                           12
```

Figure 42. SuperC Process Statements, Post-Fix to 19xx Data Comparison

- Pairs of statements for the old and new file describe the location and format of the dates as being identical. The comparison is between two files that have converted dates.
- The last statement, "OY2AGE" on line **12**, tells SuperC to age old dates by three years. So SuperC compares 1997 with 2000, and finds that they are equal.
- The "Y2Past" statement (line **11**) applies to dates with unexpanded years, which in this case is the SECURITY-EXP date. The window of 1903 to 2002 safely covers the dates for active employees.

The failure to match records illustrates the difficulties faced in Year 2000 testing. The matching failed for two reasons:

- When dates are held as "YYDDD," then the number of days may change when switching between leap years and non-leap years.

- When system dates are stored as data items, they are not be aged, although other information is.

When you find that records fail to match, you cannot assume that there are problems with your converted programs. Instead, you will have to try and work out the reasons for the mismatch, and decide whether the mismatches come from programs or testing methods.

13.3 Report Comparisons - SuperC

1. Execute job SUPERC2B.

   This job compares the Post-Fix and 19xx report files. The compare job is similar to SUPERC1B, but uses a Year 2000 feature, to age Post-Fix dates before comparing them with 19xx dates.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report output. This should show that 12 lines have matched. The reports have produced the same results.

The SuperC process statements are:

```
OEXCLUDE ROWS 1:2              1
NEXCLUDE ROWS 1:2              2
OY2AGE 3                       3
Y2PAST 1903                    4
OY2C 12:19 MM/DD/YY            5
NY2C 12:19 MM/DD/YY            6
```

Figure 43. SuperC Process Statements, Post-Fix to 19xx Report Comparison

By describing the expiry date (lines **5** and **6**), it is possible to age the old date three years (line **3**), and then compare identical dates. Lines **1** and **2** exclude the heading line, and line **4** sets the window for the dates (from 1903 to 2002).

Chapter 14. Creating the 20xx Results

This process creates a set of 20xx results, which can be compared with the 19xx results (from Chapter 12, “Creating the 19xx Results” on page 107). The 20xx results reflect the final state of the system. Code has been changed to handle Year 2000 dates, dates beyond 1999 have been included in the data, and the system is run with the system clock set beyond 1999.

Input Converted program code, aged testbed, aged ATP scripts, system clock set in 20xx

Output 20xx master files, reports, and screen images

This process, shown in Figure 44, is similar to that used to create the 19xx results.

The only difference is that the system clock is advanced to a date beyond 1999. If your application makes use of the system clock, then the advance in the system date should be the same as the number of years by which dates were aged.

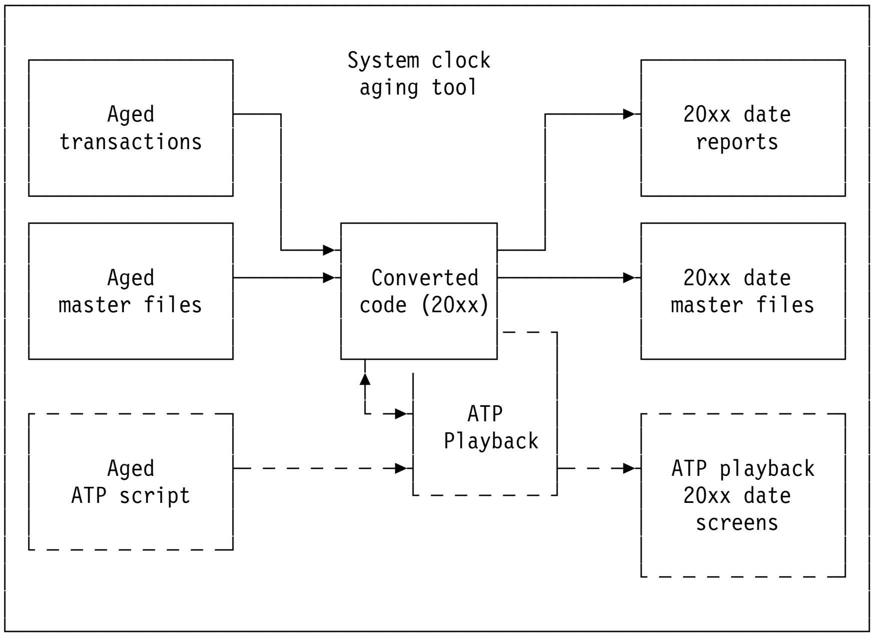


Figure 44. Creating the 20xx Results

14.1 Creating a Testbed for 20xx Testing

The current testbed has been converted and aged, and is ready for updating. However, you can not use the current version of the Employee Master file, since it has already been updated. Instead you must create a version prior to updating.

1. Execute REXX procedure `TARDVSM4`.

   This job creates EMP.MASTER.ONLINE4 from the backup of EMP.MASTER.ONLINE3.

You do not need to process this file any more because it is ready for updating. Neither do you need to back it up, since the file *hlq*.EMP.MASTON3.BACK is adequate.

Neither do you need to process the Auto Test Performer (ATP) script file any more because it is ready to run.

14.2 Date Simulation

At this point, you must now change the system date so that it falls beyond 1999.

If you are running the tutorials on an independent computer, the easiest way to change the system date is to IPL the computer, and start it with a new date. Make sure that this computer is completely separate from any production systems. You do not want a change of date compromising the integrity of production systems.

An alternative is to use a tool that simulates a different date. One such tool is TICTOC (IBM product order number 5620-BFV). TICTOC makes it possible to specify a virtual date to one job or a generic group of jobs, through an ISPF interface or through simple JCL in the job at execution.

There are other products available which simulate a different date. If you have one of these, you may prefer to use it. The same warning applies—do not compromise the integrity of a production computer.

Whatever means you use, make sure that the computer is running with a year of 2000 for this tutorial.

When you are testing your own applications, make sure that you age the system clock date the same amount as you have aged the dates in your files. For example, if you are converting programs in 1998, and you age dates by 4 years, set your system clock date to 2002.

14.3 Playback and Screen Capture - Auto Test Performer

This process is identical to the process used to create the 19xx results.

14.3.1 Starting the CICS Session

1. Start a CICS session.

   You must now make sure that the Employee Master file is open for the exclusive use of CICS.

2. Enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

   For OS/390:

   a. Find the line starting `Fil(EMPMST2)`.

      The file is pointing to the third version of the Employee Master file, and not the fourth. This must be changed.

   b. Move the cursor to next line, and overtype the final digit, so that the DSN now reads *hlq*EMP.MASTER.ONLINE4

c. Press Enter.

d. Move the cursor to the second status word for this file, and type 0, and then press Enter.

The status word changes to Ope (for Open).

For VSE:

a. Find the line starting Fil(EMPMST4).

b. Move the cursor to the second status word for this file, and type 0, and then press Enter.

The status word changes to Ope (for Open).

c. Check the status of the other EMPMSTX files, and make sure that they are all closed. Close them if you need to.

3. Press PF3 to exit the CEMT display.
4. Clear the screen.
5. COBOL Start program NB3X COBOL

   PL/I Start program MB3X PL/I
6. Make sure that Caps Lock is on.

When the screen is displaying the Employee Update window, your CICS session is ready for you to enter data.

14.3.2 Using Auto Test Performer to Enter Transactions

The equivalent WITT Year2000 for Windows procedure is found at B.4, "Using WITT Year2000 for Windows to Enter Transactions" on page 137.

1. Start Auto Test Performer Manager.

   ATP starts, and the Manager window appears.
2. Click on the + sign at the left of the TARTRANS group icon.

   The testcases are listed under the group.
3. Double-click on the CICSONE testcase entry.

   The right of the screen lists four scripts and other items.
4. Right-click on the MAIN.TSF entry.
5. Select **Playback** from the pop-up menu.
6. If the **Playback Controls** window is not at the bottom of the screen, drag it to the bottom, where the buttons are not obscured by the CICS session.
7. If the CICS window is minimized, restore it to its normal size.
8. From the Playback Controls, click **Start/Resume playback**.

   Your aged transactions are now entered into the CICS session.

   When the playback has finished, ATP displays the message that the script playback is completed.
9. Click **OK**.

14.3.3 Ending the CICS Session

1. In your CICS session, clear the window, then enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

2. For OS/390:

   - Find the line starting `Fil(EMPMST2)`.

   For VSE:

   - Find the line starting `Fil(EMPMST4)`.

3. Look along this line to the second status word, which comes after `Vsa`.
4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed). The file is now available for processing by other jobs.

5. Press `PF3` to exit the CEMT display.
6. Clear the screen.
7. Close the CICS session.

14.3.4 Viewing the Log File

The equivalent WITT Year2000 for Windows procedure is Steps 8 to 10 of B.4, “Using WITT Year2000 for Windows to Enter Transactions” on page 137.

1. Return to the **MAIN.TSF - Playback** window, and browse the log. There should be no evidence of compare mismatches.
2. Close the **MAIN.TSF - Playback** window. Click **Yes** to save the current log.
3. Close ATP Manager.

   The next tutorial shows you how to use ATP to visually compare the screens.

You have now updated the Employee Master file, creating the 20xx results, ready for comparison with the 19xx results.

14.4 Creating a 20xx Report

This job creates the report `TARRPA4.REPORT`, which is the Security Card Expiry Dates report.

The JCL is the same as for TARRPA3, except that it uses a different version of the master file.

1. Execute job `TARRPA4`.
2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report `TARRPA4.REPORT`.

   The report should list six employees, starting with Drummond Rapper and ending with Vi Sproika. Note that the dates now show a year of “01” and “02.”

14.5 Returning the Date to Today

At this point you should arrange for the system clock to be reset to the present day. The remaining tutorials can be carried out without reference to the system date.

Chapter 15. Comparing the 19xx Results with the 20xx Results

This process compares the 19xx results with the 20xx results, to check that running dates beyond 1999 through the system at a future time is not causing any problems.

Input	19xx and 20xx files, reports, and screen images
Output	Comparisons of files, reports, and screen images

Figure 45 shows the process of comparing the 19xx and 20xx master files.

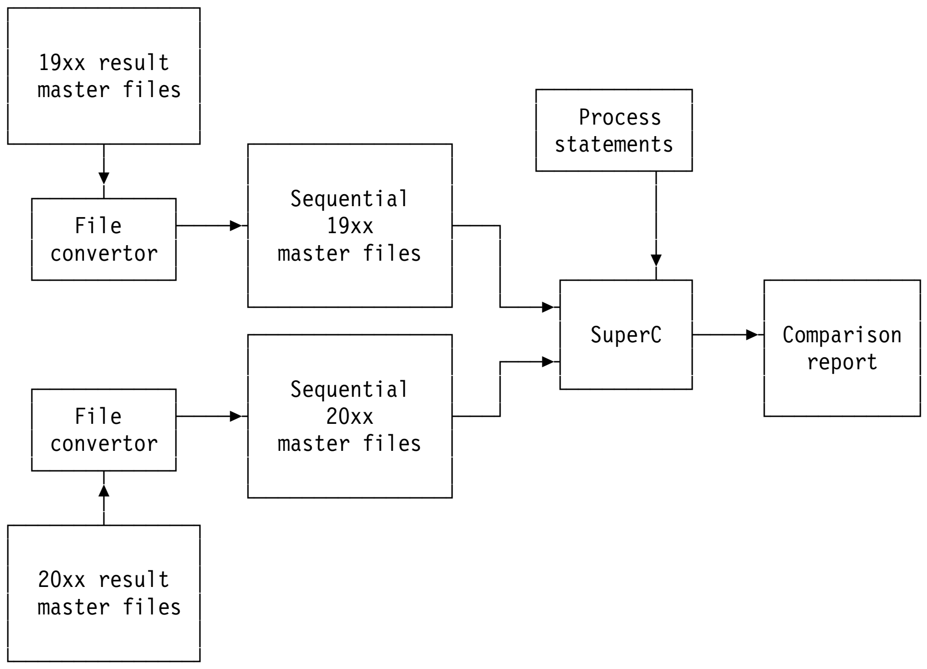


Figure 45. Comparing 19xx and 20xx Master Files

This process is the “cleanest” comparison, in that the result files should be identical. There is no need to make allowances for dates stored in different formats, or dates that are aged or not, because the process is comparing dates of the same format and same age.

The only possible difference in dates is the MAINTAINED date.

15.1 Screen Comparisons - Auto Test Performer

This tutorial uses Auto Test Performer (ATP) to compare the 19xx and 20xx screen images.

The equivalent WITT Year2000 for Windows procedure is found at B.5, “Screen Comparisons - WITT Year2000 for Windows” on page 138.

1. Open ATP Manager, and display the contents of the CICSONE testcase.
2. Double-click the IMAGES icon.

   The **CICSONE - Images** window appears.

3. Note that there are no entries in the “Current” column.

   There are no entries in this column, because the images that were snapped were identical to the benchmark images. The screens for 20xx are the same as those for 19xx.

4. Close the **Image** window, and close ATP.

That completes the comparison of the screens.

15.2 Data Comparisons - SuperC

1. Execute job SUPERC3A.

   This job compares the master files and is similar to SUPERC2A. The differences are:

   - It uses different input files.
   - The OY2AGE parameter has been removed, because both files have been aged by the same amount (3 years).
   - The OEXCLUDE and NEXCLUDE parameters remove comparison of columns 151 to 154. This removes the MAINTAINED date from the comparison.

2. Check the output of the job.

3. If the job failed, determine the reason, correct, and resubmit.

The job produces the SuperC comparison report which includes the summary:

