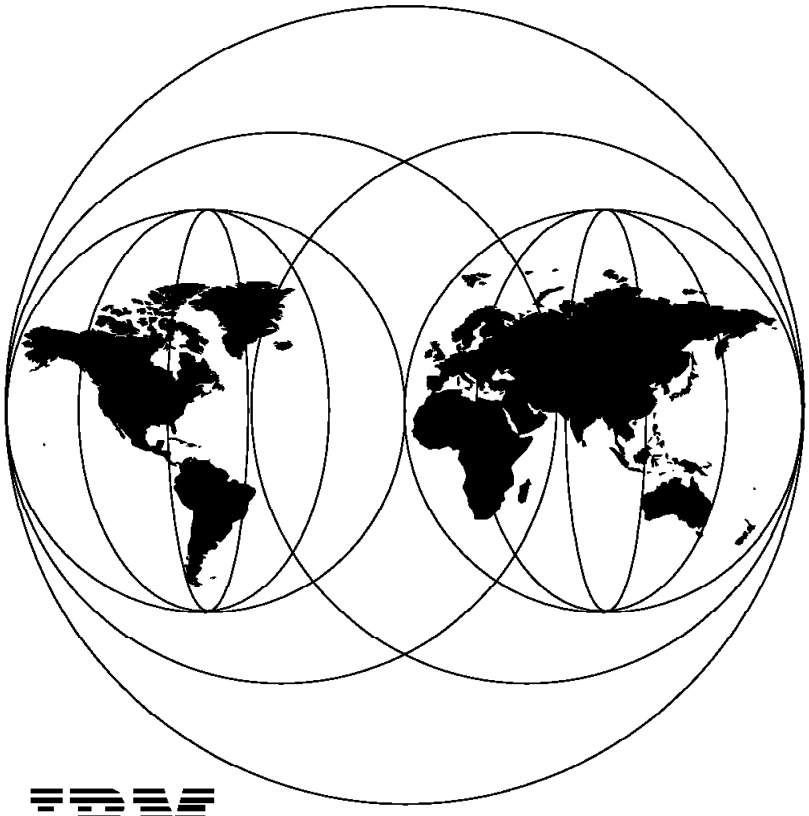
SG24-4829-00

AS/400 Applications: Year 2000 Enablement & Services Considerations

January 1997



IBM

International Technical Support Organization
Rochester Center

International Technical Support Organization

SG24-4829-00

AS/400 Applications: Year 2000 Enablement & Services Considerations

January 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix H, “Special Notices” on page 159.

First Edition (January 1997)

This edition applies to OS/400 Version 3 Release 1, 5763-SS1 and OS/400 Version 3 Release 6, 5716-SS1.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

Contents

Figures

Tables

Preface

The new millennium is just around the corner. A pervasive problem exists for many Information Technology systems and applications where they make use of dates represented by only two digits for the year. These applications will fail as they deal with years outside of the 1900 and 1999 range.

This redbook is a collection of information developed for service providers to help them to assemble a service offering to assist their customers in the Year 2000 enablement process.

Some knowledge of the AS/400 system, systems and application design, and programming is assumed.

How This Redbook Is Organized

This redbook contains 172 pages. It is organized as follows:

- Chapter 1, “Introduction”

  This chapter provides a description of the Year 2000 problem, its scope, and complexity. Also outlined are the Year 2000 enablement process and date representation techniques.

- Chapter 2, “IT Environment Inventory”

  This chapter describes the first phase of the Year 2000 enablement process. It covers taking inventory of the customer′s systems information and resources. It includes a high-level sizing of the effort, some considerations for the development of a service contract, and a statement of work.

- Chapter 3, “Tool Selection”

  This chapter describes a set of criteria that can be used in evaluating and selecting tools that help in enabling applications for Year 2000. It addresses each phase of the Year 2000 process and the criteria pertinent to each. It also contains a section on manual work which may need to be addressed beyond the scope of the selected tools.

- Chapter 4, “Impact Analysis”

  This chapter describes what to look for when selecting an Impact Analysis Tool. Such things as a report on fields, variables, and so on should be produced, which later can be used for automatic conversion. It also addresses additional work that may need to be done manually. The combination of the two analyses defines the total effort required.

- Chapter 5, “Search, Modify, and Build”

  This chapter addresses the core conversion functions. It describes four date conversion techniques, their associated benefits, and limitations and considerations in selecting any one of them. Also provided are guidelines and considerations for each of the search, modify, and build functions.

- Chapter 6, “Data Conversion”

  This chapter deals with data conversion considerations that include possible new or additional hardware, what data to convert, conversion and bridge programs, customer responsibilities, the conversion plan, and the actual conversion process.

- Chapter 7, “Test and Implement”

  This chapter examines the most critical and time consuming phase of the entire process. It covers the development of a master test plan, test preparation, running the tests, testing techniques, and Year 2000 testing. Also contained is a section on pitfalls.

- Chapter 8, “ Documentation”

  This chapter contains information to help the customer fully understand the scope of the changes made by providing comprehensive documentation. Brief discussions of the change log, end user training, and data dictionary are included.

- Chapter 9, “Post Implementation Services”

  This chapter describes briefly some of the support services considerations once the conversion and testing phases are complete.

- Appendix A, “Year 2000 Enablement - The Big Picture”

  This appendix contains a detailed flowchart of the entire Year 2000 process.

- Appendix B, “Application Dictionary Services (ADS/400)”

  This appendix contains a brief description of ADS/400 and how it might be used to analyze the impact of date fields.

- Appendix C, “OS/400 Products”

  This appendix contains a brief description of the enhancements to OS/400 V3R1 and OS/400 V3R6 to enable them to be Year 2000 ready.

- Appendix D, “IBM System Features for AS/400 System”

  This appendix outlines some of the IBM products, tools, and functions to address the Year 2000 transition.

- Appendix E, “Design and Coding Tips”

  This appendix gives some tips that might be worth considering when making changes to address the Year 2000 issue.

- Appendix F, “Year 2000 Certification Steps for Solution Providers”

  This appendix identifies a some guidelines for the Year 2000 Solution Providers.

- Appendix G, “Testing Handbook”

  This appendix addresses additional activities and responsibilities that are not covered in the Test and Implement chapter. It also contains sample worksheets that are useful to plan, prepare, and utilize during the testing phase.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Rochester Center.

Barbara Jordan is currently the Year 2000 Server Group Project Manager, IBM Corporation. Her work includes Year 2000 readiness projects for IBM Server Group operating systems as well as programs to support third party vendors in their efforts to produce Year 2000 ready applications and Year 2000 enablement tools and services.

Klas Karlsson is a Senior Instructor with IBM Education and Training in Sweden. He has 22 years of experience in the mid-range field. His areas of expertise include Application Development, Systems Management, Performance, and Client/Server Computing. He has developed and taught IBM classes on all areas of the AS/400 system.

Saadat Anwar is an Information Technology Specialist in IBM Islamabad. He graduated from the prestigious FAST-Institute of Computer Science (Lahore, Pakistan) with a Gold Medal (Top Student honors) in January 1993. He joined the fast growing SADD division. Saadat has worked on many application development and porting projects, using IBM and non-IBM products, tools, and databases (VisualGen, Informix, DB2 and Oracle). He manages his own field territory of multiple AS/400 customers and RS/6000 customers with many large LANs and WANs. Saadat is a Regional instructor on the AS/400 system and RS/6000 system.

Erik E. Myhra is a Senior Advisory SE at the Customer Support Center in Norway. His areas of expertise include application development and performance. Erik has worked at IBM for 20 years, including 3 years at the Rochester ITSC.

Neil Willis is a Senior ITSO Specialist at IBM Rochester. He has 11 years of experience in Performance, Capacity Planning, and Application Development. He has presented at COMMON and to other AS/400 Special Interest Groups. Prior to joining the ITSO, he was an Advisory Systems Specialist and AS/400 Team Leader in Sydney, Australia.

George M. Yeung is a manager in SWS Product Services, Toronto Lab. His current responsibilities are in AS/400 Application Development Services Enablement.

Susan M. Gantner is a Technical Advocate for AS/400 Application Development products and is located in the IBM Lab in Toronto. Susan specilaizes in application development and database on the AS/400 and is a regular speaker at COMMON and technical Conferences for AS/400 customers around the world. Susan has spent 21 years in the field of application development. Prior to joining IBM, she developed applications for corporations in Atlanta, GA, working with a variety of hardware and software platforms. Susan joined IBM in 1985 as a Systems Engineering Specialist for the S/38. She spent 5 years working in Rochester, Minnesota supporting customers and IBMers around the world with database and programming challenges.

Thanks to the following people for their invaluable contributions to this project:

Aki Fukai
IBM Software Services Planning, Availability Services, Rochester

Jon Paris
IBM Toronto

E.B. (Brian) Sawyer
IBM Midrange Technology Manager, Oak Brook, Ill.

Bruce Vining
Base System Enablers, IBM Rochester

Gabriella (Babi) Erler
AS/400 Requirements Manager, AS/400 Division, IBM Somers (New York)

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Introduction

As we approach the end of this millennium, there is a real problem that needs to be addressed by all computer software users. The computer industry has only been in existence since the 1940′s and as such, has not had to deal with a century change before. Many older applications use two-digit dates. In this chapter, we show you many unexpected problems that can result when using two-digit dates, especially on the upcoming century boundary.

1.1 Description of the Problem

A phenomenon exists in the Information Technology (IT) Industry that will cause some computer systems and most applications to fail on or before the Year 2000. This problem exists because, historically, computer programs make use of dates represented by only two digits for the year (for example, 96 instead of 1996). This practice causes programs (both system and application) that perform arithmetic operations, comparisons, or sorting of date fields to yield incorrect results when working with years outside of the 1900 to 1999 range. For instance, a person whose birthday is November 10, 1961 is considered to be minus 61 (-61) years old, rather than 39 years old on November 10, 2000. This occurs if the years 1961 and 2000 are represented by 61 and 00 respectively. The -61 might even be interpreted as the absolute value 61. Not only is the value still incorrect, but it is harder to detect than the incorrect -61. Sorting problems occur when only two digits are used to represent a year. The year 2000 (if represented as 00) is ordered prior to the year 1999 (if it is represented as 99).

There are problems with date formatting, with calculating the day of the week or of the year, and with calculating the week of the year. For example, some programs determine the date and time format (that is, MMDDYY, DDMMYY, or YYMMDD) by testing an appropriate part of the date field. The program may check to see if the first two characters of the date field are values within an acceptable month, day, or year range (such as 1-12, 1-31, or >= 32).

When dates are used as special values, problems can also occur. For example, the value 99 in the year field may be used to indicate ″no expiration date″ or 00 may be used to indicate an ″unknown year″.

1.2 Scope/Complexity of the Problem

The problem spans all levels of hardware and software from microcode to application programs, to files and databases, and is present on all platforms. The AS/400 system will be Year 2000 enabled in Version 3 of the operating system. See Appendix C, “OS/400 Products” on page 97 for a description of the enablement in each of the Version 3 releases. System/36 users are encouraged to move to an AS/400 Advanced 36 with SSP 7.1 or 7.5, or directly to AS/400 Version 3.

The biggest challenge, however, is to enable the suite of more than 25 000 AS/400 applications to work with the change of the millennium. In addition to these applications that are marketed by AS/400 Solution Providers, there are thousands of applications that were written or modified by AS/400 customers that also need to be Year 2000 enabled. Fixing the problem is conceptually quite simple once all of the date fields, variables, and so on have been located.

However, finding all of the date-related occurrences and testing to assure that they are handled correctly is not a trivial process. Solution providers and customers must devote significant development resources to solving this problem, or they can engage a service provider to do the work for them.

1.3 Purpose of this Document

This paper is a collection of the information needed for service providers to assemble a service offering to assist customers and solution providers in the Year 2000 enablement process. It is intended for people who have developed technical service offerings in the past and provide information specific to the Year 2000 problem. It is also targeted at AS/400 service providers, but most concepts also apply to other platforms.

This document does not give guidance on how to price a service offering. Sizing the job depends on many factors including the skill and experience level of the service provider, how much of the work can be automated, and the characteristics of the customer's IT and business environment. It is up to you to assess these variables and estimate the size and cost of the job.