```
                 LINE COMPARE SUMMARY AND STATISTICS

14 NUMBER OF LINE MATCHES               0  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
 0 REFORMATTED LINES                    0  PAIRED CHANGES (REFM+PAIRED INS/DEL)
 0 NEW FILE LINE INSERTIONS             0  NON-PAIRED INSERTS
 0 OLD FILE LINE DELETIONS              0  NON-PAIRED DELETES
14 NEW FILE LINES PROCESSED
14 OLD FILE LINES PROCESSED
```

Figure 46. SuperC Summary Output Comparing 19xx and 20xx Master Files

This table shows that every data item has matched.

The SuperC process statements are:

```
OEXCLUDE COLS 151:154
NEXCLUDE COLS 151:154
```

Figure 47. SuperC Process Statements, 19xx to 20xx Data Comparison

For this comparison everything should be the same, except possibly for the maintenance date, and so the two SuperC process statements exclude the maintenance date.

There is one potential problem with this approach. If the comparison fails, it will be difficult to work out the dates that were being compared. If this happens, you may prefer to describe the dates fully to SuperC.

15.3 Report Comparisons - SuperC

1. Execute job SUPERC3B.

   This job compares the 19xx and 20xx report files.

2. Check the output of the job.

3. If the job failed, determine the reason, correct, and resubmit.

4. Check the report output. This should show that 12 lines have matched. The reports have produced the same results.

The SuperC process statements are:

```
OEXCLUDE ROWS 1:2
NEXCLUDE ROWS 1:2
```

Figure 48. SuperC Process Statements, 19xx to 20xx Report Comparison

These process statements exclude the header lines.

Congratulations. You have completed the main tutorials for the Year 2000 test solution.

If you have not yet worked through the Debug Tool tutorials, you may wish to return to these, at 5.8, "Using Debug Tool to Check Coverage" on page 57 and 5.10, "Using Debug Tool to Build a Distilled Key List" on page 63.

The tutorials are designed so that they can be repeated again and again. Don′t install from the CD-ROM; just return to Chapter 5, "Developing Path Coverage and Distilling the Testbed" on page 37 or Chapter 6, "Creating the Baseline Results" on page 69, and start again.

Part 3. Appendixes

Appendix A. Other Matters to Consider Before You Test

This appendix briefly outlines some further matters that you should be aware of when you start planning your Year 2000 testing.

A.1 Critical Dates

When you are dealing with dates, you have to distinguish between two types of dates:

- System dates
- Dates stored as data

For system dates, there are two "must-test" dates. These are 31 December 1999, rolling over into 1 January 2000, and 28 February 2000, rolling over into 29 February 2000—2000 is a leap year. If you can run your system spanning these two times, you should do so.

For data dates, there are many more possibilities. The following dates are a minimum to test:

- 31 December 1999
- 1 January 2000
- 28 February 2000
- 29 February 2000
- 1 March 2000
- 31 December 2000
- 1 January 2001
- 31 December 2001
- 1 January 2002

However, for some of your systems, there may be other dates that are relevant. For example, you may want to include the end and beginning of your financial year, the end and beginning of financial quarters, dates when you send out statements, the first of each month, and so on.

As you develop path coverage of programs, you should find dates that have special relevance to your business. However, such dates may not reside in the programs that perform the every-day tasks of your business. They are more likely to reside in the special programs that are run every now and then, for example, programs that handle end-of-period processing.

A.2 Including Dates to Allow for Coverage After Conversion

A coverage tool provides the means to check that you have the maximum test coverage for programs.

When you apply the coverage tool to a legacy program, you are developing coverage for the program before conversion. But when you convert the program, it is possible that you increase the number of nodes in the program.

This applies particularly if you are using windowing since you have to include code that determines to which century the date belongs, based on the current window.

The data that you use to provide maximum coverage for the legacy program may no longer provide maximum coverage for the converted program.

After you develop the maximum coverage for a legacy program or system, you should add records to test the converted code fully. These dates are not required for coverage of the legacy system, but are essential for testing the converted system.

The alternatives to including extra data now are:

- Do not include the extra dates. This means that converted programs are not tested fully.
- Create a supplementary test suite and retest using this. Since you have to enter the supplementary dates either at the start or at the end of testing, it is better to add them at the start, and then you only have to perform the test process once.

Note that you if you check the coverage of the converted program and then add records to increase the coverage, you throw the testing process out, because the post-fix results are generated by more information than is provided for the baseline results.

A.3 How Much to Age?

In the tutorials, the information was aged three years.

When you are testing your systems, you are certain to find that you want to use a different aging value. It is likely that you will want to age the testbed by several different values. Only by looking carefully at your system will you be able to determine how much and how often you want to age your data.

The decision to age the data by three years in the tutorial was driven very much by the Security Expiry date. This date is projected about two years into the future and is the only date in the converted system that is stored as a windowed date. By aging three years, some of these dates were projected beyond 2000, and some were left in 1999. However, none of the birthdates were projected into 2000 or beyond.

To test the sample system thoroughly, it should be aged again, by maybe 20 or 30 years, so that many more dates will fall beyond 1999.

If you have a system where the day on which a date falls is critical, then you should look at aging it exactly 28 years. When you age by 28 years, every date ends up on the same day of the week.

A.3.1 Watch out for Leap Years

When you are aging data, be aware that leap years can cause problems.

An immediate problem is if you age 29 February in a leap year to a non-leap year, because this is not a valid date in a non-leap year. For example, the TARADJST routine supplied with the tutorials does not account for the leap year day. But there are other problems with leap and non-leap years, which are more subtle.

For example, if you store dates in the Julian format, when you age a date from a leap year to a non-leap year, or a non-leap year to a leap year, Julian dates after 29 February differ by 1. This was illustrated in Chapter 13, “Comparing the Post-Fix Results with the 19xx Results” on page 111.

A similar effect happens if your system depends on the number of days between two dates (for example, calculating interest on a daily basis). Once again, the interval may change by one day (increasing or decreasing depending on the position of the start date and end date of the interval. For example, the interval between 8 June 1997 and 8 June 1998 is 365 days, but the interval between 8 June 1999 and 8 June 2000 is 366 days.

A simple way of eliminating this effect is to age by a multiple of four years. In this way, dates within a leap year age to a leap year, and dates not in a leap year, age to a non-leap year. However, there may be occasions when aging by four years is not satisfactory. For example, the range of working dates may be less than four years, and if you age by four years, you may push all of your dates into the next century, which means that you don′t test your programs satisfactorily.

A.4 Skipping the 19xx Results

If your system relies on the system date for checking dates and for default date entries, then you may wish to skip creating the 19xx results, and go from creating the post-fix results to creating the 20xx results. Since the 19xx results are calculated with the system date set to “today,” the 19xx results may be different from post-fix results because the system detects that transaction dates are illegal (they are beyond today), or the status of transactions changes (they become overdue).

If you skip the 19xx results, the method still works. You just compare post-fix results with 20xx results. However, if the comparisons fail, you will have to determine whether the failure was caused by aging data, or by the system being unable to run with a system date that is set past 1999.

A.5 Stress and Performance Testing

Your Year 2000 changes create more data (expanded dates) or more lines of code to execute (for example, to handle windowing). As a result, the converted programs may take more time to run, or your data may take up more storage space.

If you have plentiful resources, then these changes are not liable to have much impact when they are put into production. But if resources are currently limited, you may wish to run stress and performance testing, before changes are put into production.

When you undertake stress and performance testing, you are seeking to establish the increase in resource usage. Ultimately, you are trying to establish whether your existing resources will be able to cope with the increase. If not, you will have to either obtain more of the appropriate resource, or else, adjust the changes you have made to programs.

It may be that when you work out the methods for converting programs, you choose a conversion method that suits your resource mix. However, if resources are potentially a problem, you should check by running stress and performance testing.

Appendix B. Using WITT Year2000 for Windows

An alternative to using Auto Test Performer (ATP) to create scripts and capture and compare screen images is to use WITT Year2000 for Windows, which works under Windows 95 and Windows NT.

This appendix provides the WITT Year2000 for Windows equivalents to the ATP procedures provided in the tutorials. Signposts within the tutorial procedures point to the relevant procedures of this appendix.

B.1 Starting WITT Year2000 for Windows

This procedure replaces 6.1.4, "Starting Auto Test Performer" on page 72.

1. Select **Programs** from the **Start** menu.
2. Select **IBM WITT Year2000 for Windows**.
3. Select **IBM WITT Year2000**.

   WITT Year2000 for Windows is now started, and the main window appears.

   The left side of this window is the Tree View, and the right side is the List View.

Having started WITT Year2000 for Windows, you now have to create a test project, a test unit, and a test script. This is a hierarchical structure, holding the information that is captured as you enter the CICS details.

B.1.1 Creating the Test Project

The test project is a Windows folder.

1. From the **File** menu, select **New Test Project...**

   The **Create New Test Project** dialog box appears.

2. Enter `TARTRANS`.

   The List View shows that two items are created automatically: Control Definitions and class.

B.1.2 Creating a Test Unit and Script

A test unit is a convenient way of holding in one place all the scripts to be applied against one program.

1. Select the TARTRANS Test Project.
2. From the **File** menu, select **New**, then select **Test Unit**.

   The new test unit is created in the List View.

3. Type `CICSONE` as the name, and press Enter.
4. Click on the CICSONE test unit in the Tree View (left window).

   The script New Test Script appears in the List View.

5. Rename the script to `Script One` by typing

   ```
   Script One
   ```

   over the top of the original name and pressing Enter.

6. Click on the CICSONE test unit in the Tree View (left window).
7. From the **File** menu, select **Properties**.

   The **CICSONE Properties** dialog box appears.
8. On the Test Script tab, click **Host**, to indicate that the recording session is from a host computer, and not from the workstation.
9. If the "Generates benchmark names automatically" box is *not* selected, select it.
10. Click OK.
11. Double-click `Script One`.

    The **Script One - Script Editor** window appears.
12. From the **Script** menu, select **Start Record**.

    The **Recorder Open** dialog box appears.
13. Select the appropriate Session ID (for example, `B`).
14. Click OK.

You can now return to 6.1.5, "Enter the Transactions" on page 73 to continue creating the baseline results.

B.2 Looking at the Script

This procedure replaces 6.1.8, "Looking at the Script" on page 76.

The script that you have created in WITT Year2000 for Windows is a text file which you can edit:

1. Activate and maximize the **Script One - Script Editor** window.

   The script is in the top (Object:) window.
2. Look at the script.

   You should be able to see where you pressed the Tab key or Enter key, or typed a date. With a little more effort, you should also be able to see where you entered a field.

   The two "ThWindowCompare" lines come from the screen captures.

   When you entered the script, it is possible that you may have typed a wrong key, and then pressed Backspace or Left Arrow to move the cursor to replace the wrong keystroke. If you did, the routine that converts the script will not convert everything, as it is expecting dates to be in "MM/DD/YY" format.

   Check your script, and make sure that all your dates are properly entered. If not, edit the script to make sure that they are. (This normally involves deleting a few lines, to "close up" the date information.)

   The following sample shows the script when the start of the JOINED date is entered as "08/41." The mistake was noticed and corrected by back spacing and typing the day "14," before adding "/97."

   ```
   ThEnterText "08/41"
   ThEnterKey "<BCKSP>", 2
   ThEnterText "14/97"
   ```

This mistake is corrected by deleting from (and including) ′41′ on the first line up to (and including) the ″ before ′14′ on the third line, to consolidate the date entry into one line:

```
ThEnterText ″08/14/97″
```

3. Close the **Script Editor** window.

   The **Save Test Script** dialog box appears.

4. Click **Yes** to save the script.
5. Close the WITT Year2000 for Windows main window.

In the process you have just completed (from B.1, "Starting WITT Year2000 for Windows" on page 133 through B.2, "Looking at the Script") you achieved three things:

- You used WITT Year2000 for Windows to capture data entry, and thus create a script which you can play back with WITT Year2000 for Windows.
- You used WITT Year2000 for Windows to capture screen images, which you can use for comparison.
- You updated the Employee Master file, and along the way exercised the update program.

The updated Employee Master file is the baseline result that you will later compare with the post-fix results.

The script is going to be converted, so that it can be used to provide data entry to the converted code.

You can now return to 6.2, "Creating a Baseline Report" on page 77 to create the baseline report.

B.3 Converting the WITT Year2000 for Windows Script File

This procedure replaces 7.3, "Converting the Auto Test Performer Script File" on page 82.

The WITT Year2000 for Windows script file `Script One` is a text file. The data it holds includes dates in the "MM/DD/YY" format. The converted CICS screen has dates entered in "MM/DD/YYYY" format. If you replay `Script One` against the converted CICS screen, then you will experience problems.

The following procedure explains how to convert Script One so that all the dates in the script file appear in the correct format:

1. Copy the file `Taradjst.tsc` from the directory `d:\va\` to the directory `d:\ibmwittw\Bin\Tartrans\Cicsone\`, adjusting the drive letter if necessary.

   The first directory is where the file `Taradjst.tsc` should have been copied as part of the installation (see 3.4, "Installing Files on Windows 95 or Windows NT" on page 22). If you can′t find the file on your PC, you can also copy it from the installation CD-ROM, from directory `e:\windows`.

   The second directory is where your WITT Year2000 for Windows scripts are stored. If you can′t find this directory, use Windows Find to look for the file `Script One.tsc`, which is the script.

2. Start WITT Year2000 for Windows.

3. From the **File** menu, select **Open Test Project...**

   The **Open Test Project** dialog box appears.

4. Select TARTRANS.vap and OK.
5. Double-click the CICSONE test unit.

   The components of the test unit appear.

   The component list should now include the script Taradjst.tsc.

6. If the script Taradjst.tsc is not included, within Windows, rename the file to Tar_adjst.tsc, then shut down the CICSONE test unit, then re-open the CICSONE test unit. The new script Tar_adjst.tsc should now be listed. Rename back to Taradjst.tsc, and close and re-open the CICSONE test unit.
7. Double-click the TARADJST.TSC script.

   The **Taradjst - Script Editor** window appears.

8. From the **Script** menu, select **Start Playback**.

   The **Playback Status** window appears. As well, the **InputBox** window for a batched request file appears.

9. Press Cancel.

   The request for the source file name appears.

   The batched request file is a text file. Each set of five lines holds information about one file that is to be converted (where the five lines correspond to the five questions that are asked when information is entered manually). Any line preceded by an asterisk (*) is a comment line, and is disregarded.

10. Enter the full source file name, and click OK. The full source file name is d:\ibmwittw\bin\tartrans\cicsone\script one.tsc, adjusting the drive letter if necessary.

    The request for the source date format appears.

11. Enter the source date format MM/DD/YY and click OK.

    The request for the target file name appears.

12. Enter the full target file name, and click OK. The full target file name is d:\ibmwittw\bin\tartrans\cicsone\script two.tsc, adjusting the drive letter if necessary.

    The request for the target date format appears.

13. Enter the target date format MM/DD/YYYY and click OK.

    The request for the number of years to age the script appears.

14. Enter 0 and press OK.

    For this script, the transactions are not aged.

    The request for the source file name appears again.

15. Press Cancel.

    A report on the script run appears.

16. Close the report without saving it.

    The CICSONE testcase now holds the additional script, Script two. You can open this script and look at it. The script is identical to Script One, except that all dates now have a year that is four digits, instead of two.

This script file is used in Chapter 8, “Creating the Post-Fix Results” on page 85. You can return there to start creating the post-fix results.

B.4 Using WITT Year2000 for Windows to Enter Transactions

This procedure replaces:

- 8.1.2, “Using Auto Test Performer to Enter Transactions” on page 86 (Chapter 8, “Creating the Post-Fix Results”)
- 10.2.2, “Using Auto Test Performer to Enter Transactions” on page 98 (Chapter 10, “Restarting Testing”)
- 12.1.2, “Using Auto Test Performer to Enter Transactions” on page 109 (Chapter 12, “Creating the 19xx Results”)
- 14.3.2, “Using Auto Test Performer to Enter Transactions” on page 119 (Chapter 14, “Creating the 20xx Results”)
- 5.8.4, “Using Auto Test Performer to Enter Transactions” on page 59 (5.8, “Using Debug Tool to Check Coverage”)

This procedure uses WITT Year2000 for Windows to enter transactions into a CICS program, and along the way captures screen images for later comparison.

1. Select **Programs** from the **Start** menu.
2. Select **IBM WITT Year2000 for Windows**.
3. Select **IBM WITT Year2000**.

   WITT Year2000 for Windows is now started, and the main window appears.
4. From the **File** menu, select **Open Test Project...**

   The **Open Test Project** dialog box appears.
5. Select `TARTRANS.vap` and click OK.
6. Double-click the CICSONE test unit.

   The components of the test unit appear.
7. The script that you now invoke depends on what you are doing:

   - If you are using Debug Tool to check coverage, double-click `Script One`.
   - If you are creating the post-fix results, or restarting the testing, double-click `Script two`.
   - If you are creating the 19xx results, or creating the 20xx results, double-click `Script three`.

   The **Script Editor** window appears.
8. From the **Script** menu, select **Start Playback**.

   Your transactions are now entered into the CICS session.

   When the playback has finished, the **Report - Test Report** window appears.
9. Note the results of the window captures, shown in the Event List, the window at the bottom right:

   - For post-fix, restart, and Year 19xx results, the compares Fail.
   - For Year 20xx results, the compares Pass.

These results are explained more fully in the following section B.5, “Screen Comparisons - WITT Year2000 for Windows.”

10. Close the Test Report, saving the report to the default name.
11. Close the Script Editor.
12. Close the WITT Year2000 for Windows main window.

You can now return to:

- 8.1.3, “Ending the CICS Session” on page 87 to continue creating post-fix results.
- 10.2.3, “Ending the CICS Session” on page 98 to continue creating post-fix results, after the code correction and restart.
- 12.1.3, “Ending the CICS Session” on page 109 to continue creating 19xx results.
- 14.3.3, “Ending the CICS Session” on page 120 to continue creating 20xx results.
- 5.8.5, “Looking at the Frequency Counts” on page 60 to look at the frequency counts from the Debug Tool session.

B.5 Screen Comparisons - WITT Year2000 for Windows

This procedure replaces:

- 9.1, “Screen Comparisons - Auto Test Performer” on page 90 (Chapter 8, “Creating the Post-Fix Results”)
- 10.4, “Screen Comparisons - Auto Test Performer” on page 99 (Chapter 10, “Restarting Testing”)
- 13.1, “Screen Comparisons - Auto Test Performer” on page 111 (Chapter 13, “Comparing the Post-Fix Results with the 19xx Results”)
- 15.1, “Screen Comparisons - Auto Test Performer” on page 123 (Chapter 15, “Comparing the 19xx Results with the 20xx Results”)

This procedure compares two sets of screen images.

1. Open WITT Year2000 for Windows Manager, and display the contents of the TARTRANS testcase.
2. Double-click the CICSONE test unit.
3. Double-click the latest Test Report.

   The **Test Report** window appears.
4. Double-click the first ThWindowCompare command in the Event List.

   The **Host Image Result** window appears.

   The **Host Image Result** window shows the difference between the benchmark and current image, as well as the benchmark image and the current image.

   The **Host Image Result** window doesn′t appear if there are no differences in the images, which should be the case when comparing 19xx and 20xx windows. In this case, you can look at the screen image by double-clicking it where it is listed in the contents of the test unit.

5. Maximize the window, and increase the image display area, until all the screen image appears.

6. Click on the Difference tab, if the **Difference** window is not displayed.

   This image shows the information for Charles Rogers. The differences are shown in reverse video.

   The differences are:

   - For benchmark to post-fix (before and after fixing)

     In the benchmark image, the years are two digit, and in the current (post-fix), they are four digit.

   - For post-fix to 19xx

     Aging has changed every year from a 19xx date to a 20xx date, so all four digits in the year are changed, except for BIRTHDATE, where only the last two digits are changed.

   - For 19xx to 20xx

     There should be no difference in the images. If the report shows a match, with a blue circle, when you double-click on the line, nothing happens.

7. Close the **Host Image Result** window.

8. Double-click the second ThWindowCompare command in the Event List.

   The **Host Image Result** window appears.

9. Maximize the window, and increase the image display area, until all the screen image appears.

10. Click on the Difference tab, if the **Difference** window is not displayed.

    This image shows the information for Mary Gardener.

    The differences are as for the first image, except:

    - For benchmark to post-fix (before fixing)

      As well as highlighting the years, the second digit of the month and day in the SECURITY CARD EXPIRY date is highlighted.

      Check the benchmark and current images, to see that this date changes from “06/07/96” to “01/03/98.” This is possibly a problem with the conversion of the program.

11. Close the **Host Image Result** window.

If you are comparing baseline results with post-fix results for the first time (and not as part of restarting testing), then return to 9.2, “Data Comparisons - SuperC” on page 92. Otherwise, continue with this procedure.

Once the screen comparison is complete and there is no need to retain benchmark images, you must archive them. Archiving removes the benchmark image, and moves the current image to the benchmark, ready for comparison when the next current image is captured.

1. Double-click the first ThWindowCompare command in the Event List.

   The **Host Image Result** window appears.

2. Select **Archive** from the **File** menu.

3. Click **Yes** in response to the question about really wanting to archive.

4. Close the **Host Image Result** window.
5. Repeat the process for the second ThWindowCompare command.
6. Close the **Test Report** window.
7. Close the WITT Year2000 for Windows main window.

You can now return to:

- 10.5, "Data Comparisons - SuperC" on page 100 to continue comparing benchmark and post-fix results.
- 13.2, "Data Comparisons - SuperC" on page 113 to continue comparing post-fix and 19xx results.
- 15.2, "Data Comparisons - SuperC" on page 124 to continue comparing 19xx and 20xx results.

B.6 Aging the WITT Year2000 for Windows Script File

This procedure replaces 11.4, "Aging the Auto Test Performer Script File" on page 105.

The procedure explains how to convert Script Two so that all the dates in the script file are aged by three years:

1. Start WITT Year2000 for Windows.
2. From the **File** menu, select **Open Test Project...**

   The **Open Test Project** dialog box appears.
3. Select `TARTRANS.vap` and OK.
4. Double-click the CICSONE test unit.

   The components of the test unit appear, including the script `Taradjst.tsc`.
5. Double-click the TARADJST.TSC script.

   The **Taradjst - Script Editor** window appears.
6. From the **Script** menu, select **Start Playback**.

   The **Playback Status** window appears. As well, the **InputBox** window for a batched request file appears.
7. Press Cancel.

   The request for the source file name appears.
8. Enter the full source file name, and click OK. The full source file name is `d:\ibmwittw\bin\tartrans\cicsone\script two.tsc`, adjusting the drive letter if necessary.

   The request for the source date format appears.
9. Enter the source date format `MM/DD/YYYY` and click OK.

   The request for the target file name appears.
10. Enter the full target file name, and click OK. The full target file name is `d:\ibmwittw\bin\tartrans\cicsone\script three.tsc`, adjusting the drive letter if necessary.

    The request for the target date format appears.
11. Enter the target date format `MM/DD/YYYY` and click OK.

The request for the number of years to age the script appears.

12. Enter 3 and press OK.

    For this script, the transactions are aged three years.

    The request for the source file name appears again.

13. Press Cancel.

    A report on the script run appears.

14. Close the report without saving it.

    The CICSONE testcase now holds the additional script, Script three. You can open this script and look at it. The script is identical to Script One, except that all dates are aged by three years.

This script file is used in Chapter 12, “Creating the 19xx Results” on page 107, and in Chapter 14, “Creating the 20xx Results” on page 117. You can return to Chapter 12, “Creating the 19xx Results” on page 107 to start creating the 19xx results.

Appendix C. The Slide-Show Demonstrations

The slide-show demonstrations provide a quick way for you to look at some of the products, without needing to install or set up anything.

There are four primary demonstrations:

- The overview lists the testing processes and the tools relevant to the process, and shows about half a dozen screens of each tool at work.
- The detailed product demonstrations (for COBOL Tester, Auto Test Performer, and WITT Year2000 for Windows) work through the steps of the tutorials, and show appropriate screen captures and an explanation.

  Because these product demonstrations are based on the tutorials, they make an ideal introduction to the tutorials. The demonstrations don′t work through every step of the tutorial, nor are they "how-to" training demonstrations. Instead, they focus on what each package can give you, especially with relevance to Year 2000 testing.

Each demonstration lasts about five minutes.

The slide-show runs directly from the CD-ROM. It runs under Windows 3.1 (and hence Win-OS/2), Windows 95, and Windows NT.

C.1 Starting the Slide-Show

1. Insert the CD-ROM into your CD-ROM drive.
2. Find the file VA2000T.EXE, on the directory **demo** of the CD-ROM.
3. Start the program.
   - If you are running Win-OS/2, run the program by finding it under File Manager, and double-clicking on it. Running it directly from an OS/2 prompt lengthens automatic advancement pauses.
   - If you are running under a native Windows, you can run the program through the Run command. Run `g:\demo\va2000t.exe`, where `g` is the CD-ROM drive letter.
   - You can′t run the program directly from a DOS window.

C.2 Working the Demonstration

The slide-show consists of screens of text, and screens that are a screen dump plus annotation (in a blue text box).

- To move forward from one screen to the next, press PgDn.
- To move backwards to the previous screen, press PgUp.
- For alternative navigation methods, press F1 while running a demonstration.
- If the text finishes with **...** the slide-show advances automatically to the next screen after an appropriate pause. You can also advance manually if you want.
- The main menu lets you select your demonstration, by clicking on a line, or typing a number.

If you wish, you can copy the files from the CD-ROM to your hard disk, so that you can look at a demonstration at your convenience.

Appendix D. A Taste of The Millennium Language Extensions

One of the most popular ways of changing applications to address the Year 2000 problem is by using a windowing technique when processing 2-digit years in dates. The windowing technique addressed the Year 2000 problem by removing the assumption that all 2-digit year fields represent years from 1900 to 1999. Instead, the technique enables 2-digit year fields to represent years within any 100-year range from 1900 to 2098.

For example, if a 2-digit year field contains the value 15, most current applications would interpret the year as 1915. However, if the window of valid years is 1960 to 2059, then the year is interpreted as 2015.

With this concept in mind the Millennium Language Extensions (MLE) have been added to COBOL and PL/I Year 2000-ready compilers.

The primary objective of the MLE enhancements is to extend the useful life of existing programs using 2-digit years. Program source changes are held to the minimum, and the data itself is preserved in its current non-expanded form.

This appendix shows an example of a program converted to use MLE. The sample is not meant to be a comprehensive exposition of MLE and how it works. Instead, the sample provides you with a snippet to whet your appetite—a taste of MLE.

The sample program is the program used in the tutorials to provide a report. The COBOL version is based on TARMU5, and the PL/I version is based on PL1MU5. The converted MLE version is called MLEMU5 for both languages.

The following listings show changed segments of the programs. Lines starting with “>” are source statement lines, and lines tagged with **C** are MLE modifications. Explanatory text follows the source lines. The complete listings are available as part of the VA2000 files loaded from the CD-ROM.

The code listings are followed by listings of the report output before and after modification.

D.1 A COBOL Example

The sample uses a windowing compiler option, and the DATE FORMAT “move” statements, to expand the 2-digit year to its 4-digit year equivalent.

D.1.1 Compiler Directing Statements

```
>  C     CBL DATEPROC YEARWINDOW(1935)
```

`DATEPROC` enables the MLE extensions.

`YEARWINDOW(base-year)` sets the window parameters. `base-year` is the first year of the 100-year window. In this case the 100 year window is from 1935 - 2034.

D.1.2 Data Division clauses

```
>        01  WORK-DATE-YYMMDD.
>   C          03  WORK-DATE-YY  PIC XX DATE FORMAT IS YY.
>            03  WORK-DATE-MM  PIC XX.
>            03  WORK-DATE-DD  PIC XX.


>        *    ** report headings & detail line
>         01  REPORT-HEAD1.
>             03  FILLER                      PIC X(01)  VALUE ″1″.
>             03  FILLER                      PIC X(48)  VALUE
>                 ″REPORT ID  MLEMU5″.
>             03  FILLER                      PIC X(62)  VALUE
>                 ″EMPLOYEE DETAILS REPORT″.
>             03  RPT-HD1-DATE.
>                 05  RPT-HD1-MM              PIC XX.
>                 05  FILLER                  PIC X      VALUE ″/″.
>                 05  RPT-HD1-DD              PIC XX.
>                 05  FILLER                  PIC X      VALUE ″/″.
>    C              05  RPT-HD1-YY  PIC X(4) DATE FORMAT IS YYYY.
>             03  FILLER                      PIC XX     VALUE SPACES.
>             03  RPT-HD1-TIME                PIC X(8).
>             03  FILLER                      PIC XX     VALUE SPACES.


>         01  TARDATE-PARAMETERS.
>             03  TARDATE-DATE.
>                 05  TARDATE-MMDD            PIC X(6).
>    C              05  TARDATE-YY PIC X(2) DATE FORMAT IS YY.


>         01  REPORT-DETAIL.
>             03  FILLER                      PIC X(01)  VALUE ″0″.
>             03  FILLER                      PIC X(10)  VALUE SPACES.
>             03  RPT-DET-SEC-EXP.
>                 05  RPT-DET-MMDD            PIC X(6).
>    C              05  RPT-DET-YY PIC X(4) DATE FORMAT IS YYYY.
>             03  FILLER                      PIC X(04)  VALUE SPACES.
>             03  RPT-DET-ID                  PIC X(06).
>             03  FILLER                      PIC X(04)  VALUE SPACES.
>             03  RPT-DET-NAME-ADDR           PIC X(30).
>             03  FILLER                      PIC X(04)  VALUE SPACES.
>             03  RPT-DET-ERROR               PIC X(30).
>             03  FILLER                      PIC X(34)  VALUE SPACES.
```

`DATE FORMAT IS` indicates that a data item is a date field or year field. The year field can be represented in a non-expanded form (YY) or in an expanded form (YYYY).

D.1.3 Procedure Division ′MOVE′ Statements

```
⋮
>    C       MOVE WORK-DATE-YY          TO RPT-HD1-YY.
⋮
>    C       MOVE TARDATE-YY            TO  RPT-DET-YY.
```

These move statements move a windowed year field to an expanded year field. The move is performed with the value in ′...YY′ right justified in YYYY, which is prefixed with the appropriate century.

D.2 A PL/I Example

The sample uses a windowing compiler option, and the new attributes of the DATE built-in function. Implicit date conversions take place when 2-digit year fields are assigned to 4-digit year fields.

D.2.1 Compiler Directing Statements

```
>  C    *PROCESS SOURCE,LIST,STMT,RESPECT(DATE),WINDOW(1935)
```

RESPECT(DATE)causes the compiler to honor any specification of the DATE attribute when assigning the result of the DATE built-in function.

WINDOW(1935) represents a 100 year window commencing at 1935.

D.2.2 Builtin Function

```
>  C    DCL DATE                 BUILTIN;
```

This built-in function has been extended to support additional date patterns, such as YY, YYYY, YYMM, and YYYYMM.

D.2.3 Variable Declarations

```
>       DCL 1 REPORT_DETAIL,
>             2  RD_CTL                CHAR(1) INIT(′0′),
>             2  R1                    CHAR(10) INIT(′          ′),
>             2  RPT_DET_SEC_EXP,
>                3  RPT_DET_MMDD       CHAR(6),
>  C               3  RPT_DET_YY         CHAR(4) DATE(′YYYY′),
>             2  R2                    CHAR(5)  INIT(′     ′),
>             2  RPT_DET_ID            CHAR(6),
>             2  R3                    CHAR(4)  INIT(′    ′),
>             2  RPT_DET_NAME_ADDR     CHAR(30),
>             2  R4                    CHAR(4)  INIT(′    ′),
>             2  RPT_DET_ERROR         CHAR(30),
>             2  R5                    CHAR(33) INIT(REPEAT(′ ′,33));


>       DCL 1 PARAM              CHAR(54) EXTERNAL;
>       DCL 1 PARAM_RED DEF PARAM,
>             2 INPUT_DATE,
>               3 INPUT_DATE_MMDD CHAR(6),
>  C              3 INPUT_DATE_YY   CHAR(2) DATE(′YY′),
>             2 INPUT_FORMAT     CHAR(8),
>             2 OUTPUT_FORMAT    CHAR(8),
>             2 MSG              CHAR(30);
```

Variables have the new DATE attributes (YY and YYYY).

D.2.4 Variable Assignments

```
>       RPT_DET_YY          = INPUT_DATE_YY;
```

This assignment will implicitly expand the 2-digit source date to a 4-digit target date, using the windowed year values of 1935-2034.

D.3 Report Outputs

Here is the output from the original program TARMU5/PL1MU5:

```
REPORT ID  xxxMU5       EMPLOYEE DETAILS REPORT      12/09/97  09:14:34

          SEC/EXPIRED   ID        NAME

          01/01/01      000313    DRUMMOND RAPPER

          04/30/01      002131    RICK LAYER

          02/28/02      044026    ALISON CLARKE

          06/16/02      000101    PETER SELLER

          06/22/02      006101    PENNY MARKER

          08/31/02      002292    VI SPROIKA
```

Here is the output from the converted program MLEMU5:

```
REPORT ID  MLEMU5       EMPLOYEE DETAILS REPORT      12/09/1997  09:18:13

          SEC/EXPIRED   ID        NAME

          01/01/2001    000313    DRUMMOND RAPPER

          04/30/2001    002131    RICK LAYER

          02/28/2002    044026    ALISON CLARKE

          06/16/2002    000101    PETER SELLER

          06/22/2002    006101    PENNY MARKER

          08/31/2002    002292    VI SPROIKA
```

Appendix E. Using Debug Tool to Check Code Conversion

Tutorial examples in this book show how Debug Tool distilling files and listing path coverage for a program.

This final tutorial further illustrates the versatility of Debug Tool It shows how Debug Tool can test a small segment of code. This code is used heavily by other programs, and so it makes sense to exercise and test it thoroughly.

Debug Tool is driven by a script, and the script produces an output file. This approach has many benefits:

- You can test a small segment of code within an existing program, without needing to copy the code into an isolated test program.
- Because a script is used, you can repeat the test, and apply it to code before and after conversion.
- Because an output file is produced, you can run SuperC to compare the two output files (before and after conversion).
- In this particular instance, the script does not read any values from a testbed, so there is no need to restore any testbed. Neither does it update any values, so there is no need to restore any testbed after the script is run.

The example used in this tutorial forces three different date values through the TARDTE3 date conversion routine. This is the same example as that used for COBOL Tester (see 5.4, “Path Coverage and Data Distillation - COBOL Tester” on page 41).

The example can be extended by forcing through more date values, or by testing different parts of code. If you do this, you will need to adjust the Debug Tool scripts, and also adjust the SuperC comparison parameters.

This tutorial should be run after all the tutorials in the main part of the redbook have been run.

E.1 Recreating the Distilled Testbed and Converted Testbed

There is no need to recreate either testbed, since values are not read from any files.

E.2 Starting the CICS Job and Opening Files

1. Start a CICS session.

   You must now make sure that both the distilled and converted Employee Master files are open for the exclusive use of CICS.

2. Enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

3. Find the line starting `Fil(EMPMAST)`.
4. Look along this line to the second status word, which comes after `Vsa`.

5. If the second status word is Ope, go to step 8 on page 150.
6. Move the cursor to the second status word, Clo.
7. Type O, then press Enter.

   The status word changes to Ope (for Open).
8. Find the line starting Fil(EMPMST2).
9. Look along this line to the second status word, which comes after Vsa.
10. If the second status word is Ope, go to step 13.
11. Move the cursor to the second status word, Clo.
12. Type O, then press Enter.

    The status word changes to Ope (for Open).
13. Press PF3 to exit the CEMT display.
14. Clear the screen.

The files are now open.

E.3 Running the First Debug Tool Script

This procedure creates a log file for the unconverted code.

1. Enter the transaction DTCN

   The Debug Tool CICS Interactive Facility appears.
2. COBOL Against the Transaction Id (entry line 2) type NB03 and against the Command File (entry line 9) type *hlq*.VA2000TS.DEBUG(COBDB1) for OS/390 or userlib(COBDB1.CMD) for VSE COBOL

   PL/I Against the Transaction Id (entry line 2) type MB03 and against the Command File (entry line 9) type *hlq*.VA2000TS.DEBUG(PL1DB1) for OS/390 or userlib(PL1DB1.CMD) for VSE PL/I
3. Press PF4 (Add).

   If the add fails because the profile exists, press PF5 (Replace).
4. Press PF3 (Exit).
5. COBOL Start program NB03 COBOL

   PL/I Start program MB03 PL/I

   There is a delay, while Debug Tool is initialized, then the Debug Tool window appears.
6. Press F9 (GO) repeatedly, pausing after each press until the X SYSTEM message disappears.
7. When the message Do you really want to terminate this session? appears, type Y and press Enter.

   The transaction name reappears.
8. Clear the screen. The running of the first script is finished.
9. Close DTCN, by entering DTCN and pressing F10.

E.4 Running the Second Debug Tool Script

This procedure creates a log file for the converted code.

1. Enter the transaction `DTCN`

   The Debug Tool CICS Interactive Facility appears.

2. COBOL Against the Transaction Id (entry line 2) type `NB3X` and against the Command File (entry line 9) type *hlq*`.VA2000TS.DEBUG(COBXDB1)` for OS/390 or `userlib(COBXDB1.CMD)` for VSE COBOL

   PL/I Against the Transaction Id (entry line 2) type `MB3X` and against the Command File (entry line 9) type *hlq*`.VA2000TS.DEBUG(PL1XDB1)` for OS/390 or `userlib(PL1XDB1.CMD)` for VSE PL/I

3. Press PF4 (Add).
4. If the add fails because the profile exists, press PF5 (Replace).
5. Press PF3 (Exit).
6. COBOL Start program `NB3X` COBOL

   PL/I Start program `MB3X` PL/I

   There is a delay, while Debug Tool is initialized, then the Debug Tool window appears.

7. Press F9 (GO) repeatedly, pausing after each press until the `X SYSTEM` message disappears.
8. When the message `Do you really want to terminate this session?` appears, type `Y` and press Enter.

   The transaction name reappears.

9. Clear the screen. The running of the second script is finished.
10. Close DTCN, by entering DTCN and pressing F10.

E.5 Ending the CICS Session

The Employee Master files must be closed, so that other jobs can use them.

1. Enter the transaction

   ```
   CEMT I FI (EMP*)
   ```

   The list of files starting with EMP appears.

2. Find the line starting `Fil(EMPMAST)`.
3. Look along this line to the second status word, which comes after `Vsa`.
4. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

   The status word changes to `Clo` (for Closed).

5. Find the line starting `Fil(EMPMST2)`.
6. Look along this line to the second status word, which comes after `Vsa`.
7. Move the cursor to the second status word, which is `Ope`, and type `C`, and then press Enter.

The status word changes to Clo (for Closed).

8. Press PF3 to exit the CEMT display.
9. Clear the screen.
10. Close the CICS session.

The creation of the two log files is now finished.

E.6 Log Comparisons - SuperC

1. Execute job SUPERC5A for COBOL, or SUPERC5B for PL/I.

   This job compares the logs from the two Debug Tool sessions.

2. Check the output of the job.
3. If the job failed, determine the reason, correct, and resubmit.
4. Check the report output. This should show that 3 lines have matched. The reports have produced the same results.

E.7 An Explanation of the Debug Tool Scripts

This section briefly explains the Debug Tool scripts used for COBOL and PL/I. The scripts are not identical, but are sufficiently similar that the explanation of one also explains the other. The script for the unconverted COBOL program is:

```
* IBM Debug Tool Version 1 Release 2 Mod 0
* 11/18/1997 10:11:19 AM
* 5688-194 (C) Copyright IBM Corp. 1992, 1995
  01 CNT PIC 9(4) COMP ;
  01 DISPLAYED-DATE PIC X(30) ;
  MOVE 0 TO CNT ;
  AT APPEARANCE TARDTE3
    PERFORM
      AT ENTRY TARDTE3
        PERFORM
          COMPUTE CNT = CNT + 1 ;
          IF CNT = 1 THEN                         A
            MOVE '11/01/97' TO INPUT-DATE ;
            MOVE 'MM/DD/YY' TO INPUT-FORMAT ;
            MOVE 'YYDDD' TO OUTPUT-FORMAT ;
          END-IF ;
          IF CNT = 2 THEN
            MOVE '96060' TO INPUT-DATE ;
            MOVE 'YYDDD' TO INPUT-FORMAT ;
            MOVE 'YYMMDD' TO OUTPUT-FORMAT ;
          END-IF ;
          IF CNT = 3 THEN
            MOVE '960229' TO INPUT-DATE ;
            MOVE 'YYMMDD' TO INPUT-FORMAT ;
            MOVE 'MM/DD/YY' TO OUTPUT-FORMAT ;    A
          END-IF ;
        END-PERFORM ;
      AT EXIT TARDTE3
        PERFORM
          MOVE SPACES TO DISPLAYED-DATE ;                    B
          MOVE '*' TO DISPLAYED-DATE ( 31 : 1 ) ;
          IF CNT = 1 THEN
            MOVE INPUT-DATE TO DISPLAYED-DATE ( 1 : 10 ) ;
          END-IF ;
          IF CNT = 2 THEN
            MOVE INPUT-DATE TO DISPLAYED-DATE ( 11 : 10 ) ;
          END-IF ;
          IF CNT = 3 THEN
            MOVE INPUT-DATE TO DISPLAYED-DATE ( 21 : 10 ) ;
          END-IF ;
          LIST UNTITLED ( DISPLAYED-DATE ) ;                 B
        END-PERFORM ;
    END-PERFORM ;
  AT LINE 497                                     C
    PERFORM
      IF CNT >= 3 THEN
        QUIT ;
      ELSE
        GOTO 493 ;
      END-IF ;
    END-PERFORM ;                                 C
  AT OCCURRENCE CEE067
    QUIT ;
```

Figure 49. Debug Tool Script for COBOL Code Testing, Unconverted Program

The lines between the two **A** markers set up the input parameters for the TARDTE3 routine. There are three possibilities.

The lines between the two **B** markers format the output from the routine. The "*" marker in position 31 on the output line marks the output log lines as different from all the other lines in the log.

Different outputs are placed in different positions on an output line. This makes it possible for SuperC to look at three different output formats in one job.

The lines between the two **C** markers force program execution to return to the start of the TARDTE3 routine for three passes, then quits the running of the script after three passes.

The script for the converted COBOL program has the same structure, but uses different values as input to the TARDTE3 routine.

The log file produced by this script starts with a listing of the script. Appended to this is the output built by the script, which looks like this:

```
  GO ;
  GO ;
  GO ;
* '97305                    *'
  GO ;
  GO ;
* '         960229          *'
  GO ;
  GO ;
* '                02/29/96 *'
  GO ;
```

Figure 50. Output from Debug Tool Script for COBOL Code Testing

This figure shows clearly how the dates are placed in different positions on the output line, and how the "*" tags the end of each date output line. The script for the converted COBOL program is similar. This portion from the start of the script shows how the dates are expanded:

```
            AT ENTRY TARDTE3X
               PERFORM
                 COMPUTE CNT = CNT + 1;
                 IF CNT = 1 THEN
                    MOVE '11/01/1997' TO INPUT-DATE;
                    MOVE 'MM/DD/YYYY' TO INPUT-FORMAT;
                    MOVE 'YYYYDDD'    TO OUTPUT-FORMAT;
                 END-IF;
                 IF CNT = 2 THEN
                    MOVE '1996060'    TO INPUT-DATE;
                    MOVE 'YYYYDDD'    TO INPUT-FORMAT;
                    MOVE 'YYYYMMDD'   TO OUTPUT-FORMAT;
                 END-IF;
                 IF CNT = 3 THEN
                    MOVE '19960229'   TO INPUT-DATE;
                    MOVE 'YYYYMMDD'   TO INPUT-FORMAT;
                    MOVE 'MM/DD/YYYY' TO OUTPUT-FORMAT;
                 END-IF;
               END-PERFORM;
```

Figure 51. Portion of Debug Tool Script for COBOL Code Testing, Converted Program

E.8 An Explanation of the SuperC Job

The SuperC job that compares the two log files (before conversion and after conversion) has two parts. The first part uses DFSORT to extract just the date records from the log file, using the

```
INCLUDE COND=(40,1,CH,EQ,C'*')
```

command. Note that the examples are from the COBOL version. For the PL/I version, the column positions are five characters to the left, though the principle is exactly the same.

The second part of the job uses SuperC to compare the three dates. The process statements for this part are as follows:

```
OFOCUS COLS 10:39                1
NFOCUS COLS 10:39                2
OY2C 10:14 YYDDD,EMPTY           3
NY2C 10:16 YYYYDDD,EMPTY
OY2C 20:25 YYMMDD,EMPTY
NY2C 20:27 YYYYMMDD,EMPTY
OY2C 30:37 MM/DD/YY,EMPTY
NY2C 30:39 MM/DD/YYYY,EMPTY      8
Y2PAST 65
```

Figure 52. SuperC Process Statements, Debug Script Logs

Lines **1** and **2** discard any information outside columns 10 to 39. Lines **3** to **8** provide descriptions of the dates, in pairs, for the date before conversion and the date after conversion. The EMPTY parameter warns SuperC that there might be no date. Since matching pairs are empty, the comparison succeeds.

E.9 Extending This Example

The technique used in this tutorial can be readily extended to cater for more dates and more date formats.

There is a limitation of 80 characters to a line of output in the log file. However, this is probably irrelevant, since the three different date formats probably account for most dates.

Instead of changing the position of each date, you could add a position flag. This means that you could test with a large number of different dates, placing them all in the same position.

The main thing is to make sure that whatever dates and formats are in the Debug Tool script for testing the routine before conversion, the equivalent converted dates and formats must be in the Debug Tool script for testing the routine after conversion. Provided you keep the one-to-one correspondence, SuperC can compare the two log files.

Appendix F. Establishing a Test Environment

It is imperative that Year 2000 testing is quarantined from normal production runs. Otherwise the different system dates that are used as part of Year 2000 testing are going to contaminate the data held in the production system, with undesirable consequences.

This appendix discusses the two most common methods of establishing a separate test environment, and points to more information about these methods.

The IBM RS/6000 and System/390 Server on Board (R/390), and the IBM PC Server System/390 (P/390), provide a totally separate environment for you to use for testing. In contrast, Logical Partitions (LPARs) make it possible for you to use existing hardware, and establish an isolated test environment that co-exists on the hardware with other environments.

For more information about setting up a test system, see `http://www.software.ibm.com/year2000/y2ktestsystem.html`.

F.1 Using a P/390 or R/390

The P/390 and R/390 provide a low cost stand-alone environment suitable for application development and Year 2000 testing.

These devices are essentially a large server (IBM PC or RS/6000) unit and a P/390 or R/390 card, which supports a System/390 processor. The System/390 processor is fully functional. It runs standard S/390 operating systems and applications, without the need for modification or special versions. Software drivers handle the emulation of S/370 and S/390 devices, in particular, DASD which is stored within the RAID array on the server.

The S/390 operating systems (OS/390, VM and VSE) are available as preconfigured systems. They are distributed in CD-ROM format, ready for easy installation.

These devices and associated operating systems (including a S/390 operating system or systems) are available only as a complete package. They are provided exclusively by certified Business Partner - Distributors.

The P/390 and R/390 are supplied with OS/2 Warp or AIX. Supplied driver software provides the emulation of S/390 I/O devices such as tape devices (3420, 3480) and DASD (FBA, 3380, 3390). The PC processor or RS/6000 processor performs I/O.

There are many connectivity options. Lan Adapters (Ethernet or Token Ring) can be made to look like a 3172 control unit, enabling the connection of display terminals, printers and other SNA or TCP/IP devices to the S/390 operating system. The S/390 Parallel Channel Adapter connects to most S/370 and S/390 devices, but not to host DASD devices.

F.1.1 Setting up a Test Environment

The DASD system of the P/390 and R/390 is flexible. As a consequence, there are many ways to copy a host system and move it to the smaller unit, preparatory to Year 2000 testing:

- Copy volumes as DFSMS/DSS dumps, using NETVIEW FTP.
- Restore the VM/ESA system from tape backups. The backups can be from either VM/ESA DDR or VSE/ESA FASTCOPY. In either case, you use a channel-attached unit or a SCSI-attached unit which can read magnetic media such as 3480/3490 cartridges. The P/390 or R/390 can be IPLed in a standalone manner to load the DDR or FASTCOPY program.
- Use VM/ESA with OS/390 or VSE/ESA as guests.

By transferring the operating system, you can set up an isolated test machine which uses exactly the same software as your host, and where developers access the P/390 or R/390 as if it were the host. This makes the unit ideal for function testing.

Once you have set up the operating system, you can transfer application programs and test data, to complete the setting up of the test environment.

The performance of the unit is limited—the bottle-neck is I/O. This means that the unit may **NOT** be appropriate for production processing. Neither is it appropriate for stress or performance testing.

To use a P/390 or R/390 for Year 2000 testing, it must be completely isolated from any other computer. In particular, it must **NOT** share DASD, nor must it share tape drives or a tape library.

Note that IBM software run on a P/390 or R/390 is available at a special price. Many third party vendor software suppliers also support special prices for software run on these platforms.

F.1.2 The Advantages of Running Year 2000 Testing on a P/390 or R/390

Running a P/390 or R/390 as a test environment for Year 2000 testing has many advantages:

- If the P/390 or R/390 is isolated from all other computers, it provides the ideal platform for the conversion of programs. Data is isolated from production data or other test data.

  You can check the running of programs under future date conditions thoroughly. Changing the system date for a P/390 or R/390 is a matter of IPLing, and changing the date to whatever you want. There is no external reference for the system.

  This means that your developers and systems programmers can change the system date frequently, checking for the change of year from 1999 to 2000, the change of date from 28 February 2000 to 29 February 2000, and then on to 1 March 2000, and so on, for other critical dates.
- Alternatively, the P/390 or R/390 may be left “ticking over,” with the date advancing at the normal rate as time advances. By advancing at a more sedate pace, testing may find errors in routines unlikely to be exercised in spot checking.

- A testing regime can be quite exhaustive and upon completion, the P/390 or R/390 can be restored from backup to its original state. This facilitates repeat testing.
- The P/390 or R/390 system has a small footprint and requires no special environmental considerations. It can be located with any Testing Group or Development Team, giving them control of the system, in particular, IPLing to change the system date.
- You can use the P/390 or R/390 to check for date problems, without looking at source code. Set production programs running over a local copy of production data, at a date in the next century, and see what "breaks." This is not an exhaustive approach to finding Year 2000 problems, but it quickly provides a starting-point for your investigations.
- Once you have finished Year 2000 testing, you can continue to use the P/390 or R/390 for installing and testing software.

F.1.3 For More Information

For more information about P/390 or R/390 systems and Year 2000 testing, see `http://www.s390.ibm.com/stories/year2000/`. And for information about P/390 or R/390 systems, see `http://www.s390.ibm.com/products/p390/p390hp.html`.

F.2 Using Logical Partitions

Logical Partitions (LPARs) provide a way of physically dividing a processor complex into a number of individual environments, each of which can support its own operating system, such as OS/390, MVS/ESA, or VSE/ESA.

LPAR support on the IBM mainframes is provided by the hardware feature known as Processor Resource/Systems Manager (PR/SM). This handles the allocation of central processors, central storage, expanded storage, and channel paths amongst the logical partitions.

The number of Logical Partitions supported on a processor complex depends on processor model and configuration. The P/390 and R/390 processors do not support LPAR mode.

An LPAR can share central processors and channels paths (ESCON channels) only on processors with the ESCON Multiple Image Facility (EMIF) feature installed. Central and expanded storage must be dedicated to an individual LPAR but, depending of the system configuration, can be reconfigured to another LPAR when the storage is no longer in use.

PR/SM Planning Guide provides more information about logical partitions (*S/390 PR/SM Planning Guide*, GA22-7236, and *ES/9000 and ES/3090 PR/SM Planning Guide*, GA22-7123).

F.2.1 Isolating an LPAR for Year 2000 Testing

In a normal production environment a number of LPARs or physically separate machines can share external devices, including tape drives, disk storage, and networks. In an LPAR where the system date for that LPAR is to be changed to undertake Year 2000 Testing, it is important to achieve sufficient isolation of resources to the extent that it will not interfere with another system or LPAR.

Undertaking Year 2000 testing—especially changing the system date—in a multiple operating systems environment running either a Sysplex or Parallel Sysplex is beyond the scope of this redbook, but is explored in the Time Management redbook, *S/390 Time Management and IBM 9037 Sysplex Timer* (SG24-2070), also at `http://www.redbooks.ibm.com/SG242070/y2000.html`.

F.2.1.1 Central Processors

The Central Processors for an LPAR can be defined as either shared (amongst other LPARs) or dedicated. The sharing of central processors does not cause problems running a Year 2000 Test system.

F.2.1.2 Central and Expanded Storage

It is not possible to share central or expanded storage concurrently between LPARs.

Sufficient storage must be available for each LPAR to meet the workload requirements of that LPAR. The amount of central storage available to a Year 2000 Test systems determines the performance of any Year 2000 testing.

F.2.1.3 Channel Paths

Parallel channels cannot be shared concurrently between LPARs. Providing they are defined correctly, they can be dynamically reconfigured between LPARs. ESCON channels can be shared where the ESCON Multiple Image Facility (EMIF) feature is installed.

There are no issues with sharing or dynamically reconfiguring channel paths between a Year 2000 Test system and other systems. Consideration needs to be given to the devices connected to the channels as discussed in the following sections.

F.2.1.4 Disk Storage

There is a high risk of either data corruption or data loss by sharing disk storage between a Year 2000 Test system and other systems. To protect your business investment and your production data, the sharing of disk storage should be avoided.

A consequence of a Year 2000 Test system not being able to share disk storage is that the system cannot participate in the same Sysplex or Parallel Sysplex as non-Year 2000 systems. Also, it cannot share tape management catalogs, security databases, or similar resource management systems with non-Year 2000 systems.

Plan carefully to determine the amount of extra disk storage needed to set up a Year 2000 Test environment. Disk storage is required for:

- System Residence (or IPL) volume(s)
- System and Product datasets
- JES2 or JES3 spool
- Paging datasets
- Application data

Start with the storage needed to duplicate the current system packs, and then add storage needed for data packs. The test data may require much less storage than production data.

F.2.1.5 Disk Controllers

The outcome of sharing disk controllers is not clear. As with disk storage, it is important to protect your business investment. Given the uncertainty of sharing, the sharing of disk controllers is best avoided.

F.2.1.6 Tape Drive

Tape libraries require shared disk storage and therefore cannot be shared between a Year 2000 Test system and other systems.

Tapes drives generally interface with one form or other of a tape management catalog. This prevents the sharing of tape drives between a Year 2000 Test system and other systems. It also means that you should establish a separate tape pool for scratch tapes for Year 2000 testing.

F.2.1.7 Networks

Sharing networks between a Year 2000 Test system and other systems needs careful consideration.

A Year 2000 Test system generating TCP/IP traffic must not share a network with other systems. However, a Year 2000 Test system generating only SNA data streams can share.

Network Job Entry (NJE) data between a Year 2000 Test system and other systems is also supported and may provide a useful way of copying small amounts of data between the different systems. NJE connections are relatively slow and using NJE impacts the amount of JES2 or JES3 Spool space required.

F.2.1.8 Other Devices

A Year 2000 Test system requires a separate non-SNA controller for the Master console. Depending on your network requirements, this controller can also be used to provide additional 3270 terminals and isolated access to a Year 2000 Test system.

A Sysplex Timer (9037) must not be shared between a Year 2000 Test system and other non-Year 2000 systems. It is the Sysplex timer that provides the date and time when an operating system is loaded (IPLed).

Channel attached printers cannot be shared, but can be dynamically reconfigured between different LPARs. This present no problems, except that the people who distribute printouts must know when the printer is printing output from the Year 2000 Test system.

F.2.2 Setting the Date on an LPAR for OS/390

Setting the date and time for a Year 2000 Test environment depends on the hardware configuration of the environment.

For a single LPAR in a Year 2000 Test system, the date and time of the Support Element should not be changed as this affects other systems.

The preferred method is to specify the date and time at IPL time by using a CLOCKxx member in SYS1.PARMLIB with:

```
 OPERATOR PROMPT
 ETRMODE NO
 ETRZONE NO
 TIMEZONE W.00.00.00
```

The TIMEZONE parameter specifies the difference between GMT and local time. This parameter should be set as required.

During IPL, the operator is prompted to set the clock when the following messages are issued:

```
 IEA598I TIME ZONE = W.05.00.00
 IEA888A GMT   DATE=1997.116,CLOCK=15.29.24
*   IEA888A LOCAL DATE=1997.116,CLOCK=10.29.24  REPLY U,
OR GMT/LOCAL TIME

 R 00,DATE=1999.365,CLOCK=23.45.00,GMT
 IEE600I REPLY TO 00 IS;CLOCK=23.45.00,DATE=1999.365,GMT
*   IEA903A REPLY U WHEN THE ENTERED TIME OCCURS

 R 00,U
 IEE600I REPLY TO 00 IS;U
```

The operator can set the year 2000 test date with the response to message IEA888A, as shown above.

There is a lot more information about this topic in the Time Management redbook, *S/390 Time Management and IBM 9037 Sysplex Timer* (SG24-2070), also available at http://www.redbooks.ibm.com/SG242070/y2000.html.

F.2.3 Setting the Date on an LPAR for VSE/ESA

If you are using VSM for the directory entry for the VSE guest machine, you need to specify the TODENABLE option on the options card. Then reload the directory.

Once you have completed the date and time settings, use the systems procedure to enable the TOD clock for the LPAR being IPLed:

1. Modify the IPL procedure member to include the following statement after the DLF command but before the SVA command is issued:

   ```
   SET DATE=,CLOCK=,ZONE=NORMAL ZONE SETTING
   ```

2. IPL the VSE system.

   During IPL the following message appears:

   ```
   0I87D  INVALID SPECIFICATION FOR KEYWORK DATE
   ```

3. Reply to this message with the desired settings in the following format:

   ```
   SET DATE=MM/DD/YYYY,CLOCK=HH/MM/SS,ZONE=normal zone setting
   ```

   The IPL now continues as normal.

Appendix G. Program Information for Tools Used by VisualAge 2000 Test Solution

G.1.1 Debug Tool

Debug Tool is available from IBM as a full function offering of the compilers:

- IBM COBOL for OS/390 & VM (5648-A25)
- IBM COBOL for MVS & VM (5688-197)
- IBM C/C++ for MVS & VM (5655-121)
- IBM OS/390 C/C++ (before Release 4, 5645-001; after Release 4, 5647-A01)
- IBM PL/I for MVS & VM (5688-235)
- IBM COBOL for VSE/ESA (5686-068)
- IBM C for VSE/ESA (5686-A01)
- IBM PL/I for VSE/ESA (5686-069)

Debug Tool is also available from IBM as part of CODE/370 (5688-194).

G.1.2 IBM Application Testing Collection

IBM Application Testing Collection is available from IBM as IBM Application Testing Collection for MVS (5799-GBN).

This collection contains Coverage Assistant, Distillation Assistant, and Source Audit Assistant.

G.1.3 IBM Data Facility Sort

IBM Data Facility Sort with Year 2000 features is available from IBM:

- For OS/390, as IBM Data Facility Sort (DFSORT) Version 1 Release 13 (5740-SM1) with PTF UN90139 and PTF UQ05520
- For VSE, as IBM DFSORT/VSE Version 3 Release 3 (5746-SM3), or IBM DFSORT/VSE Version 3 Release 2 (5746-SM3) with PTF UN99635

G.1.4 Enhanced SuperC—part of IBM High Level Assembler for MVS & VM & VSE

IBM High Level Assembler for MVS & VM & VSE Toolkit Feature is available from IBM as an optional feature of IBM High Level Assembler for MVS & VM & VSE (5696-234).

This package includes Enhanced SuperC.

G.1.5 IBM VisualAge Test for OS/2

IBM VisualAge Test for OS/2, Version 1, is available from IBM as IBM VisualAge for COBOL, Test for OS/2, Version 1 Release 2 (33H005).

This package includes COBOL Tester and Auto Test Performer.

G.1.6 WITT Year2000 for OS/2

IBM WITT Year2000 for OS/2, Version 3, is available from IBM as IBM WITT Year2000 for OS/2, Version 3.0 (33H0036).

This package includes Auto Test Performer.

G.1.7 WITT Year2000 for Windows

WITT Year2000 for Windows Version 1, which runs under Windows 95 and Windows NT, is available from IBM as IBM WITT Year2000 for Windows, Version 1.0 (33H0050).

Appendix H. Employee Master File Descriptions

This appendix lists the record descriptions for the legacy and converted Employee Master File.

The examples are for COBOL. PL/I files have the same structure.

H.1 The Legacy Employee Master File Record

```
 01  EMPLOYEE-MASTER-RECORD.
*        ** key field
     03  EMP-ID                      PIC X(6).
     03  EMP-DEPT-CODE               PIC X(4).
     03  EMP-NAME                    PIC X(30).
     03  EMP-ADDR-1                  PIC X(30).
     03  EMP-ADDR-2                  PIC X(30).
     03  EMP-ADDR-3                  PIC X(30).
     03  EMP-ZIP-CODE                PIC X(5).
*        ** format (yyddd)
     03  EMP-DATE-JOINED             PIC 9(5).
*        ** format (yymmdd)
     03  EMP-DATE-TERMINATED         PIC 9(6).
*        ** format (yyddd)
     03  EMP-DATE-MAINTAINED         PIC 9(5).
*        ** format (yyddd)
     03  EMP-BIRTH-DATE              PIC 9(5).
*        ** format (yyddd)
     03  EMP-SECURITY-EXP            PIC 9(5) COMP-3.
     03  FILLER                      PIC X(41).
```

H.2 The Converted Employee Master File Record