1.4 Year 2000 Enablement Process

The process of converting a customer's IT environment to be Year 2000 enabled can be broken down into a series of discrete phases:

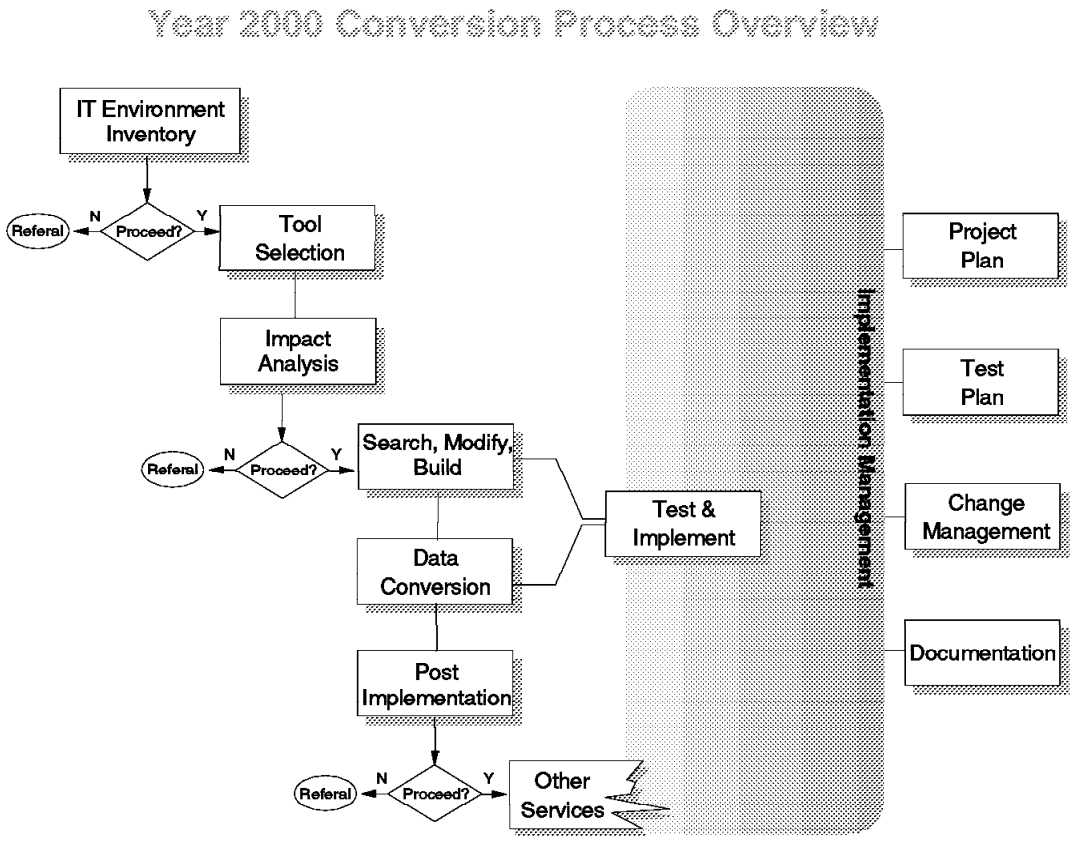


Figure 1. Year 2000 Enablement Phases

Service providers may want to become involved in one or more of these phases, depending on the size of their company and its skills and experience level.

It is probably advantageous for most service providers to purchase (or develop) a tool to automate as much of the process as possible. A chapter of this document is devoted to the tools selection process. This chapter describes the characteristics you should look for in tools for each of the phases, but does not discuss any specific tools. In the future, a matrix of the tools that have been analyzed by IBM will be made available to assist you in the selection process.

1.4.1 Date Representation Techniques

It is important to understand the different options for presenting non-ambiguous dates. The following sections discuss these options in detail.

- **Converting a two-digit year to a four-digit year format:**

  In this approach, four-digit years are stored in the application but the customer may still choose to display or print the dates in two-digit format.

- **One-digit century code:**

  In this approach, a one-digit extension is made to the date field to include a century code instead of the complete century. There is less overhead in the file using this technique; it requires that the date be converted to any of the YYYYMMDD formats for usage. For example:

  ```
  0 in the century digit means the century is 19.
  1 in the century digit means the century is 20.
  ```

- **Standard date representation:**

  This technique employs either the ISO, SAA, or the database date formats. The benefit of using these formats comes from the fact that they are supported as language primitives in the ILE environment on the AS/400 system. This means that you have to move to ILE, but there are performance advantages in doing so. By moving to ILE, you can have your specialized date routines built into service programs that take far less time to load compared to external program calls.

  The functions that you can perform on the date fields include an interval calculation between two days and the addition of a duration to some date. An additional benefit of this approach is standardization among applications between various vendors.

  If you do not want to move to the ILE, you can read the date fields in OPM programs using an embedded SQL. Note that this requires a major change in the application structure and is more geared towards application modernization.

- **Windowing technique:**

  This approach uses either a fixed window or a sliding window of dates to calculate the century based on the year.

  In both cases, parameters determine how the century is inferred from a two-digit year. For example:

  ```
  For a fixed window technique where the window is defined by the
  operating system, if 60 is the parameter, then all of the years
  below 60 have 20 as their century part and all of the years above
  and including 60 have 19 as their century part.

  In a sliding window technique, the window is based on the current
  ```

year. If the current year is 1996, the window is from 1936 to 2156. If the current year changes to 1997, the window changes to 1937 to 2157.

Obviously, window interpretation logic must be built into a common date routine that is called for each instance of a date.

Both OS/400 and SSP have a fixed window of 1940 to 2039.

- **Other date formats:**

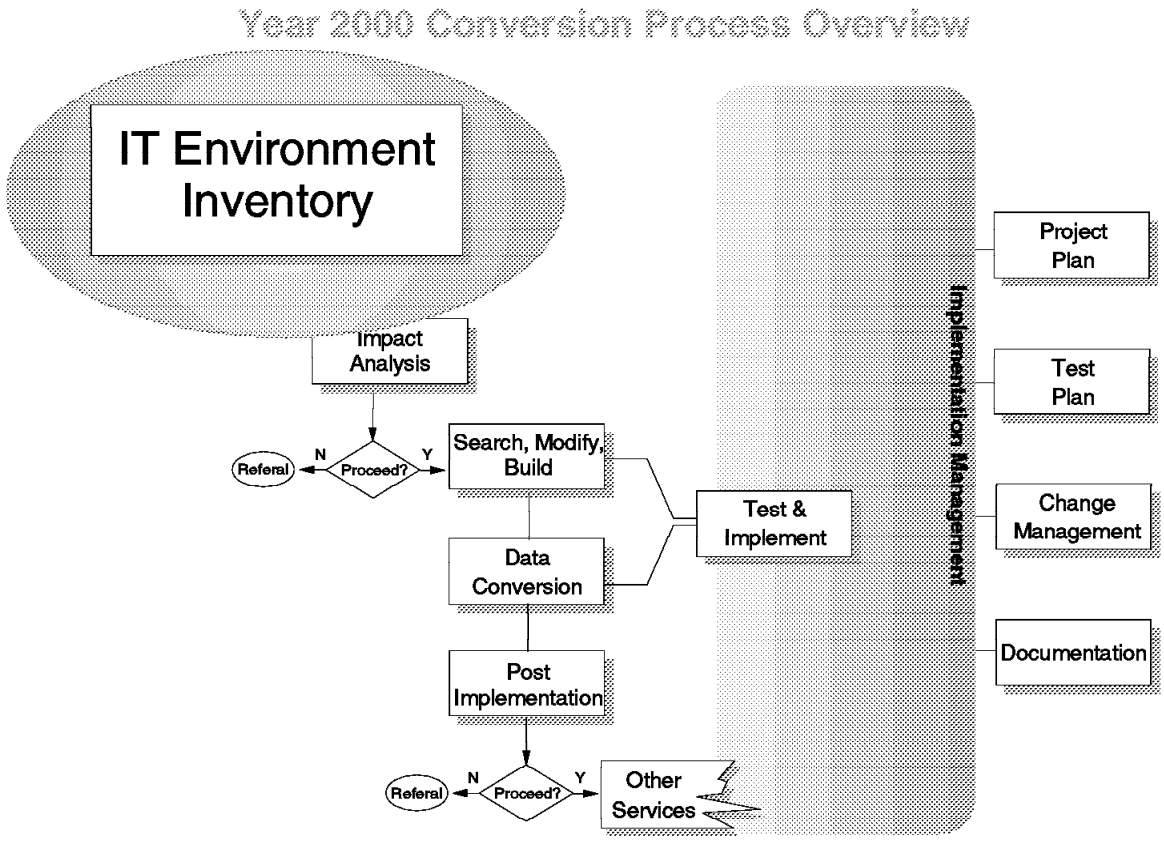
  Other date formats such as Julian dates have the same implications as previously mentioned.

- **Two-digit encoding/compressing scheme:**

  This technique is used to compress the four-digit year into the two-digit year field by encoding it in either binary or hexadecimal representation. As the approach implies, this technique also requires a common date routine to expand date fields before processing, thus it has almost the same kind of overhead as defined in the previous technique.

Note: A detailed description of some of these techniques along with their advantages and disadvantages is given in *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251.

Chapter 2. IT Environment Inventory



2.1 Inputs and Outputs

The first phase in the Year 2000 conversion process is the IT Environment Inventory phase. The goal of the IT Environment Inventory is to produce a gross estimate of the resources that are required to make the customer′s environment Year 2000 enabled. From the analysis, you can produce an IT Environment Overview report that is used to do this estimate (the Initial Proposal). The initial proposal gives the customer a rough idea of the cost of the project to make the decision to proceed with the conversion or look at other alternatives. At the end of this chapter, there is a discussion of the contracts that you need to put in place as the project develops. For now, just be aware that you should put a contract with a Statement of Work in place for the IT Environment Inventory phase of the project so that you are able to bill the customer for your services, even if the decision is not to go ahead with a conversion.

The most important task in the IT Environment Inventory Phase is to gather the necessary information to understand the customer′s environment. You are able to obtain much of this information using CL commands or other tools, but some of the information must be gathered manually by having the customer complete a survey of their environment; and some of the information emerges as a part of

the informal discussions. You need to gather enough details from the customer to feel comfortable that you understand the environment (for example, interrelationships of files and programs, importance of dates to program logic, and so on). However, you do not want to do a full impact analysis. Therefore, you do not attempt to locate all of the date fields at this time.

You need the following information to help you produce a high-level estimate of the resources that are required to complete the conversion.

2.2 Company Description

The following information should be collected using some type of survey:

- Nature of the business
- Key resources (management, technical)
- Processing cycle (for example, daily, monthly, or yearly)
- Availability requirements
- Performance requirements
- Urgency of upgrade

  Determine if there are applications that must be upgraded quickly because they are critical to the operation of the business (for example, legal or government compliance), or critical to the uninterrupted operation of the business (for example, payroll system).
- Language spoken
- Size of DP staff
- Skills

2.3 AS/400 Systems Configuration (For Each System)

The following information can be gathered using CL commands:

- CPU specifications (indicate if upgrades are planned)
- Model
- Memory
- DASD (size and utilization)
- IOPs
- CPU utilization
- Security level

```
DSPHDWRSC TYPE(*AHW) OUTPUT(*PRINT)
DSPSYSVAL QSECURITY OUTPUT(*PRINT)
```

2.4 Network Configuration

The following information can also be gathered by using the CL commands referenced in the System Configuration section:

- AS/400 systems
- OEM hardware

2.5 Type of Service to be Performed

- Impact analysis
- Program inventory
- Search/modify/build
- Partitioning and bridging
- Data conversion
- Testing
- Certification
- Post implementation work

Some customers may want you to do an estimate of the size of the entire project during the IT Environment Inventory phase. Others may want you to size the cost one step at a time. For each of these phases, it is important to understand the resources that the customer has available. For instance, during the Inventory Phase, a field reference file is very helpful. During the impact analysis phase, having a person available who knows the naming convention for date fields in files and programs makes your job much easier. The test phase is much simpler if the customer has established a good baseline for both performance and functional testing, and if there is a well-organized environment for the normal release-to-release testing. Knowledge of these resources helps you more accurately size the project.

2.6 System Software (For Each System)

The following information can be gathered using CL commands:

- OS/400 release (indicate if upgrades are planned)
- PTF level
- S/36 environment
- LPPs
- Primary national language
- Secondary national language
- Tools

```
WRKSYSACT OUTPUT(*FILE) INTERVAL(60) NBRITV(60)
DSPSFWRSC OUTPUT(*PRINT)
DSPPTF OUTPUT(*PRINT)
```

2.7 Application Description (For Each Application)

For each application:

- Do you have an inventory of all your programs?
- Is source available? (Source code is required to perform impact analysis.)
- Is the application purchased or written in-house?
- Do you have legal rights to modify the code? (Check the license agreement with the vendor.)
- Is source in sync with the object code?
- Will other revisions be made to the application during this time frame?
- Number of lines of code
- Date written
- Date compiled
- Application design
  - Client/server
  - Networked

  - Journaling
- Application complexity (degree of nesting)
- Use of date fields (high, medium, low)
- NLS considerations
- DBCS enabled
- Is documentation available?
- Coding practices/naming conventions

The Display Object Description (DSPOBJD) command can be used to find the status of the relationship between program objects and their source code. The location of the source file member and the compilation date are included in the output. The source member must then be examined to assure that it has not changed since the object was compiled. The following commands can be used together to manually examine source and object dates:

```
DSPOBJD OBJ(*ALLUSER/*ALL) OBJTYP(*PGM) DETAIL(*SERVICE)
   OUTPUT(*PRINT)
DSPFD FILE(xxx/xxx) TYPE(*MBRLIST) OUTPUT(*PRINT)
```

If Application Dictionary Services/400 is installed, it can also be used to examine programs. Information such as compile date, last change date, data areas used, and location of source code is available using ADS. Information such as calling programs, called programs, and files used is also available. For more information on how to use ADS, please refer to the *AS/400 Application Development ToolSet/400: Application Dictionary Services/400 User's Guide*, SC09-1860.

The two methods previously described provide the information needed to understand the customer's application environment. However, both approaches are extremely time consuming and labor intensive unless the environment is very simple. A good application inventory tool should automate much of this work.

The situation may arise where source programs are not found for the object programs in production. The first thing to do is to see if the programs have been renamed and continue the search. If the final conclusion is that source codes are not available, the options are either to rewrite the program/application or look for another package.

2.8 Data Definition

Ascertain which methods are used to define files:

- File descriptions:
  - Use of field reference files
  - Logical
  - Physical
  - Display
  - Communications
  - Printer
- Use files that are externally described or program described or both.
- Data in files may only be retained for a fixed time.

The process of understanding the customer's file structure is much simpler if field reference files are used because all of the field information is in one place.

On the other hand, if a customer uses program-described data extensively, the conversion is more difficult.

The Display Object Description (DSPOBJD) command can be used to determine when a file was created and where the source is located.

```
DSPOBJD OBJ(*ALLUSER/*ALL) OBJTYP(*FILE) DETAIL(*SERVICE)
   OUTPUT(*PRINT)
```

To understand the interrelationships of files, use the Display Database Relations (DSPDBR) command:

```
DSPDBR OBJ(*ALLUSR/*ALL) OUTPUT(*PRINT)
```

If Application Dictionary Services/400 is installed, it can be used to see the relationship between physical and logical files. It also allows you to work with the fields of a record format, identify date related fields (by examining the source), identify all of the files that use those fields, and identify the programs that reference those files. Again, both of these approaches provide you with all of the information that is needed regarding the structure of the files in the customer′s environment but is time consuming. A good application inventory tool is a better alternative for all but the simplest environments.

2.9 Date Solution Preference

There are several ways to enable an application for the Year 2000 including the following:

- Four-digit year format using SAA data type ″date″
- Four-digit date format (YYYY)
- One-digit century code
- Windowing (fixed or sliding technique)
- Database converted to four-digit dates or a routine called to handle the current two-digit date structure
- Four digits shown on forms, reports, and displays

Refer to Chapter 1, “Introduction” on page 1 for a more complete description of these options.

2.10 Resource Allocation

Customer participation in the project is important. The customer′s knowledge of the applications and the IT environment makes the conversion go much more quickly. Be sure to understand what skills and resources the customer is willing to commit.

- Do you want to commit resources to this project?
- How many?
- What skills?
- Which phases?
- Where do you want the services performed?
  - If off-site, are you willing to send source code? Are you willing to send it off-shore?
  - If on-site, what is your current system capacity? Do you need additional capacity or another system?

2.11 IT Environment Overview Report

This report is delivered to the customer to document your understanding of the data you have collected. The customer reviews and revises this document as appropriate to produce an accurate representation of the environment.

2.12 Initial Proposal

When the IT Environment Overview Report is complete and the customer agrees that it is accurate, you are able to generate an initial proposal. This proposal gives a high-level estimate of the resources that are required to complete the work and should be broken down by phases (impact analysis, search/modify/build, data conversion, testing, certification, post-implementation work). The customer may want you to estimate one phase at a time rather than the entire project. You should be sure that you understand which services the customer wants your estimate to include.

The estimate should be stated in terms of person months and should indicate a range rather than an exact number. The purpose of this estimate is provide the customer with enough information to decide whether to proceed with a conversion of their existing application or search for another offering that meets the needs of their business and is already Year 2000 enabled. In addition to conversion services, the service provider may perform the service (for a fee) of locating alternative solutions for customers who decide that conversion is too expensive. Partners in Development can assist the service provider in this effort by collecting and maintaining a list of Year 2000-Ready AS/400 solutions.

2.13 High-Level Sizing Methodology

Parts of the high-level sizing can be automated if a tool is used to collect system information. Points are assigned to system objects based on the amount of work that is required to convert an object of that type, and the points are converted to person months. Other costs need to be added to this total such as those listed in the following example.

2.13.1 Assign Points (Example)

```
Data File (small)       1 * # data files    = 1 *  5 =  5
Data File (medium)      2 * # data files    = 2 *  4 =  8
Data File (large)       3 * # data files    = 3 *  8 = 24
Pgm - simple            2 * # simple pgms   = 2 *  8 = 16
Pgm - complex           3 * # complex pgms  = 3 * 20 = 60
Dsp files/reports       4 * # files/reports = 4 *  5 = 20
-------------------------------------------
Total                                       =15 * 50 =750 points
```

2.13.2 Other Costs

While the point calculation previously discussed can either be done manually or with a tool, the other costs that need to be considered are more subjective. Additional or upgraded hardware may be required, both during the conversion and afterwards if the customer's DASD requirements grow as a result of the conversion. There may be schedule considerations (that is, work must be completed by a particular date), or where the work is performed may be a

critical factor. Other issues such as performance or availability requirements must be understood. You may also need to consider training costs.

2.14 Contracts

It is necessary for you to develop a separate contract with a statement of work for each of the phases of the Year 2000 conversion process if your customer wants to base the decision to proceed with the next phase of the project on the results of the previous stage. For instance, you want to put a contract in place for the work that you do in the IT Environment Inventory Phase so that you are able to charge the customer for the work that is involved in producing the initial proposal. The customer decides whether to proceed with the next phase (the impact analysis) based on the results of the initial proposal. Likewise, you need a contract in place for the impact analysis phase, and the customer can decide whether to proceed with the conversion phase based on the results of the impact analysis report. Finally, a contract needs to be developed for the conversion, testing, and follow-on support activities. These may be done separately or combined into one contract. In all phases, the statement of work is an important communications vehicle and is key to developing an effective working relationship between the service provider and the customer. It not only describes the deliverables of the project, but defines the way the teams divide the work and share information. The following outline shows the contents of a contract and a statement of work.

2.14.1 Contract Content

Purpose
Work scope and structure
Rights/intellectual property
Indemnification
Warranty
Term and termination
Contacts
Provide name of contract coordinator.
Limitations
Freedom of action
Trademarks
Expenses
Assignment
Governing law
Jury Trial
Subsidiary
Entire agreement
Signatures

2.14.2 Statement of Work

The statement of work contains the details of the arrangement between you and the customer. The following list contains an example of the information that may be included in a statement of work.

- Technical coordinators (list names and addresses)
- Tasks:

Outline the tasks that are completed, the criteria for acceptance, and the completion dates. Be sure to break the project down into milestones that can be tracked to make sure the schedule is maintained.

- Organization, responsibility, and resources:

  Because of the nature of Year 2000 work, it is likely that both the service provider and the customer will participate in the project, particularly in the testing phase. Resource requirements should be established and the division of responsibilities documented. Skill requirements must also be outlined and training plans established if needed. Leadership responsibilities and the reporting structure should be well understood. Be sure to devote adequate resources for project management to keep the project on schedule and to maintain the relationship between you and the customer.

  Hardware resource requirements must be identified. Where will the service be performed? Who is providing the hardware? Where is it located? Space, facilities, and support should all be planned for. Travel expenses should be budgeted as well.

- Project management system:

  The customer and the service provider must establish how project milestones are tracked and communicated. Processes for managing changes, problems, and issues should be put in place. Being well organized in this area is key to maintaining a positive relationship between your company and the customer. You may find your project management styles are different, so it is important to agree from the start on how the project should run.

- Risks, assumptions, and dependencies:

  The statement of work must include the assumptions upon which the project and its plans are based. External dependencies and risk assessments should also be part of the document.

  The customer′s development environment, including change control tools and processes, must be well understood when assessing the level of risk in a project. This is particularly important if the customer is doing development work on the application at the same time you are doing the Year 2000 work. If the customer is well equipped to control and manage source code and component libraries as well as component versions, the project will run much more smoothly. On the other hand, if there are no good source/object synchronization practices in place, this should be reflected in your estimates.

- Payment:

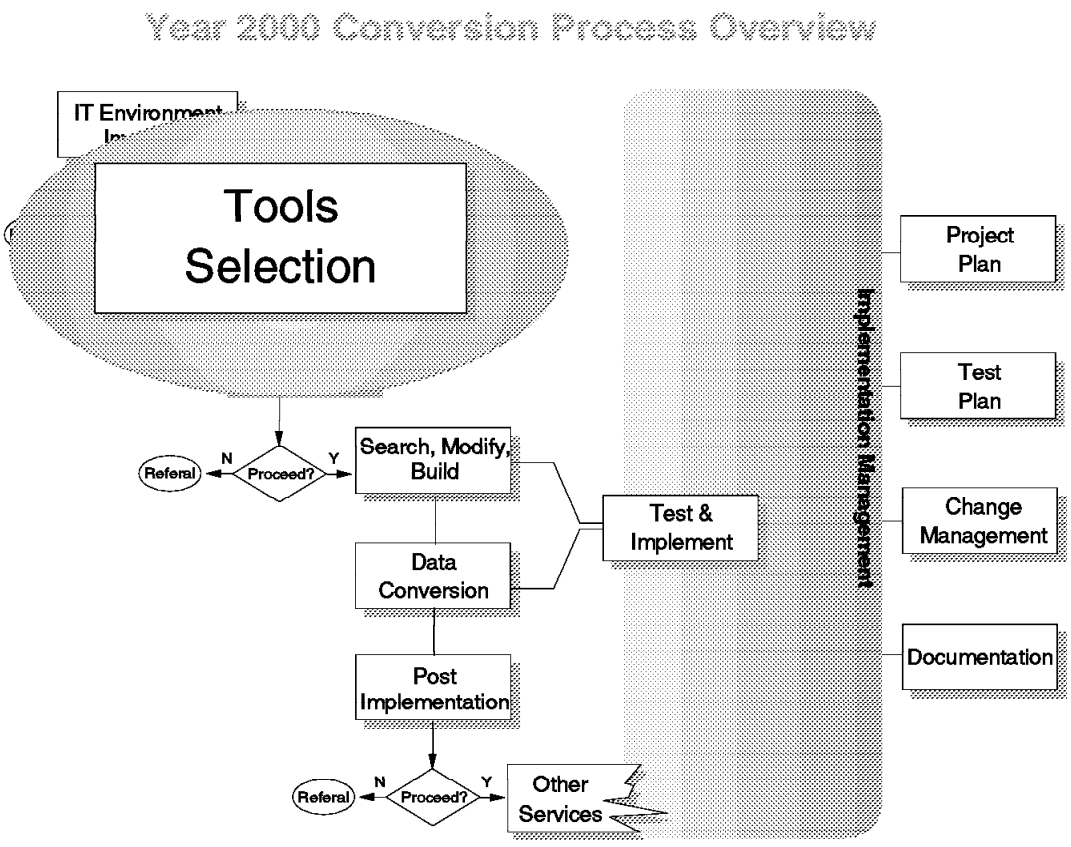
  Document the amount and schedule for payment.

- Intellectual property:

  You have to be aware of intellectual property concerns. If your employees are exposed to the customer′s source code, they may be restricted in what they can work on in the future because they have been ″contaminated″. Also, be sure your customer has the rights to modify the code if it was purchased from a third party.

The statement of work should be written so that modifications and additions are possible as the project proceeds.

Chapter 3. Tool Selection



This chapter describes a set of criteria that can help you evaluate and select tools to assist you in enabling applications for the Year 2000. All projects involve both work that can be automated using a tool and work that must be done manually, but service providers want to automate as much of the process as possible. Since the proportion of work that can be automated is dependent on the tools you choose, it is important to choose the tools carefully.

Because new tools are emerging every day and existing tools are being enhanced, we do not attempt to recommend specific tools in this chapter. Instead, we have taken the approach of describing the attributes a tool should have to perform the work for each of the stages of a Year 2000 conversion project. It is up to the service provider to study the various tools that are available and weigh factors such as cost and functionality in choosing a tool that fits your needs. At this point in time, there does not appear to be a tool that covers all phases of the conversion process. Therefore, it is likely that you might have to purchase more than one tool. Needless to say, you may want to select tools for each of the phases as they come up because there are lots of updates going into the tools in a short period of time.

3.1 Types of Tools

In the following sections, we have included a list of desirable features for Year 2000 enablement tools, but remember that this is not an exhaustive list. During all phases of a project, it is helpful for the tools you choose to have a good user interface, be interactive with the user, and have some kind of a batch processing facility for larger job steps such as modification, compilation submission, and data conversion parts.

3.1.1 Project Management

A project management tool can be used to establish control over the project. Information such as resource allocations, schedules, and milestones should be tracked in some formal way. There are a number of PC-based project management solutions available that you may find helpful in keeping the overall project under control.

3.1.2 IT Environment Inventory

An IT Environment Inventory tool should collect the data that is needed to understand the customer's environment. Remember that the goal of the IT Environment Inventory phase is to produce a high-level estimate of the overall cost of the conversion and a more precise estimate of the cost of proceeding to the impact analysis phase. Chapter 2, "IT Environment Inventory" on page 5 contains a complete description of the information that needs to be collected during this phase to produce an IT Environment Inventory report and indicates where a tool is useful in collecting the data.

3.1.3 Impact Analysis and Search

The Impact Analysis tool should collect detailed information about the application that is required to produce an accurate estimate of the work requirements for Year 2000 enablement. The impact analysis normally produces reports regarding the work that the tool can carry out automatically. Some of the tools might also provide an estimate of manual work requirements as well.

Besides producing a summary of the work that can be carried out automatically by the tool, the tool should scan and generate a list of the areas where the modifications for Year 2000 enablement are required. One of the major components is generating separate lists of selected and rejected date fields. This type of list helps in identifying places that you do not want to modify automatically because of some limitation of interface to some external or internal object.