```
 01  EMPLOYEE-MASTER-RECORD.
*        ** key field
     03  EMP-ID                      PIC X(6).
     03  EMP-DEPT-CODE               PIC X(4).
     03  EMP-NAME                    PIC X(30).
     03  EMP-ADDR-1                  PIC X(30).
     03  EMP-ADDR-2                  PIC X(30).
     03  EMP-ADDR-3                  PIC X(30).
     03  EMP-ZIP-CODE                PIC X(5).
*        ** format (yyyyddd)  date expanded
     03  EMP-DATE-JOINED             PIC 9(7).
*        ** format (yyyymmdd) date expanded
     03  EMP-DATE-TERMINATED         PIC 9(8).
*        ** format (yyyyddd) packed date compressed
     03  EMP-DATE-MAINTAINED         PIC 9(7) COMP-3.
*        ** format (yyyyddd) packed date compressed
     03  EMP-BIRTH-DATE              PIC 9(7) COMP-3.
*        ** format (yyddd) packed date uses sliding window
     03  EMP-SECURITY-EXP            PIC 9(5) COMP-3.
     03  FILLER                      PIC X(39).
```

Appendix I. Files Used In the Tutorials

This appendix briefly describes the files used in the tutorials, as they appear on the host or the work station.

I.1 OS/390 Files

There are 9 partitioned data sets (PDS) used in the tutorials.

I.1.1 *hlq*.VA2000TS.ATC

This PDS contains source programs and batch jobs used by the ATC processes:

Table 1. ATC Procedures for OS/390

Member	**Use**
CPL1MU6X	Compiles PL1MU6X
CTARMU6X	Compiles TARMU6X
DISTCOPY	Copies VSAM file to be distilled to a sequential file
DTARMU6X	Distillation listing
GPL1MU6X	Runs PL1MU6X
GTARMU6X	Runs TARMU6X
LPL1MU6X	Links PL1MU6X
LTARMU6X	Links TARMU6X
PL1MU6X	PL1MU6X source
RPL1MU6X	Creates CA reports
SPL1MU6X	Setup
STARMU6X	Setup
TARMU6	TARMU6 source
TARMU6X	TARMU6X source
XPL1MU6X	Starts monitor
XTARMU6X	Starts monitor

I.1.2 *hlq*.VA2000TS.CLIST

This PDS contains REXX procedures and CLISTs used in the tutorials:

Table 2 (Page 1 of 2). CLISTs and REXX Procedures for OS/390

Member	**Use**
TARBON2	Creates *hlq*.EMP.MASTON2.BACK and copies *hlq*.EMP.MASTER.ONLINE2 into it (creates a backup copy)
TARBON3	Creates *hlq*.EMP.MASTON3.BACK and copies *hlq*.EMP.MASTER.ONLINE3 into it (creates a backup copy)
TARCNTL	REXX procedure that customizes batch job control
TARDBG	REXX procedure that customizes Debug Tool command files.
TARDHLQ	Prompts you for *hlq* and saves it for later use—called by other REXX procedures

Table 2 (Page 2 of 2). CLISTs and REXX Procedures for OS/390

Member	Use
TARDNVSM	Allocates report data sets
TARDVSAM	Creates initial VSAM data sets (*hlq*.EMP.MASTER.START and *hlq*.DEPT.MASTER.ONLINE) with records needed by the tutorials
TARDVSM1	Creates *hlq*.EMP.MASTER.ONLINE from *hlq*EMP.MASTER.START
TARDVSM2	Creates *hlq*.EMP.MASTER.ONLINE2
TARDVSM3	Creates *hlq*.EMP.MASTER.ONLINE3 from *hlq*.EMP.MASTON2.BACK
TARDVSM4	Creates *hlq*.EMP.MASTER.ONLINE4 from *hlq*.EMP.MASTON3.BACK
TARREC	REXX procedure that receives PDS files sent from PC.
TARRON2	Restores *hlq*.EMP.MASTER.ONLINE2 from *hlq*.EMP.MASTON2.BACK
VSAMBROW	Not used by tutorials—allows you to browse a VSAM data set

I.1.3 *hlq*.VA2000TS.CNTL

This PDS contains JCL for jobs used in the tutorials:

Table 3. Job Control Language for OS/390

Member	Use
CICSDEF	Sample to create CICS CSD entries
SUPERC1A	Compares baseline and post-fix Employee Master files
SUPERC1B	Compares baseline and post-fix reports
SUPERC2A	Compares post-fix and 19xx Employee Master files
SUPERC2B	Compares post-fix and 19xx reports
SUPERC3A	Compares 19xx and 20xx Employee Master files
SUPERC3B	Compares 19xx and 20xx reports
SUPERC5A	Compares Debug Tool COBOL log files
SUPERC5B	Compares Debug Tool PL/I log files
TARAGE3T	Ages *hlq*.EMP.MASTER.ONLINE3 by 3 years
TARCNV2	Builds *hlq*.EMP.MASTER.ONLINE2 from *hlq*.EMP.MASTER.START by expanding date fields
TARRPA1	Creates baseline security card expiry dates report
TARRPA2	Creates post-fix security card expiry dates report
TARRPA3	Creates 19xx security card expiry dates report
TARRPA4	Creates 20xx security card expiry dates report

I.1.4 *hlq*.VA2000TS.COBOL

This PDS contains COBOL source for programs used in the tutorials:

Table 4 (Page 1 of 2). COBOL Source for OS/390

Member	Use
MLEMU5	MLE version of TARMU5 report program
TARAGE3	Program to age Employee Master file

Table 4 (Page 2 of 2). COBOL Source for OS/390

Member	Use
TARDTE3	Date utility program
TARDTE3X	Y2K date utility program
TARMU3	Program for CICS transaction NB03 (before conversion)
TARMU3E	Program for CICS transaction NB3E (converted, but with error)
TARMU3X	Program for CICS transaction NB3X (converted, with error corrected)
TARMU5	Report program

I.1.5 *hlq*.VA2000TS.DATA

This PDS contains SYSIN data for jobs and procedures used in the tutorials. Examine the *hlq*.VA2000TS.CNTL and *hlq*.VA2000TS.CLIST data sets to see where members are used.

I.1.6 *hlq*.VA2000TS.DEBUG

This PDS contains Debug Tool command files:

Table 5. Debug Tool Command Files for OS/390

File name	Contents
COBDB0 CMD	Checking coverage
COBDB1 CMD	Checking code conversion
COBDB2 CMD	Building a distilled key list
COBXDB1 CMD	Checking code conversion
PL1DB0 CMD	Checking coverage
PL1DB1 CMD	Checking code conversion
PL1XDB1 CMD	Checking code conversion

I.1.7 *hlq*.VA2000TS.LISTING

This PDS contains program listing files used by Debug Tool and ATC reports:

Table 6 (Page 1 of 2). Listings for OS/390

Member	Use
DTARMU6X	ATC Distillation Assistant report
PL1DTE3	Compiler listing
PL1DT3X	Compiler listing
PL1MU3	Compiler listing
PL1MU3X	Compiler listing
RPL1MU6X	ATC Coverage Assistant report
RTARMU6X	ATC Coverage Assistant report
SAAREPT	ATC Source Audit Assistant compare of TARMU6 and TARMU6X source programs
TARDTE3	Compiler listing
TARDTE3X	Compiler listing

Table 6 (Page 2 of 2). Listings for OS/390

Member	Use
TARMU3	Compiler listing
TARMU3X	Compiler listing

I.1.8 *hlq*.VA2000TS.LOAD

This PDS contains load modules used in the tutorials:

Table 7. Load Modules for OS/390

Member	Use
PL1AGE3	PL/I program to age Employee Master File
PL1DTE3	PL/I date utility
PL1DT3X	PL/I Y2K date utility
PL1MU3	PL/I CICS program used by transaction MB03
PL1MU3E	PL/I CICS program used by transaction MB3E
PL1MU3X	PL/I CICS program used by transaction MB3X
PL1MU5	PL/I report program
PL1MU6X	ATC PL/I sample program.
TARAGE3	Program to age Employee Master file
TARDTE3	Date utility program
TARDTE3X	COBOL Y2K date utility
TARMU3	Program for CICS transaction NB03 (before conversion)
TARMU3E	Program for CICS transaction NB3E (converted, but with error)
TARMU3M	Mapset for CICS transaction NB03
TARMU3X	Program for CICS transaction NB3X (converted, with error corrected)
TARMU5	Report program
TARMU6X	ATC COBOL sample program
TARM3XM	Mapset for CICS transaction NB3X

I.1.9 *hlq*.VA2000TS.PLI

This PDS contains PL/I source for programs used in the tutorials:

Table 8 (Page 1 of 2). PL/I Source for OS/390

Member	Use
MLEMU5	MLE version of PL1MU5 program.
PL1AGE3	Program to age Employee Master file
PL1DTE3	Date utility
PL1DT3X	Year 2000 date utility
PL1MU3	CICS program used by transaction MB03
PL1MU3E	CICS program used by transaction MB3E
PL1MU3X	CICS program used by transaction MB3X

Table 8 (Page 2 of 2). PL/I Source for OS/390

Member	Use
PL1MU5	Report program

I.1.10 VSAM Data Sets

There is only one Department Master file (*hlq*.DEPT.MASTER.ONLINE) as it not updated by the tutorials. The Employee Master file proceeds through 7 versions:

Table 9. VSAM Data Sets for OS/390

Data set name	Contents
hlq.EMP.MASTER.ONLINE	Copy of START, used to establish baseline
hlq.EMP.MASTER.ONLINE2	ONLINE with date fields expanded (post-fix)
hlq.EMP.MASTER.ONLINE3	ONLINE2 aged 3 years (19xx)
hlq.EMP.MASTER.ONLINE4	ONLINE3 updated with system date year set to 2000 (20xx)
hlq.EMP.MASTER.START	Initial Employee Master
hlq.EMP.MASTON2.BACK	Backup of ONLINE2, used to create ONLINE3
hlq.EMP.MASTON3.BACK	Backup of ONLINE3, used to create ONLINE4

I.1.11 Other OS/390 Data Sets

There are two other files used in the OS/390 side of the tutorials:

Table 10. Other OS/390 Data Sets

Data set name	Contents
hlq.EMP.DATA	Unloaded copy of *hlq*.EMP.MASTER.START
hlq.DEPT.DATA	Unloaded copy of *hlq*.DEPT.MASTER.ONLINE

I.2 VSE Files

The following files are punched into the nominated sublibrary.

I.2.1 *.A

This file is the CICS File Control Table:

Table 11. CICS File Control Table for VSE

File name	Contents
TARFCT A	The CICS File Control Table

I.2.2 *.C

These files are the COBOL source files for programs used in the tutorials:

Table 12. COBOL Source for VSE

File name	Contents
MLEMU5 C	MLE version of TARMU5 report program
TARAGE3 C	Program to age Employee Master file
TARDTE3 C	Date utility program
TARDTE3X C	Y2K date utility program
TARMU3 C	Program for CICS transaction NB03 (before conversion)
TARMU3E C	Program for CICS transaction NB3E (converted, but with error)
TARMU3M C	Mapset for CICS transaction NB03
TARMU3X C	Program for CICS transaction NB3X (converted, with error corrected)
TARMU5 C	Report program
TARM3XM C	Mapset for CICS transaction NB3X

I.2.3 *.CMD

These files are Debug Tool command files:

Table 13. Debug Tool Command Files for VSE

File name	Contents
COBDB0 CMD	Checking coverage
COBDB1 CMD	Checking code conversion
COBDB2 CMD	Building a distilled key list
COBXDB1 CMD	Checking code conversion
PL1DB0 CMD	Checking coverage
PL1DB1 CMD	Checking code conversion
PL1XDB1 CMD	Checking code conversion

I.2.4 *.LIST

These files are program listing files used by Debug Tool:

Table 14. Listings for VSE

File name	Contents
PL1DTE3 LIST	Compiler listing
PL1DT3X LIST	Compiler listing
PL1MU3 LIST	Compiler listing
PL1MU3X LIST	Compiler listing
TARDTE3 LIST	Compiler listing
TARMU3 LIST	Compiler listing
TARMU3X LIST	Compiler listing

I.2.5 *.LOG

These are log files created by Debug Tool command files:

Table 15. Debug Tool Log Files for VSE

File name	Contents
PL1MU3 LOG	Distillation example
PL1MU3X LOG	Distillation example
TARMU3 LOG	Distillation example
TARMU3X LOG	Distillation example
TARMU3 LOG2	Distillation example

I.2.6 *.P

These files are PL/I source code files:

Table 16. PL/I Source Code for VSE

File name	Contents
MLEMU5 P	MLE version of TARMU5 report program
PL1AGE3 P	Program to age Employee Master file
PL1DTE3 P	Date utility program
PL1DT3X P	Y2K date utility program
PL1MU3 P	Program for CICS transaction MB03 (before conversion)
PL1MU3E P	Program for CICS transaction MB3E (converted, but with error)
PL1MU3X P	Program for CICS transaction MB3X (converted, with error corrected)
PL1MU5 P	Report program

I.2.7 *.PHASE

These files are program phases:

Table 17 (Page 1 of 2). Program Phases for VSE

File name	Contents
PL1AGE3 PHASE	PL/I program to age Employee Master File
PL1DTE3 PHASE	PL/I date utility
PL1DT3X PHASE	PL/I Y2K date utility
PL1MU3 PHASE	PL/I CICS program used by transaction MB03
PL1MU3E PHASE	PL/I CICS program used by transaction MB3E
PL1MU3X PHASE	PL/I CICS program used by transaction MB3X
PL1MU5 PHASE	PL/I report program
TARAGE3 PHASE	Program to age Employee Master file
TARCONV PHASE	Program to convert Employee Master file
TARDTE3 PHASE	Date utility program
TARLOAD PHASE	Program to build Employee Master file
TARMU3 PHASE	Program for CICS transaction NB03 (before conversion)

Table 17 (Page 2 of 2). Program Phases for VSE

File name	Contents
TARMU3E PHASE	Program for CICS transaction NB3E (converted, but with error)
TARMU3M PHASE	Mapset for CICS transaction NB03
TARMU3X PHASE	Program for CICS transaction NB3X (converted, with error corrected)
TARMU5 PHASE	Report program
TARM3XM PHASE	Mapset for CICS transaction NB3X

I.2.8 *.PROC

These files are REXX procedures used during installation:

Table 18. REXX Procedures for VSE

File name	Contents
TARCHG PROC	Updates batch jobs at time of installation
TARCHG2 PROC	Updates batch jobs at time of installation

I.2.9 *.Z

These are the batch jobs submitted as part of tutorials:

Table 19 (Page 1 of 2). Batch Jobs for VSE

File name	Contents
RTARCHG Z	Customizes batch jobs at time of installation
SUPERC1A Z	Compares baseline and post-fix Employee Master files
SUPERC1B Z	Compares baseline and post-fix reports