Important

An Impact Analysis tool is able to generate information about automatic modification based upon its own features and limitations as well as user input so it should not be taken as the final word. Instead, it should be viewed as "What the tool can do automatically for you" based on the input you can provide it. Additional manual changes may still be required.

3.1.4 Application Modification

An Application Modification tool should be able to use the output generated by the Impact Analysis tool and Application Inventory tool to modify the source code for Year 2000 enablement. This modification is based on certain rules that may be defined externally. The modified application is still subject to compilation and testing.

3.1.5 Data Conversion

The Data Conversion tool should be able to convert the dates data in the source format (for example, two-digit dates) to the selected target format (for example, four-digit dates). It may either do it automatically or should be able to generate conversion programs or an interface to accomplish the task.

3.1.6 Application Build

The Application Build tool should be able to create the Year 2000 enabled source code and compile to build the complete application. It should also be able to generate applications either fully or partially based on dependencies.

The Application Build tool should be able to copy the securities that were implemented on the source objects to the target objects. This is possible in the case of a one-to-one conversion of the application. If the application is also being modernized, this may not be feasible.

3.1.7 Application Test

The Application Test tool should be able to aid in the testing of Year 2000 enabled applications. It may generate test data, run keyboard scripts, or perform display and report comparisons. The net result is to remove any errors or bugs that might have been introduced due to the Year 2000 enablement or to identify any pieces of code that, somehow, were left out during the enablement process.

3.2 Functionality in Phases

In order to understand the desired functions of a tool, you need to understand the phases of the conversion process. The following sections outline the major steps involved in Year 2000 enablement of an application and, hence, the things that you should look for when selecting the set of tools to use.

3.2.1 Application Inventory

The Application Inventory phase collects information about the customer′s IT environment. Some of this is collected manually, and some is gathered using operating system commands or tools:

- Company′s description
- Business language (or languages)
- DP staff′s skills
- System configuration, including hardware, software, and networking
- Application details including objects inventory
- Data definitions
- Coding practices and naming conventions
- Date solution and application modification preferences
- Risks and assumptions

3.2.2 Impact Analysis

The goal of the Impact Analysis phase is to identify all date-related variables, objects, and data that you want to modify to be Year 2000 enabled. A good Impact Analysis tool identifies the number of programs, files, commands, data areas, and other system objects, and produces a categorized listing of each of these. Also, as the tool scans the source code, it can identify references or locations in source code where changes are required. The following list shows some of the desirable items in the Impact Analysis report:

- List of objects impacted (categorized by “identified” and “confirmed”)
- List of objects of unknown type
- Number of lines of code (overall and per object)
- Number of comment lines (overall and per object)
- Number of lines of code to change (overall and per object)
- Date fields and their attributes of lengths, formats, and usage in objects and so on
- Rejected date fields and their attributes (may have a full list and a list by object)
- Complete list of fields, their attributes, the objects where these were encountered and scope
- List of source locations and objects that require manual work including:
  - Sending to or receiving from a data queue
  - Updating or reading a data area
  - Calling some external programs with dates or numeric literals as parameters
  - Record position overflow in reports and display files
  - Date constant definitions
  - Date literal usage
  - Query/400 queries, Query Manager queries, and OPNQRYF queries, reports, data areas, procedures, data queues, job descriptions, SQL statements, and so on
  - Messages in message files
  - Field-editing changes (for example, the “VALUES” or the “CHECK” function of DDS)
  - Unrecognized date arithmetic and comparisons
  - Help files and panel groups
  - Many more varying with the specific tool

Some tools are also able to produce information on the database expansion required such as:

- Listing database files and data areas affected by the conversion by object
- The projected increase in file record size
- Increase in the overall database size
- Archived data

3.2.2.1 Rule and Pattern Based Field Selection

In order to handle the different coding practices of developers as well as organizational standards and naming conventions, the scanning part of the Impact Analysis tool must have a flexible interface. Flexibility can be achieved by having a pattern and rule-based scan engine that is configurable, preferably, externally. A crude example of pattern-based selection is the Work Object (WRKOBJ) command where you can specify the first portion of object name as fixed and let the system search for all of the objects starting with that name. For example:

```
WRKOBJ OBJ(*ALL/T*) OBJTYPE(*ALL)
```

This command searches all of the objects starting with the letter ″T″.

A more generalized example of pattern matching is:

D.*[T|Y].*	Matches all strings starting with a ″D″, having a ″T″ or a ″Y″ somewhere in it following the ″D″.

Where:

.	Means any single character match.
*****	Means match zero or more occurrences of character preceding it.
|	Means match either of the characters separated by ″|″.
[ ]	Are single character selection list delimiters.

Pattern matching may be enhanced by including attributes such as the length of the fields to be selected or the data type of the fields to be selected.

A rule-based tool augments field selection by identifying those fields that are dependent upon a date field through some date calculations or reference. Any such field is a potential date field. For example:

```
C                   DTMDY  MULT 10000.01     PRTFLD2
```

The preceding code fragment[1] is reformatting the date field ″DTMDY″ from a ″MMDDYY″ format to ″YYMMDD″ format and the result is being placed in ″PRTFLD2″, which is used in printing dates in reports. So ″PRTFLD2″ is a potential date field. A smart tool also detects the format of the result field if the format of the source field is known.

3.2.2.2 Date Field Format Detection

Impact Analysis tools detect the format of the date fields from the following main sources:

- Directly from some indicators in the date field names[2]
- Directly from some indicators in the date field descriptions
- Directly from the database file data stored in database fields
- Derived from other date fields with reference to some calculation

---

[1] This is not a recommended way of doing a format change as it generates numeric overflows and underflows, requiring additional processing time that results in performance degradations.

[2] As explained in the *Review of Preferences & Practices*.

3.2.3 Application Modification

Once the date fields have been identified, a tool can be used to actually convert them to a format that is suitable for the Year 2000. Remember that a modification tool can only be as thorough as the selection tool because the selected fields act as input to the modification tool. Ideally, you can use the same tool to identify the needed changes and then modify the application. More manual data entry is required if the modification tool does not directly use the output of the impact analysis tool. Characteristics of a good modification tool include:

- Rule-based modification criteria
- Option of year field representation and storage types in the Year 2000 enabled application
- Option for multiple programming languages processing
- Translating source references to the destination references as well as the support for field reference files
- Ability to handle various date field representations and formats
- Ability to process objects of various language environments
- Ability to process internally described files, reports, displays, and communication files, as well as externally described ones
- Generating a list of objects processed. This helps to identify objects that might have been missed by the tool. This, in turn, helps identify objects such as data areas and data queues that may need manual attention.
- Ability of tool to identify areas needing manual attention
- Logging modifications made to the application in some viewable and printable logs
- Logging modifications within the application source code.
- Modifying other system objects such as data areas that can be changed by using some system commands
- Ability of the tool to do something more than just the Year 2000 enablement part

For further information on this topic, please refer to Chapter 5, “Search, Modify, and Build” on page 45.

3.2.3.1 Rule-Based Modification

Rule-based modification criteria has the same benefits as rule-based field selection.

A rule-based modification tool understands a set of primitive operations on which its rules are based. These primitive operations define the actual function to be performed. If the tool encounters a line shown in the following example:

```
C                   <MMDDYY>      MULT  10000.01       <YYMMDD>
```

After Year 2000 enablement, the tool should modify this line to:

```
C                   <MMDDYYYY>    MULT  10000.0001  <YYYYMMDD>
```

In the preceding example, the tool replaced the constant ″10000.01″ (which changed the format of an ″MMDDYY″ field to a ″YYMMDD″ format) to ″10000.0001″ (which changes the format of an ″MMDDYYYY″ field to a

″YYYYMMDD″ field). The rule used to replace 10000.01 is something similar to the following example:

```
IF FOUND('C      <MMDDYY>     MULT 10000.01  <YYMMDD>') THEN
   REPLACE       FACTOR2       WITH              10000.0001
```

The preceding example is a simple one. Actually, a tool can have many modification criteria such as this or even more complicated criteria that spans a number of lines of code (for example, if it encounters a set X of lines, it should replace it with a set Y of lines replacing such and such variables or constants). These rules can be similarly extended to non-standard date calculations as well.

3.2.4 Application Build

The Application Build phase deals with the generation of the application after its enablement for Year 2000. Some of the desirable features are:

- Facility to generate the application by starting at the field reference file to physical and logical files all the way up to programs, menus, and commands.
- Building applications based on all of the object specific keywords that were enforced in the source objects such as wait times on display files. These may be stored for multiple build.
- Copying all of the objects from the source library to the target library that cannot be compiled or that cannot be processed by the modification tool. Examples are data queues, data areas, user spaces, and job descriptions. Alternatively, a list of all such objects may be generated for manual intervention.
- Some way of submitting the long compilation jobs in batch.
- Logging the build activity in some activity log with all of the success, failure, and generation options.
- Possibility of building the complete application or part of it multiple times as per requirements (thus, the ability of the build tool to build dependencies starting from some program or data file and so on).
- Copy the securities from the source object code to the target object code.

If you intend to do a manual build of an application with a certain flexibility of work distribution and version control, you may want to look into ADM/400.

3.2.4.1 Security Considerations

Security is probably not an issue for small applications where objects may be granted the required securities manually, but for larger applications where individual securities are granted to objects, it might become necessary to match the securities in the source application to the Year 2000 enabled application. This can be done either manually by checking each object for its security in the source objects and making these security changes to the target objects, or a tool may be used to do the task automatically.

If you want to build the application without using a tool, you may want to consider using ADM/400 to assist you. See the Appendixes in this redbook for details. The securities, in this case, have to be copied manually using system commands.

For further information on the topic, please refer to Chapter 5, “Search, Modify, and Build” on page 45.

3.2.5 Data Conversion

Once the application has been built successfully, the next step is to test the application on the existing data. In order to carry out this test, all of the existing data in the database files needs to be translated according to the target date format selected. Some of the desirable features of a data conversion tool are:

- Customizable window of source dates to target dates.
- Direct conversion of data from source date format to the target date format or through programs generated by the tool.

  Manual conversion is also possible using the interactive SQL (Query/400) or Query Manager/400. So either of them may be used as a tool. For some of the situations where the dates are coded, some modification programs are required that must be written manually.
- Logging the converted data, the errors encountered, and the procedure adopted to get over these errors.
- Automated production of bridge programs if needed.

Another source of data that might be automatically updated by a tool is the date data area. This may not be possible for all of the cases so they might need manual attention.

For further information on this topic, please refer to Chapter 6, “Data Conversion” on page 63.

3.2.6 Application Test

The test phase is an important part of the Year 2000 enablement process. This phase deals with the elimination of any errors or flaws in the application that might have been left over after the Year 2000 conversion or that might have been emerged because of Year 2000 enablement of the application. The test tool must have the following features:

- Running test scripts on the converted application.
- Putting in redirection code in the programs so that they read some user-defined clock instead of the system clock.
- Macro recorders and players to record the key strokes, modify them with the macro editors, and play them on the Year 2000 enabled application.
- Generating data for the years through Year 2000.
- Speed testing by a quick walk beyond Year 2000.
- Comparing displays and reports.

Some of the common testing techniques used include:

- **Operations testing:** Verify that the application is ready for use along with all of the documentation, procedures, and training. Do not forget to test the operations for Month-end, Quarter-end, and Year-end.
- **Stress testing:** Verify the application and the AS/400 system′s capabilities in stress conditions.
- **Recovery testing:** Verify that the application recovers from conditions where data may be lost.
- **Requirements testing:** Verify that the application meets the requirements of the users as well as the specifications of the previous system.

- **Regression testing:** Verify that the application produces the same results on a set of data before modification and after modification.
- **Error handling testing:** Verify the error handling capabilities, including the new format checking and range checking.
- **Manual support testing:** Verify the content of help text, application documentation, and data dictionaries.
- **Intersystem testing:** Verify the operability of the application with other application softwares.
- **Parallel testing:** Verify the application as a whole by using the same data with both the original and modified application software over a longer period of time.

Needless to say, if the recovery feature was not there in the original design of the application, it will not be introduced by Year 2000 enablement.

For application software that has been broken into partitions, you must write bridges to handle interfaces between partitions that have not been converted and those whose date fields have been changed. As an alternative, you may want to Year 2000 enable the database and create logical files over it (that present data in older format by taking away two century digits), replacing them in the programs that are not Year 2000 enabled instead of the old files.

In case you do not want to use some sophisticated test bed or test utility, you may consider using the macro recorder facility provided in the Client Access/400 for Windows or OS/2 in the RUMBA/400 or the PC/5250 interfaces.

For further information on this topic, please refer to Chapter 7, “Test and Implement” on page 69.

3.3 Common Considerations

There is a set of desirable features that are common to almost all of the tools shown in the following examples.

3.3.1 Externally Configurable Tool Parameters

By externally configurable tool, we mean that the tool’s Year 2000 modification parameters must be selectable by running some utility (included in the tool) or by using the tool itself and that you do not have to regenerate the tool for each application setup. The following list contains some of the externally configurable features:

- Selection of source and target libraries
- Definition of search criteria
- Definition of modification criteria
- Specification of logging options
- Selection of the format of year fields for the modification process

3.3.2 Logging and Reporting

If you are going to use a tool to make code, data, or any object changes, it is extremely important that the tool maintains a complete change log of these modifications. Change logs can later be used in updating the existing system documentation as well as sorting out post-conversion problems as they occur. A change log should include:

- Lines of code modified by the tool organized on an object name basis.
- Updated source listing with comments about the update done or required manually (for unsure situations).
- Modification log of data files with all of the errors and causes of these errors. This should also be a categorized one (say, one log per data file).
- Application build logging including the generation options as well as the success and failure. This should also include any other objects modified by the Year 2000 enablement tool set using some system commands.
- Logging of unknown or lost objects as well as the objects that cannot be modified, generated, or granted authorities by the tool due to some security restrictions.
- Feature to somehow log manual changes into the respective logs.