SUPERC2A Z	Compares post-fix and 19xx Employee Master files
SUPERC2B Z	Compares post-fix and 19xx reports
SUPERC3A Z	Compares 19xx and 20xx Employee Master files
SUPERC3B Z	Compares 19xx and 20xx reports
SUPERC5A Z	Compares Debug Tool COBOL log files
SUPERC5B Z	Compares Debug Tool PL/I log files
TARAGE3T Z	Ages *hlq*.EMP.MASTER.ONLINE3 by 3 years
TARBON2 Z	Creates *hlq*.EMP.MASTON2.BACK and copies *hlq*.EMP.MASTER.ONLINE2 into it (creates a backup copy)
TARBON3 Z	Creates *hlq*.EMP.MASTON3.BACK and copies *hlq*.EMP.MASTER.ONLINE3 into it (creates a backup copy)
TARCNV2 Z	Builds *hlq*.EMP.MASTER.ONLINE2 from *hlq*.EMP.MASTER.START by expanding date fields
TARCSD Z	Set-up job, defines transactions and programs for CICS processing
TARDVSAM Z	Creates initial VSAM data sets (*hlq*.EMP.MASTER.START and *hlq*.DEPT.MASTER.ONLINE) with records needed by the tutorials
TARDVSM1 Z	Creates *hlq*.EMP.MASTER.ONLINE from *hlq*EMP.MASTER.START
TARDVSM2 Z	Creates *hlq*.EMP.MASTER.ONLINE2

Table 19 (Page 2 of 2). Batch Jobs for VSE

File name	Contents
TARDVSM3 Z	Creates *hlq*.EMP.MASTER.ONLINE3 from *hlq*.EMP.MASTON2.BACK
TARDVSM4 Z	Creates *hlq*.EMP.MASTER.ONLINE4 from *hlq*.EMP.MASTON3.BACK
TARRON2 Z	Restores *hlq*.EMP.MASTER.ONLINE2 from *hlq*.EMP.MASTON2.BACK
TARRPA1 Z	Creates baseline security card expiry dates report
TARRPA2 Z	Creates post-fix security card expiry dates report
TARRPA3 Z	Creates 19xx security card expiry dates report
TARRPA4 Z	Creates 20xx security card expiry dates report

I.3 OS/2

There are two OS/2 files:

Table 20. OS/2 Data Sets

Data set name	Contents
TARADJST.CMD	REXX procedure to convert and age ATP scripts
TARDTE3.COB	Program used in COBOL Tester example

I.4 Windows

There is one Windows files:

Table 21. Windows File

Data set name	Contents
TARADJST.TSC	Lotus Script procedure to convert and age WITT Year2000 for Windows scripts

Appendix J. Special Notices

This publication is intended to help people involved in Year 2000 testing to understand the Year 2000 Test Solution. The information in this publication is not intended as a specification of any programming interfaces that are provided by the tools discussed in this publication. See the PUBLICATIONS section of the IBM Programming Announcement for the tools listed in Chapter 2, "The Tools Used by VisualAge 2000 Test Solution" for more information about what publications are considered to be product documentation.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals. All of these names are fictitious, and any similarity to the names and addresses of actual people is entirely coincidental.

Any pointers in this publication to websites are provided for convenience only and do not in any manner serve as an endorsement of these websites.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

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

Appendix K. Related Publications

The publications listed in this section provide more information about the Year 2000 challenge.

K.1 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	**Subscription Number**	**Collection Kit Number**
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

K.2 Web Publications

These World Wide Web publications are relevant as further information sources:

- *VisualAge 2000: Methodology and Tools Implementation*, `http://www.software.ibm.com/ad/va2000/y2k/`
- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, `http://www.software.ibm.com/year2000/resource.html`
- *Year 2000 scripts for Debug Tool*, `http://www.software.ibm.com/year2000/tools19.html`
- *DFSORT's Year 2000 and performance enhancements*, `http://www.storage.ibm.com/software/sort/srtmy2p.htm`

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at ″Redpieces″ on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.htm`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at ″Redpieces″ on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

- Invoice to customer number

- Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

acceptance tests. Formal tests of the system, carried out by the users when all changes to the system have been made. If the results of the acceptance tests are not what the users expected, further changes are required. See also *performance test*.

aged testbed. The *converted testbed* after it has been aged. The aged testbed holds information in the formats required for Year 2000 readiness, and includes dates in 20xx. The aged testbed is used to produce Year 19xx results and Year 20xx results.

aging. The process whereby dates in testbeds are advanced. For testing purposes, dates should be advanced beyond 1999. For example, if a date includes the year 1987, this can be advanced to 2015. Testbeds can also be aged by adding master file records and transactions that include dates beyond 1999, but this method introduces problems when results are compared. All dates in the testbed must be advanced by the same interval.

archive. Applies to Auto Test Performer and WITT Year2000 for Windows screen images. When the images are archived, the Benchmark images are discarded, and the Current images are made the Benchmark images, and hence become the basis for any future image comparisons.

B

baseline. The output that results from processing transactions against the master files. A baseline is then stored, and later in the test cycle, is compared against other results. The differences between the baseline and other results should arise from the Year 2000 conversion. The main purpose of testing is to find differences that cannot be explained.

black box testing. Testing that is driven by the functional specifications or external design specifications of the system, without regard to the internal composition of programs. See also *white box testing*.

C

control box. The control box is the small icon at the top left corner of an OS/2 window. Click on it once to show the control menu, or double-click it to close the window.

converted code. The code after it has been modified to deal with any Year 2000 problems.

converted testbed. The *distilled testbed* after it has been converted, so that any dates are in the format required for Year 2000 readiness. If a window technique is used for a date, then it does not require converting. Otherwise, when expansion or compression techniques are used, the date must be converted. The converted testbed is used to produce the Post-fix results, and is aged to produce the *aged testbed*.

coverage. The lines of code that have been executed when a set of data is run against the code. Coverage is increased by providing more data items. The ideal is 100% coverage.

critical application. The same as a *high-risk application*.

D

distillation. The process of removing records that do not contribute to an increase in path coverage.

distilled testbed. The *original testbed* after it has been augmented by checking the coverage, and refined by distillation. All dates are in original formats. The distilled testbed is used to produce the Baseline results, and is converted to produce the *converted testbed*.

E

enhanced testbed. The testbed after additional data has been added to it, so that coverage is at a maximum.

H

high-risk application. An application which, if it failed, would cause major operational difficulties to your business. For example, if an invoicing application fails, customers will stop making payments to your business.

L

low-risk application. An application which, if it fails, can be fixed without causing too much inconvenience to your business. Any failure will be treated as a low-priority fix.

N

node. A line of code where processing can proceed more than one way, depending on a decision made on the basis of the value of a variable, or the values of some variables.

O

original code. The code before it is converted to deal with any Year 2000 problems.

original testbed. The testbed that is available before any Year 2000 conversion and testing commences. It is either a testbed that has been maintained for testing purposes, or a cut-down version of production files. The original testbed is distilled, to produce the *distilled testbed*.

P

performance test. A test that focuses on the time taken for processing and the resources used by the processing, especially hard disk storage. A performance test is usually undertaken with a large volume of information. Results are possibly not checked thoroughly.

R

results. In this redbook, results refer to the updated files, the reports produced after updating, and the screen images captured during the entry of transactions. There are four sets of results, baseline, post-fix, 19xx, and 20xx. These results are compared. An exact comparison (after allowing for conversion and aging) is a strong indication that program conversion is correct.

S

sub-path. The part of a *path* between two *nodes*. Because there are nodes (or decision points) in the sub-path, all of the code in the sub-path is executed to process a particular transaction.

system clock. The clock provided by the computer or operating system. Provides the *system date*.

system date. The date provided by the computer or operating system. Often provide the validation check for a date—the date must not be in advance of "today"—or the date stamp for transactions. When testing, extreme precautions should be taken before changing the system date, to make sure that the changed date does not flow into current production systems.

system test. Test of the entire system of programs and modules. (Testing one program or module is a *unit test*.) The purpose of the system test is to check the flow of information between programs, and also to check the relationship between programs written for the system and utilities provided by the system.

system test plan. A document describing the testing to be carried out for the entire system, and the expected result for each test.

T

targeted coverage. Developing *coverage* by comparing source code before and after conversion, and making sure that all changed lines of code are covered.

testbed. A set of databases (or master files) and transactions. Transactions are processed by code to produce results (reports and updated master files). See *original testbed*, *distilled testbed*, *converted testbed*, and *aged testbed*.

U

unit test. The test of one program or module, to ensure that there are no programming or analysis errors. See also *system test*.

unit test plan. A document describing the testing to be carried out for each program or module, and the expected result for each test.

W

white box testing. Testing that is produced by examining the internal structure and logic of programs. See also *black box testing*.

Y

year 2000 ready. A program or system that is able to correctly process, receive and provide dates within and between the 20th and 21st centuries.

Z

zapped module. A module used by Coverage Assistant and Distillation Assistant, where breakpoints have been inserted into the object code. This makes it possible for Coverage Assistant and Distillation Assistant to monitor the execution of code, and thus create a coverage report or distill files.

List of Abbreviations

APA	all points addressable
ATC	Application Testing Collection
ATP	Auto Test Performer
CICS	Customer Information Control System
DT	Debug Tool
LE	Language Environment
IBM	International Business Machines Corporation
MLE	Millennium Language Extensions
ITSO	International Technical Support Organization
PDS	partitioned data set
PROFS	Professional Office System
PTF	Program Temporary Fix
WITT	Workstation Interactive Test Tool

Index

ITSO Redbook Evaluation

VisualAge 2000 Test Solution: Testing Your Year 2000 Conversion
SG24-2230-01

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:
____

____

____

____

What other redbooks would you like to see published?
____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)
____

____

____

____

____

VisualAge 2000 Test Solution: Testing Your Year 2000 Conversion **SG24-2230-01**

SG24-2230-01
Printed in the U.S.A.