3.3.3 Tool Execution Platform

The execution platform of a Year 2000 tool is a major consideration. It affects the financial planning aspect, resource scheduling, and training. A Year 2000 enablement tool may execute on an AS/400 system in native mode, on a PC as a stand-alone, a client/server environment, or on some other platform. Planning decisions should account for:

- Hardware and software requirements for the tool
- Expertise required to execute the tool effectively
- Data transfer technique, either media-less or using tapes or diskettes
- Transfer overhead versus processing overhead:

  In the case of a stand-alone or a client/server environment, there is also data transfer overhead to consider. Using a tool that runs natively in the AS/400 environment consumes processor resources. Stand-alone tools also do not usually include facilities for sharing code, building applications, or version control.
- Costs involved in acquiring the tool, setting up the environment, and training the staff
- Considerations for inventory, application build, and data conversion

3.3.4 Supported Language Environments and Objects

A tool may support multiple coding languages and multiple Year 2000 enablement options. For example, some of the languages supported might be:

- RPG
- COBOL
- C
- PASCAL
- REXX
- CL

- Others

Within these languages, the tool may support only certain language constructs. For example, an RPG processing tool might have the following limitations:

- RPG C, O, I, F and E specifications only
- Checks parameters to calls to other programs but cannot point out the parameters passed to it
- Cannot process I/O done on data areas
- Cannot process I/O done on data queues

At a higher level, the tool may be able to recognize certain objects in addition to the regular programs and files such as:

- Query/400 queries
- QM queries and QM SQL
- Data queues
- Data areas
- Job descriptions
- Printer overlays
- User spaces
- Panel groups

3.3.5 Geographical Boundaries

When selecting a particular tool, you must make sure that its support is available in your country or the region. This may have an additional benefit that the tool uses the same code page language as the customer. Moreover, tools often come with predefined selection and modification rules that can save a lot of work if these rules are closely related to the geographical location. If you want to process DBCS applications, you must look for a tool that supports this feature.

3.3.6 Performance

It is important to understand the processing speed of the conversion tool itself to determine if it can process the required amount of code in a reasonable time period. Things to consider include whether it uses one pass or does it take multiple passes through the source code; and if it takes multiple passes, does it do it automatically or is there some manual work required. Moreover, what kind of data conversion facility does it provide and so on.

It is even more important to know the performance characteristics of the code that the tool has generated. If the tool has generated bridge programs, what kind of performance does it produce and how efficient is the code to call those bridge programs.

3.3.7 Tool Setup

Be sure to check the time it takes to set up each tool and to run it successfully. At some places, the logistics might not allow timely installation of a certain tool. For those locations, you may want to select some other tool. The following list contains some of the setup activities:

- Hardware, software, and communications environment
- Logging options regarding comments in programs, modification, and generation logs
- Application build options
- Language environments

- Application specifications, including source code, object code, source libraries, and target libraries (whether you have to type in all of the names or whether you can make a selection)
- Date options and user processing preferences for the tool
- Rules and criteria for selection or rejection of date fields, modification of application programs, generation of application, and logging options.
- Help text, documentation, and support available for setup

3.3.8 Manual Work Required After Tool Usage

The tools selected, depending upon their limitation, leave some of the work to be done manually. The following list gives you some idea of the manual work that may have to be performed after the execution of certain tools for the Year 2000 enablement process. Needless to say, the list is not based on a particular tool and is just meant to be just for your understanding.

- Updates to the field selection, rejection and modification rules, and the object list to be processed after looking at the impact analysis report.
- Date cosmetics and format changes.
- Date calculations and date special values not handled by the tool. This may be the conversion of text date to numeric format.
- SBCS and DBCS national language modifications.
- Named constants, literal constants, and table data in the source code.
- Redefined fields that cause errors on reports and displays.
- Date fields used as sentinels and special values.
- Data areas and data queues.
- Utilities such as Sort/400, DFU, and Query/400 as well as utilities for data transfer and so on.
- Client/server environments as well as multi-platform applications.
- Non-standard development tools.
- Case tools (this may include case tools).
- Carrying out all of the steps for dates that are stored in some non-standard coded form.
- Correcting existing date fields that are using wrong formats.
- Record positioning overflows and splitting of cluttered displays from one display to two.
- Windowing parameters for the date bridge if the windowing technique is used.
- Application building, testing, and application modification for testing errors.
- Copying or creating target object that were not copied to the destination library by the tool (for example, text files used for printing letters).
- Modifying or re-compiling the programs that use special keywords in generation options.
- Modification of naming conventions.
- Assignment of the exact securities to the generated objects.

- The following examples show objects that may not be recognized by a tool and objects that may be recognized but cannot be or are not processed by a tool:

  Query/400 queries
  Recognized by tool but cannot be processed through the tool; these require manual update.

  QM queries Recognized by the tool but cannot be processed through the tool; these require manual update.

  QM SQL Recognized by the tool but cannot be processed through the tool; these require manual update.

  QM procedures The tool cannot recognize this as a valid form of source code.

  Job descriptions
  Recognized by the tool and can be changed using a CHGJOBD command, but the tool does not have this feature built into it.

  User spaces Not recognized by the system as valid form of source code.

  Data queues Recognized by the system as a valid object and copied to the target library with no attributes changed. The data queue attributes such as queue record length might differ and have to be done manually.

  REXX code Recognized by the system as valid source code, but the tool does not handle REXX code.

  Data areas Recognized by the tool as valid form of data with known format. The target data area was adjusted for length and date was converted to the target format.

  Data areas Recognized by the tool as valid form of data; cannot detect the format or the data fields. Hence, it just reports the object.

- Manual procedures including some kind of data transfers.
- Testing application as well as verifying procedures.

3.4 Tool Versatility

Customer applications are coded in different language environments. For example, a typical AS/400 application consists of a number of RPG or COBOL programs, a set of CL programs (that control those RPG and COBOL programs), some Query/400 queries, some SQL and QM queries, some DFU programs, a number of menus, physical files, logical files, display files, message files, job descriptions, a small number of CL Programs and CL Commands, and a small number of other types of objects. This implies that an enablement tool should be selected that can handle, if not all, most of these language environments. Other versatility criteria includes being able to use the various date representation techniques (as described before), as well as handling various date cosmetics.

Furthermore, you must see how much area of the Year 2000 enablement process the tool covers in terms of phases and how many additional tools are not required by using this tool.

The best way to judge a tool's versatility is to test it on a pilot application that has virtually all of the representative cases for processing options that the tool provider claims. This gives you a fairly good idea of a tool's capabilities.

3.5 Special Considerations for Impact Analysis and Modification Tools

The following sections contain common requirements that need to be looked for and assessed when performing an impact analysis or using an application modification tool. Note that this is not an exhaustive list of requirements.

3.5.1 Date Cosmetics

Date cosmetics refers to the transformation of the date format from the one stored in the database to the one that is shown on reports or displays:

```
Date format in database:  YYYYMMDD
Date format in reports:   MM/DD/YY
```

Program logic is required to accomplish this transformation and most of the tools existing today do not support this feature so the code must be entered manually. Note that in some cases, program code may have to be changed back to its original format (change the date format back to a two-digit year and then proceed). ADTS/400, ADS/400, and CODE/400 may be helpful in this process.

Once you have modified the displays and reports, you can identify the dependent programs by using ADS/400, and then change them to include the necessary code. See the following example:

```
                          Work with Fields

Dictionary . . . . . :   A960124C1
Position to  . . . . .                 Starting characters of field

Type options, press Enter.
  10=Work with programs referencing file                  14=Compile
  17=Display file description          18=Display object information .


Opt  Field       Record      File        Library     Attribute
10   AAR         OR070B05    OR070D01    IBMPGM2.3   DSPF
     AAR         OR070B06    OR070D01    IBMPGM2.3   DSPF
     AAR         OR070H01    OR070D01    IBMPGM2.3   DSPF
     AAR         OR170S04    OR170U01    IBMPGM2.3   PRTF
     AAR         SFRAR       SFRAF       IBMDAT2.3   PF-DTA
     AAR         SFRAR       SFRAL01     IBMDAT2.3   LF
     AAR         UTSKR       UTSKF       IBMDAT2.3   PF-DTA
                                                                    More...
Command
===>
F3=Exit            F4=Prompt           F5=Refresh         F6=Add fields
F9=Retrieve        F12=Cancel          F23=More options   F24=More keys
```

Figure 2. Searching for Dependent Objects for Field AAR Using ADM/400

On the preceding display, we have changed the field "AAR" in a display file and we want to know its impact on any programs. We can do that using the *Work with programs referencing file* option (option 10). Using this option produces the following display:

```
                    Work with Programs Referencing File

Dictionary . . . . . :   A960124C1         Attribute . . . . :   DSPF
File . . . . . :   OR070D01          Library . . . . . :   IBMPGM2.3
Text . . . . . :   Order Entry Source-code File

Type options, press Enter.
  2=Edit             4=Delete           14=Compile         15=Scan RPG source
  18=Display object information         25=Find string


Opt  Program     Library     Attribute   Use  Text
     OR070       IBMPGM2.3   RPG          03  Print Order Document









                                                                        Bottom
F3=Exit             F4=Prompt          F5=Refresh         F10=Command entry
F12=Cancel          F13=Repeat         F16=User options   F24=More keys
```

Figure 3. Searching for Programs Affected by the Change in a Display File

The preceding display shows the names of the impacted programs. You may now proceed either directly from here using the *Scan* option (option 15) or you may choose to edit the programs either from this display or from PDM.

3.5.2 Arithmetic Calculations Using Dates

Most of the two-digit year programs perform arithmetic calculations in two digits also. So at year 2000, operations such as year differences (to calculate age) can result in negative values. Here are some common date arithmetic errors:

Table lookup LOOKUP VAR1 ... results in a two-digit year

Age calculation 00 - 76 .... (= -76, an error)

Comparison If current year = 93, then ....

Calls with date literals
Call some pgm (″010195″, ″052896″, result)

Format change
```
C         DTMDY    MULT 10000.01     DTYMD
```

The Year 2000 enablement tool must check for these errors and if it cannot correct them, it should at least point out the occurrences.

3.5.3 External and Internal References Resolution

Most of the newer programs on the AS/400 system use some kind of field reference files or reference a common resource, either internal or external. The Year 2000 enablement tool must be able to handle the external as well as internal references. The following various types of external and internal references can exist in the programs:

- Fields referenced from some field reference file or database file:

```
A            CHOICE     R              REFFLD(ORDREC/ORDDT ORDSRC/ORDFREF)
```

- Fields referenced from within the source:

```
A            CHOICE1    R              REFFLD(CHOICE)
```

- Fields defined based on the definition of some other field:

```
C           *LIKE     DEFN ORHNBR    NBR
```

- Files included as include files and macros:

```
I/COPY ORDLIB/ORDSRC,DTSRV
```

- Redefining or aliasing of fields referenced:

```
ILEVER
I              ORDRAL                          LORDRA
```

- Externally described data structures:

```
IDATA1     EUDSLDADS
```

  These data structures are used in conjunction with an externally described data file. For example:

```
A                                      REF(IBMFRF)
A          R LDAR                      TEXT('Local data area')
A            WFKODE    R               REFFLD(FKODE)
A            WFIRMA    R               REFFLD(FIRMA)
             :
             :
A            WFELTE        74
```

- Parameters to called programs:

```
C                        :
C                    CALL 'BKORDR'                    BOOK ORDER
C                    PARM           ORDDT
C                        :
```

- Parameters from calling program that are used in some date calculations in the subject program:

```
C                        :
C           *ENTRY    PLIST
C                     PARM           CUSNO   5
C                     PARM           ORDNO   5
C                     PARM           RETDT   6 0
C                     PARM           RETCOD  1
C                                           :
C                     MOVE UDATE     RETDT   60
C                        :
```

- Data areas, data queues, communications, and so on.
- Fully-qualified object names and references:

```
C                    MOVEL'ORDLIB/' USRPGMN   13
C                    MOVE 'ORDPGM'  USRPGMN
C                    CALL USRPGMN
```

The Year 2000 enablement tool should identify these cases and ensure that the source references are translated to the destination (target) references. That is, the library names and file names are updated accordingly.

For example, for a source library of ORDSRC and target library of ORDTGT, the following source code is shown:

```
A           ORDDT  R     REFFLD(ORDREC/ORDDT ORDSRC/ORDFREF)
```

and should be changed to:

```
A           ORDDT  R     REFFLD(ORDREC/ORDDT ORDTGT/ORDFREF)
```

3.6 Other Considerations

You may not need to use a special Year 2000 enablement tool if your environment is small. In this case, ADS/400 might provide enough help to locate the files and programs to be modified and ADM/400 can be used for limited application inventory and application build. See Appendix B, “Application Dictionary Services (ADS/400)” on page 93 for further information on ADS/400 and the *Application Development Manager/400 User's Guide*, SC09-2133, for further information on ADM/400.

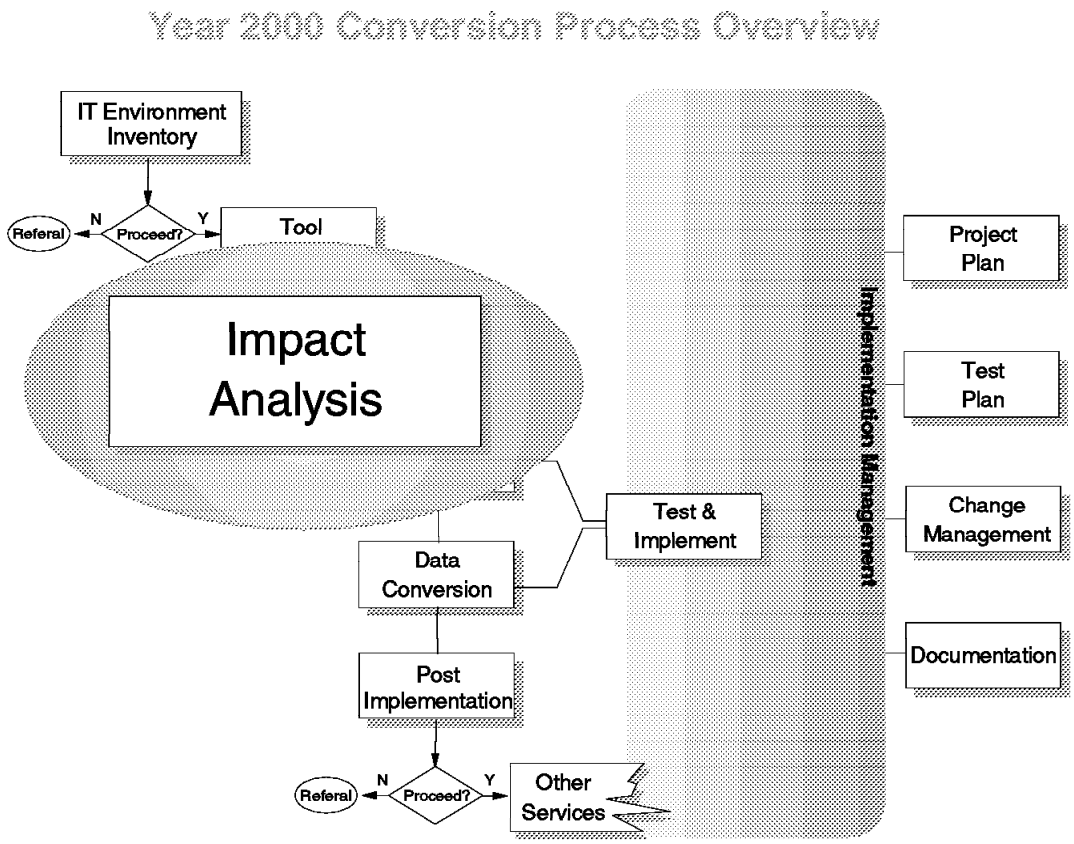
In a client-server environment, a sophisticated host-based tool might not be helpful. In this case, the following options are available:

- Source code may be processed by a client-based tool.
- Source code from a client may be moved to the AS/400 system if possible.
- The client part may be re-developed or modified using the visual development environments such as VisualAge and VisualGen if functionality is known.

3.7 Conclusions

You must be sure about what work you plan to do automatically and what work you want to do manually. You must take into consideration the customer requirements and special considerations. You must select the technique for date representation and plan for the application outlook. Then you can select the correct tools to do the job.

Chapter 4. Impact Analysis



4.1 Inputs and Outputs

The Impact Analysis phase uses the IT Environment Overview Report for input as well as additional information that is gathered using an Impact Analysis tool or manual process. The output of the Impact Analysis Phase is an Impact Analysis Report and a Detailed Proposal[3] for continuing with the subsequent modification, test, and post implementation phases. During the Impact Analysis stage, the IT Application Inventory is expanded to include information about specific changes that need to be made to date fields in programs and data. All of the date fields are identified either by using a tool or by manual inspection. Moreover, the detailed proposal encompasses the customer preferences and requirements.

The combination of the information in the IT Environment Overview Report and the Impact Analysis Report contains enough detail for you to size the conversion phases of the project including code and data conversion, testing, and any post implementation work. The Detailed Proposal assigns a monetary value to the

[3] Refer to Chapter 3, "Tool Selection" on page 13 for an outline of a proposal. The Impact Analysis phase only refines the scope of work as well as the charges.

services that you are performing. You need to use your business judgement in determining this value based on a past experience of sizing work associated with your services.

Important!

In principle, if the Impact Analysis and Modification tools are separate tools, you must add the risk factor for some of the work that may be left out because of the difference of the modification rules and capabilities.

4.2 Impact Analysis Report

An impact analysis tool produces a report of fields, variables, and so on that can be fixed automatically. It should also give you some sort of an indication of the remaining work that must be done manually. The information generated by the tool and the information that is gathered manually is combined to produce the Impact Analysis Report. The following example contains information that needs to be gathered to complete the Impact Analysis Report.

IMPACT ANALYSIS REPORT
Cash Flow System - Zen Oil Company

Impact Analysis Summary:

No. of lines of code:	25,000
No. of comment lines:	500
No. of date fields identified:	150
No. of date fields confirmed:	100
No. of date fields rejected:	50
No. of other fields:	10,000
No. of referenced fields in all categories:	5,000
Lines of code to be modified automatically:	5,000
Lines of code to be modified manually:	1,000
No. of object in the application:	8,000
RPG Programs:	5,000
Display Files:	1,000
Report Files:	50
Query/400 Queries:	100
SQL Packages:	10
QM Queries:	10
Menus:	20
CL Commands:	60
CL Programs:	500
:	
:	
No. of object that cannot be processed by tool:	800
Query/400 Queries:	100
SQL Packages:	10
QM Queries:	10
Menus:	20
CL Commands:	60
:	
:	
Current database size:	50M
Database size after enablement:	53M

List of objects processed:	Status & Attributes (Confirmed, Potential or Unknown)
:	
:	

List of fields selected for date processing:	Attributes
:	
:	

List of fields rejected for date processing:	Attributes
:	
:	

List of all the fields:	Attributes
:	
:	

List of code-segments, bridges
or date routines to be written
:
:

Documents, Procedures & Pre-printed stationary to update:
:
:

Miscellaneous:
:

4.3 Impact Analysis Process

The Impact Analysis process can be viewed as a set of certain higher level activities and processes. These activities may be a part of the main stream Year 2000 enablement process directly or may just augment it. The following sections contain a description of these activities.

Note: Make sure that you add the effort required as well as the tool costs and training requirements for whatever applies to your customer's application.

4.3.1 Review of Preferences and Practices

Almost every tool comes with a set of built-in or predefined date field selection criteria as well as a set of standard rules. The usual practice is to include an abbreviated field or field description for the date. Some commonly used examples include:

```
AS-OF       ASOF        BEGIN       BGN
BGNDT       START       ST          END
ENDDT       ED          TERM        DTE
DAT         DAY         DD          DOB
DOH         EXP         JULIAN      JUL
MONTH       MON         MO          TIME
TIME-STAMP  TMSTMP      THISDATE    TOD
WEEK        WEEKDAY     YEAR        YR
YY          YMD         YYMMDD      MDY
MMDDYY      DMY         DDMMYY      CCYY
CYMD        and many more ...
```

Each company may have its own coding practices, naming conventions, and set of preferences. Understanding these practices is the key to searching for date fields whether it is done manually or using a tool. Be sure to review the following:

- Date variables names:

  These can vary with geographical locations. Look especially for the non-English letters that are used.
- Date constants and date literals
- Program described table data
- Date calculations and check for non-standard date calculations, especially conversions of a date from date text to date numerics. For example:

  `Conversion of '4 July 1996' to '1996.07.04' through some coding and parsing.`
- Constant definitions, internally described files, reports, and redefinitions
- Languages, utilities, and other objects being used
- Program and data specifications
- Application design specifications
- Quality assurance and standards requirements
- Checking, editing, and cosmetics of date fields
- Structural analysis of the coding techniques as unstructured, procedural, or OO
- Data sharing, communication, and batch update of on-line or off-line remote or local systems as well as applications
- Inter-relations with other vendors' software and hardware (for example, the credit card verification machines and the ATMs)
- Performance preferences (a customer does not want to lose the current performance after the Year 2000 enablement).

- DASD usage preferences (a customer does not want to increase the DASD that is presently used).
- Preferences regarding security of the application regarding some off-shore application modification and so on
- Level of involvement of customer's DP or IS personnel in the enablement exercise

You may want to analyze the application structure and data flow using structure analysis tools in order to suggest some improvements to the application but this may go beyond the scope of most Year 2000 conversions.

As an additional step, you should make a dependency listing of the application objects. This list helps in understanding the application as well as helping in application generation/build.

4.3.2 Review Company Procedures

Reviewing company procedures and workflow is necessary before proceeding with Impact Analysis. Most of this information should have been collected during the IT Environment Inventory phase, but additional details on forms and displays need to be collected at this time. Some of the items include:

- Review manual procedures and identification of required updates.
- Review preprinted stationary as well as the soft-copies that have a fixed year field (for example, a template or overlay cover sheet in which the year is fixed as "19").
- Identifying the organizational policies regarding date fields.
- If applications are converted using a phased approach, you have to allow for the work required to write bridge programs.
- Whether or not the ongoing changes on the application can be frozen while the Year 2000 enablement process is running.

4.3.3 Tool Setup

Having reviewed the coding practices, the next step is to install the tool in its execution environment and customize it for the specific customer application. The following steps are required to configure the tool:

- Set up the tool environment including hardware, software, communications, and space requirements. This may involve a temporary upgrade of some hardware or software.
- Install the tool on the designated hardware.
- Set up the tool's preferences.
- Set up the application including:
  - Source and target libraries, files, and members
  - Source analysis options
  - Source modification options
  - Application build options
  - Date representation selection
  - Date cosmetics options
  - Modification logging options for source, objects, and data
  - Reporting options
- Determine field selection, rejection, and modification criteria, including:

- A list or pattern of fields to be accepted along with their attributes of data types and lengths.
- A list or pattern of fields to be rejected along with their attributes.
- Rules for the selection of fields derived from the ones accepted directly through patterns. These fields get their data from some calculation on the selected date fields. Note that the identification of these types of fields may take a number of passes through the source code for some of the tools.
- Rules for modifying the source code to the target code.

4.3.4 Customer Involvement

An Impact Analysis tool selects and rejects fields as date fields. Once this is complete, you have to review the date fields to confirm that the correct fields were selected. You may need to modify your search criteria and run the tool several times to produce a more accurate listing. A person who knows the naming conventions used in the applications (for example, the person who originally wrote the code) can make this work go much faster since this person knows what to look for. This is a good time to involve the customer in the analysis and the modification part if the customer is willing and able to assist.

Customer involvement is necessary in the application testing phase because without this involvement, you may not be able to conduct a comprehensive application test including policies and procedures.

4.3.5 Manual Work

There are several manual tasks that you should be aware of in estimating the time it takes to do the actual code conversion. The following work should be factored into the Impact Analysis Report:

- Use of sentinels:

```
IF CARD_EXPIRATION_YEAR = 99 THEN
    GENERATE_ERROR('CARD HAS EXPIRED')
END
```

  A tool may try to fix the numeric year constant to something such as "1999".

- Field editing and cosmetics:

```
A          R RECXYZ
A            JOINYR          2SOB  2 10COMP(GE 85)
A* Checking that the joining year is greater than or equal
A* to the year, the company was formed.
A                                    EDTWRD('__/__/__')
A* Edit word for the display field with two digit year.
A                                    EDTMSK('__/__/__')
A            BIRTHYR         2SOB  3 10EDTCDE(Y)
```

  The preceding data may be replaced with the following data:

```
A          R RECXYZ
A            JOINYR          4SOB  2 10COMP(GE 1985)
A* Checking that the joining year is greater than or equal
A* to the year, the company was formed.
A                                    EDTWRD('__/__/____')
A* Edit word for the display field with four digit year.
A* This editing change may not be required if the user
A* wants to display year in two digits in the target
```

```
A* application.
A                                      EDTMSK('__/__/____')
A            BIRTHYR         4SOB  3 1OEDTWRD('__/__/____')
A* Edit code 'Y' is not supported for eight digit numeric
A* fields
```

- Wrong detection of date field format or the field itself from description:

  Field named **"DUEDT"** with a format of **"YYMMDD"** and
  attributes referenced from another field with **"MMDDYY"**
  in the field description.

  This causes the field **"DUEDT"** to be detected as a
  "YYMMDD" format field, which is wrong.

- Wrong detection of date field format based on data stored in database files:

  This usually occurs when the tool scans the database file for the data stored for some field and tries to determine the field format.

- DDS record format, field overflow, or overlap:

```
A          R LC37OBOO
A            ORDDT           8SOO  2 1ODSPATR(HI)
A* The ORDDT field has overlapped with the CUSTNO field after
A* conversion.
A            CUSTNO         12SOO  2 17DSPATR(HI)
```

- Where a cluttered display has been broken up into two displays and code has been written for handling the split display.

- Error in date field operations:

  – Array/table lookup with either date field as argument or the date field as result.

  – Date differences (for example, age calculation in two digits).

    CURRENT AGE = CURRENT YEAR - BIRTH YEAR

    Where "1965" is the birth year and "2001" is the current year,
    the result is "-64". The matter becomes even more complicated
    for fields that are not declared as signed numeric fields.

  – Date-related logical errors, such as the incorrect calculation of a leap-year.

- Build options:

  CRTDSPF ... RSTDSP(*YES) -- For **"Display Files"** that
  require the Operating System to do the restore display once the
  control is transferred back to these displays.

  CRTDTAF ... SIZE(1 0 0) ALLOCATE(*NO) -- For
  "Field Reference Files" that do not have any data in them.

  CRTDTAF ... MAXMBRS(5) -- for **"Data Physical Files"**
  that have multiple members in them.

- Text date to numeric date conversion:

  Conversion of '4 July 1996' to '19960704'.

- Interaction with OfficeVision/400 APIs and Hierarchical File System:

- Other objects:

```
Data Area              ICF Files              JOB Descriptions
CPPs                   CL Commands            Personalized Tools
Data Dictionaries      Overlays               Communication Descriptions
Subsystem DescriptionsDFU definitions         SORT Definitions
Documents              Folders                Other utility programs
Hierarchical File      and others missed by the tool-set....
Systems
```

- Code Standardization:

  Add code that may be required to use common date routines.

- Case tools:

  Case tools usually require that the application generated using them is modified and regenerated through them.

- Non-standard development environments:

  Some customers as well as the solution providers may create their specialized environment for easier formatting and generating reports for some specialized multi-purpose application. This is achieved using a standard language program (may be written in standard language such as C) and running that program on a text file member (that contains the report generation and formatting options) to produce the desired report.

- Security Implementation:

  This may require manually checking the securities of the source objects and copying them to the target objects.

- Data conversion for some specialized case.
- Data entry of rejected records or the records in error.
- Documentation and help text update.
- Conducting application test including unit tests.
- Verification of organizational procedures and policies.
- Creating a test manual.

Another area not covered by most tools is updating help text and documentation. The amount of work required here can vary greatly from having to produce all new documentation to just listing the changes and having the customer make the actual updates.

If the year notation on screen layouts and reports are to be expanded, the work can be easily accomplished using the CODE/400 tool.

Important

Some of the tools may use database files date data as well as the field descriptions and field names to detect the date field formats. So be careful while browsing through the list of variables and their attributes in the Impact Analysis report.

4.3.5.1 Manual Procedures

Changes to data entry procedures and using preprinted stationary cannot be automated. New forms must be designed or previous ones modified, and manual procedures must be updated to reflect the changes. For example, order entry instructions need to be provided to enter the year in four digits and in the changed format. Also, the old format must still be interpreted correctly so instructions might be changed to say, ″Orders dated before such date are to be read like this″.

4.3.5.2 Partitioning

Partitioning refers to breaking down the application into autonomous segments. This may be a requirement due to some kind of a staged conversion or due to the limitation of some external or internal object or device′s limitation.

The amount of partitioning that is required has a direct impact on the amount of work to be put in and, hence, the cost, especially the modification and the testing phases. For large and complicated application software, it is necessary to chalk out the partitions and include the effect in the Impact Analysis report. The testing phase is the most affected one because the test must be conducted once for each partition and then on the application as a whole (after all of the partitions have been converted). Needless to say, you must not forget the planning aspect of the partitioning approach and the extra effort of writing the bridge programs, if needed.

4.3.5.3 Parallel Improvements

Since the customer is going to modify the application anyway, it is a good idea to suggest making some improvements to the application. These may include:

- Structuring code by introducing common service routines.
- Standardizing date routines.
- Moving from an OPM to ILE application generation model.
- Moving to an AS/400 native environment from an AS/400 S/36 environment.
- Moving to an object-based design instead of a procedural one.
- Moving to a GUI-based, client/server application.
- Changing the manual procedures and forms to some new standard based on the methodology adopted.
- Creating a test document or a test plan out of the testing phase.

A customer might have planned to make these types of changes, but just did not get around to them in the past. Since they are making changes based on the Year 2000, some customers may opt to enhance their applications in other ways also. For example, they may have PCs in their organization but are running them as terminals to an AS/400 system. A good GUI-based client/server application may help in reducing the load on the AS/400 CPU as well as revitalizing the user interface.

A side benefit of the tools that are used in the process of Year 2000 enablement is that they can be used for other purposes as well. For example, if you want to move some calculations out of the source code into a separate module for flexibility and maintainability, you can define a rule such as:

```
REPLACE

      'CURRENT_AGE=CURRENT_YEAR MINUS BIRTH_YEAR'

WITH

      'CALL "AGE_DIFFERENCE" PARAMETERS: BIRTH_YEAR'
```

While this is a simple example, you may find this type of feature in the application modification tools very powerful.

If you plan to move to the ILE RPG environment from an OPM RPG III or RPG/400 code, you may consider looking into the CVTRPGSRC command. This command converts the OPM source to ILE RPG source. Once you have completed the conversion, you may extract common code segments from various programs and put them in a *Service Program*. For details, see *ILE Concepts*, SC41-3606, and *ILE Application Development Example*, SC41-3602.

Some of the GUI based tools, such as GUI/400, Visual RPG, CODE/400, and VisualAge C++ can be used for this type of application modernization.

Make sure that you put in some estimate of the work required to make changes such as those previously listed.

4.3.5.4 Additional Hardware and Software Requirements for Customer

The impact analysis report should include information to help in evaluating additional disk space requirements for the customer by calculating the increase in the size of current database. Still, the requirements for the Year 2000 application modification and build and the testing phases must be gathered manually.

4.3.5.5 Application Update Presentation for Users

The service provider should arrange for some kind of an update session or a presentation to the customer's operational and development staff regarding the changes in the preprinted stationary, procedures, data-entry forms, reports, and database structure. This helps them in maintaining their application as well as maintaining the Information System (IS) effectively.

4.3.5.6 Testing Requirements

You need to understand the customer's testing requirements, methodology, and existing test environment. A large part of the resources required to enable an application for the Year 2000 is devoted to testing. If the customer has a comprehensive, well-organized test plan, your job is much easier. Be sure to understand this part of the customer's environment well, and weigh it heavily in formulating the detailed proposal.

Be sure to ask about month-end, quarter-end, and year-end as well as Backup and Recovery procedures to ensure they work accurately.

Using specialized program patches may make the testing easier as shown in the following example:

```
C                   MOVE *YEAR    YRFLD
```

replaced by:

```
C*---TSTSTB(B)
C*                    MOVE *YEAR    YRFLD
C                     CALL 'GTCYR'
C                     PARM YRFLD
C*---TSTSTB(E)
```

This patch code remains there until the testing is complete. The code is a little slower in execution but the GTCYR (get current year) program can be set up to get the year field from some testing data area. This data area may be changed to carry out the speed testing as needed without affecting the other system. Once the testing is done, the testing tool may be used again to replace the patch code with the actual code.

Some of the other testing tool elements include:

- Key strokes recorder/player utilities
- False data generation tools
- Data modification tools for manual generation of data
- Screen and report comparison tools

The actual testing process may include the following high-level activities:

- Unit testing the application
- Complete application testing
- Performance and preferences testing
- Validation of policies and procedures

The type of test as well as the methodology has a direct impact on the cost as well as the system requirements.

Understanding the complexity of the testing procedures and the testing techniques available (as given in Chapter 7, “Test and Implement” on page 69), you must add the effort required and involvement of the customer to the statement of work.

4.3.5.7 Documentation

Documentation is vital to every organization because it is the way that new personnel can understand the application that was built by their predecessors. Moreover, these documents reflect the company’s policies and procedures. Organizations usually have the following types of documentation:

- Users guides for application
- Reference manuals for application
- Updated procedures and policies
- Test methodology
- Pre-printed stationary

  and others ...

4.3.6 Performance Impact

The Service Provider should be well aware of the performance of any code that is added such as:

- Date processing bridge routines
- Miscellaneous date routines for accessing and converting dates
- Date cosmetics routines
- Code to call common service routines or bridges

Performance analysis can be done on a pilot sample of the customer application or it can be done using some pilot application of your own. The customer might want to do away with the bridges and cosmetic routines in the final product.

The following example shows a performance threat:

```
C           YMD       MULT 100.00001 MDY
```

The compiled code, when executed, carries out a floating point operation that consumes far more processor cycles than an integer or a move operation. Moreover, the compiler puts in the code a statement to convert factor 1 to a floating point field prior to carrying out the calculation. It also generates code to change the result back to integer form to place in the result field.

In this particular case, the execution generated an overflow and an underflow as well. During these conditions, the flow of execution is disrupted and control is transferred to the condition handling routines. Each such disruption causes some idle processing cycles to branch off and then resume the operation. Imagine doing this for processing one million records! This condition can be corrected by using the following code:

```
I           IDS
I                                       1   20YY1
I                                       3   40MM1
I                                       5   60DD1
I                                       1   60YMD

I           IDS
I                                       1   20MM2
I                                       3   40DD2
I                                       5   60YY2
I                                       1   60MDY

:
:

C                     MOVE YY1       YY2
C                     MOVE MM1       MM2
C                     MOVE DD1       DD2
 * After execution of the preceding code, the date stored in "YMD"
 * is stored in "MDY" with format conversion.
 * There are no overflows or underflows in this particular piece
 * of code, but it does requires a number of extra lines of RPG
 * code.
```

Apart from that, the customer might want to test the impact on the data entry if the application is highly batch oriented.

4.4 Manual Impact Analysis

In rare cases, it might be appropriate to do impact analysis manually. A rough estimate may be achieved by using ADS/400 and manual exploration.

- If you have a field reference file, review it to identify the fields impacted by the year 2000 enablement process.
- Create an application dictionary for your library using ADS/400. Note that the library to create a dictionary on must be a compiled object library.

- Review the fields extracted by the ADS/400 and select the ones that are date fields and are impacted by the Year 2000 change.
- For each of the fields, list the objects that require modification if these changes were enforced.
- Scan each of the program source members for these date fields one-by-one.
- Scan each of the program source members for suspected date fields.
- From the list of the scanned program source members, collect other variables that are dependent upon the date variables.
- Scan for the variables selected in the previous step within the program source code.
- Get the list of all other kinds of objects in the application, such as data areas, queries, job descriptions, data queues, and so on.
- For each of these miscellaneous objects, list the dependent programs.
- Collect this data and summarize it.
- Add in the estimate of the building tasks, testing tasks, and consider all of the elements described before within this chapter.
- Add in the estimate of scope of work the manual work required based on your past experience.

The summary should give you an estimate of how much work is to be done (in terms of lines of code, fields, and objects to process). Add to it the risk factor and implementation charges to reach an approximate figure of amount of work to be done.

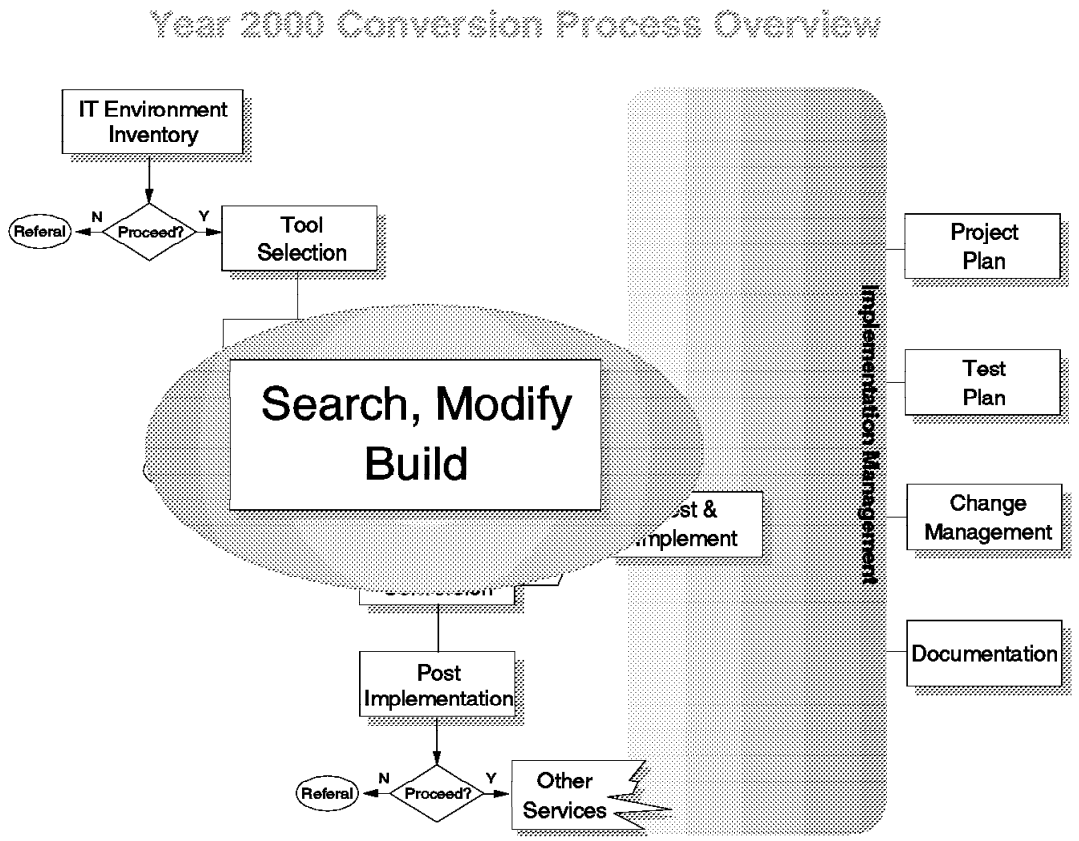
Note: A typical application consists mainly of programs, reports, display files, and physical and logical files, so you should base your estimate on these objects with some cushion for other objects.

For details regarding usage of ADS/400, please refer to the *AS/400 Application Development ToolSet/400: Application Dictionary Services/400 User's Guide*, SC09-1860.

4.5 Manual Application Build

Applications can be built manually with ADM/400. For details regarding usage of ADM/400, refer to *AS/400 Application Development ToolSet/400: Application Development Manager/400 User's Guide*, SC09-2133.

Chapter 5. Search, Modify, and Build



5.1 Inputs and Outputs

The search, modify, and build phase uses the Impact Analysis report that identifies the date fields and variables that need to be changed to be Year 2000 compliant as input. The result of this phase is a converted application environment. It is assumed that the customer has the source code for all of the applications, and the source is at the correct level.

5.2 Partitioning

You probably cannot change the customer's entire environment in one step, so you need to decide how to partition the work into logical steps. During the conversion process, parts of the application use Year 2000 compliant dates, and other parts (those not yet converted) use dates in the old format. As an interim solution, you must build bridge programs to allow these programs to interoperate.

5.3 Conversion Techniques

The following discussion contains conversion techniques you may employ:

5.3.1 Solution #1: Conversion to Four-Digit Year Format Using Eight-Digit Date Field

This option uses a four-digit solution to change the data type of the date field from P(packed decimal), S(zoned decimal), or A(character) to L(date).

This approach requires changes to both the data and the program by **converting all references and uses** of the two-digit year format (YY) to a ″date″ data type format. The default format for an SAA ″date″ data type field is *ISO, which is YYYY-MM-DD. You can change this format using keyword DATFMT either in DDS or in RPG into *USA (=MM/DD/YYYY), *EUR (=DD.MM.YYYY), or *JIS (=YYYY-MM-DD). You should avoid using DATFMT(*MDY, *DMY, *YMD) as it presents dates in a two-digit year format.

An SAA ″date″ data type field can be used in both OPM and ILE environments. The real benefit of using this format comes with ILE. For example, ILE RPG/400 provides new commands such as ADDDUR, SUBDUR, TEST, and EXTRCT. With ADDDUR/SUBDUR, you can add or subtract a duration to and from a date. You can extract part of a date with EXTRCT and test for a valid date with TEST without creating a user calendar or complicated date arithmetic, such as leap year calculations.

This method also requires that you convert all software programs that reference or use the updated data simultaneously, or use a bridging mechanism to perform the conversion between old and new data and programs. (You can accomplish this program and data conversion in steps.) Otherwise, you immediately encounter data integrity problems caused by the inconsistency of date/time data formats.

Pros and Cons

Pros

1. This solution can provide a four-digit year format. It is considered to be a complete, permanent, and obvious solution.
2. It employs the ISO format (YYYY-MM-DD) by default.
3. It provides increased security against potential inappropriate decisions today if you do not selectively ignore ″cosmetic-only″ situations.
4. It can ease your migration if you selectively ignore ″cosmetic-only″ situations.

Cons

1. You need to convert the year data from a two-digit format to a four-digit format in all cases.
2. It requires you to relocate adjacent fields in the date field layout, and usually requires you to increase record lengths.
3. An inherent future risk in initial assessment that determined a particularly situation can be ignored as ″cosmetic-only″.

4. Increased DASD space usage is required due to the data field expansion of data (consider including not only active but also archive data) and duplicate DASD space during conversion.
5. You might experience a performance impact due to increased time in processing and date access.

5.3.2 Solution #2: Conversion to Full Four-Digit Year Format

This option is a four-digit solution that externalizes a four-digit year format.

This approach requires changes to both the data and the programs by **converting all references and uses** of a two-digit year format (YY) to a four-digit year format (YYYY). It also requires that you convert all software programs that reference or use the updated data simultaneously, or use a bridging mechanism to perform the conversion between old and new data and programs. (You can accomplish this program and data conversion in steps.) Otherwise, you immediately encounter data integrity problems caused by the inconsistency of date/time data formats.

To ease your migration, you might consider ignoring any non-impact (cosmetic) data fields in the YY format. A cosmetic date is one that, if externalized, is only interpreted by humans. Such occurrences might include the date on an output separator page or a display-only date on a display in a panel-driven application.

Note: Be careful when selecting those situations that you decide to ignore and call cosmetic only. Be certain that they do not cause any data integrity exposures or ambiguity or are not accessed by any other program. Such instances of non-problem YY formats appear in a report header that shows the printing date of the report. The date is meant for human understanding only, not computer program manipulation. Consider the potential for future change. For example:

- Today′s reports might be written to a data set tomorrow.
- Display-only dates today may prove useful as a collating value when archiving that output tomorrow to meet a new business or government standard.
- Even when viewed by a human, two-digit dates can prove ambiguous if the data spans 100 years.

If you allow the end user to continue to input two-digit dates for compatibility and ease of data entry, the responsibility to translate that data into a full four-digit date falls to you, the application or systems programmer. One possible solution is to apply a context-sensitive prompt to allow the user to select a century indicator. For example, allow all dates to be entered as two-digit dates and automatically prefix those with the current century unless the date is a future date or historical date. What constitutes “future” or “historical” is your decision but can be any date other than today′s current day, week, month, year, and so on. Using this scheme, a future date in context of a loan maturity date can be set to 20yy, or a historical date automatically forces the user to select a century from a ″choose a century″ (...16, 17, 18) prompt list.

Pros and Cons

Pros

1. This solution can provide a four-digit year format. It is considered to be another complete, permanent, and obvious solution.

2. It provides increased security against potential inappropriate decisions today if you do not selectively ignore ″cosmetic-only″ situations.
3. It can ease your migration if you selectively ignore ″cosmetic-only″ situations.

Cons

1. You need to convert the year data from a two-digit format to a four-digit format in all cases.
2. It requires you to relocate adjacent fields in the date field layout, and usually requires you to increase record lengths.
3. An inherent future risk in initial assessment that determined a particularly situation can be ignored as ″cosmetic-only″.
4. Increased DASD space usage is required due to the data field expansion of data (consider including not only active but also archive data) and duplicate DASD space during conversion.
5. You might experience a performance impact due to increased time in processing and date access.
6. Some programming languages allow integer dates that are offset from a base date to be stored in files, databases, or passed as parameters between programs. Such integer dates provided by COBOL intrinsic functions, Language Environment callable services, the CICS FORMATTIME command DAYCOUNT option, and other similar functions must conform to the standard YYYYMMDD format. This standard eliminates potential ambiguous data and errors due to each integer-date system using a unique starting date. Therefore, the potential for mixing incompatible integer dates when passed outside of a single source module is extremely high and must be avoided.

5.3.3 Solution #3: One-Digit Century Code

This is basically a two-digit year solution with an additional one digit to define the century. There are two possible ways to use this method. First, add a separate one-digit century code field to an existing six-digit date (″external century code″). Second, expand the date field to seven digits by adding a one-digit century code in front of the date (″embedded century code″).

5.3.3.1 External Century Code

This solution basically uses an existing two-digit year date and adds a separate one-digit century code to identify the correct date and year. For example, OS/400 V3R2 provides a new system value called QCENTURY. The format of QCENTURY is CHAR(1) and the system value supports the values ″0″ for 19YY, and ″1″ for 20YY so that you can retrieve this value and make use of it in your application.

This approach requires changes to both the data and the programs. For the database, the change is fairly simple: add a one-digit field to the end of the current record format (for each date field in the record) so that you do not need to recompile the related programs if they are not using the date field. If the date field is used as a key in an access path, you must also change the access path to include the new century code field in the key fields section so that the correct collating sequence is achieved.

For the programs, the change is not that simple. You should closely examine your program for the date arithmetics. For example, in RPG, you may have to use the data structure to combine the six-digit date field and the century code to obtain the correct result from date duration calculation. Here again, you might consider ignoring any ″cosmetic″ date fields in the YY-format, especially in display and printer files. Refer again to Solution #1 for the discussion about this cosmetic data.

The biggest benefit of this conversion method comes with its transition process, that is, to enable you to make a two-step conversion. You can convert the DB format first adding the century code at the end, and then copy the current data into the new format with the *MAP option. When you finish adding the appropriate century code data, you can start modifying your programs later. As the base data format including date fields remains the same, the transition does not have to take place at one time.

Pros and Cons

Pros

1. You are able to convert the database format first and modify the programs later.
2. This solution provides an easy solution for data migration into the new format; just use CPYF with the *MAP option, and then add Century Code data later.
3. You can ease your migration if you selectively ignore ″cosmetic-only″ situations.

Cons

1. Additional logic is required for all programs that perform date calculations, such as the use of data structures to combine the date fields and the century code fields.
2. The collating sequence is not correct if the date field is used as a key unless the Century Code is added to the key.
3. It requires you to increase record lengths.
4. An inherent future risk in initial assessment that determined a particularly situation can be ignored as ″cosmetic-only″.
5. Increased DASD space usage is required due to the addition of the new Century Code field.

5.3.3.2 Embedded Century Code

This solution is meaningful only when you are using a date format of YMD in a six-digit, packed decimal field. You can expand your six-digit date field into a seven-digit packed decimal, and add century code data as the first digit of the field. Because of the nature of a packed decimal, both six digits and seven digits are stored in the same length of four characters in the physical file record. Therefore, the real benefit of this solution is that it does not increase DASD space usage.

This approach requires changes to both the data and the programs by converting all references and uses of the two-digit year format (YY) to a three-digit year format (CYY). Since it changes the date format itself, the conversion steps are not as simple as it is in the External Century Code method. It also requires that you convert all software programs that reference or use the

updated data simultaneously, or use a bridging mechanism to perform the conversion between old and new data and programs. The same considerations apply here for this bridging mechanism and ″cosmetic″ data handling as discussed in Solution #1.

Pros and Cons

Pros

1. The correct data sequencing is retained.
2. No extra disk space increase is required.

Cons

1. You need to convert the year data from a two-digit format to a three-digit format in all cases.
2. It requires the data conversion from YY-format to CYY-format.

5.3.4 Solution #4: Windowing Techniques

This is a two-digit solution that externalizes either two-digit or four-digit year formats. This approach requires changes to your programs only; no data changes are required.

Important!

These approaches can be applied only to dates within a maximum 100-year period at any one time. This solution is considered temporary because there is no guarantee that in the future, your applications will not expand to process dates that are more than 100 years apart. Therefore, this approach always carries with it a potential future exposure. (For example, humans are living longer. Therefore, databases that include birthdays (medical, civil, insurance, and so on) and the applications that access that data are already at risk with many dates spanning 100+ years.)

Two types of **windowing** techniques have been defined: the fixed window technique and the sliding (rolling) window technique.

5.3.4.1 Fixed Window Technique

The **fixed window** technique uses a static 100-year interval that generally crosses a century boundary. This technique determines the century of a two-digit year by comparing the two-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to a specific year within the 100-year interval.

Consider this specific example: if the years of date-related data of your application fall in the range of January 1, 1960 to December 31, 2059, you can use a two-digit year to distinguish dates prior to the year 2000 from the year 2000 and beyond. If using the current system year of 1995, the number of years in the past and future are specified as 35 and 64, respectively. Program logic determines the century based on the following data checking. If the two-digit year representation of a specific year is xy then if:

- $xy \geq 60$, then it is a 20th century date (19xy).
- Otherwise (that is, $xy \leq 59$), it is a 21st century date (20xy).

If, for example, you need to maintain a window of 35 past years and 64 future years such that next year, 1996, your application can successfully deal with dates in the range 1961 through 2060, you need to adjust this program checking every year. The inherent future risk when employing this technique is obvious, and when compared to the sliding window technique, is far less desirable.

Pros and Cons

Pros

1. There is no need to expand the two-digit year data to a four-digit format.
2. It can provide a four-digit year format for data reference.
3. It can distinguish years from different centuries using only a two-digit year format (provided the years being processed are in the range of 100 years at any one time).
4. It can be useful if the particular program is being phased out and a temporary solution is appropriate.

Cons

1. Potential exposures exist when or if the function of the software application needs to process years beyond the range of 100 years.
2. Expect a performance impact in direct proportion to the quantity of date processing the particular application handles due to the overhead of a two-digit to a four-digit year conversion.
3. All programs that use the fixed window technique may need to be manually updated on a yearly basis depending on how your date routine is packaged.
4. All programs that accept output from the fixed window technique must use the same assumptions (current date, past, and future windows).
5. Retaining a two-digit year representation does not provide collating sequence support nor does the use of a fixed window technique provide indexing sequence support when two-digit years are used as index keys in indexed files. You need to provide additional processing to obtain correct collating and indexing sequence output.

5.3.4.2 Sliding Window Technique

The **sliding window** technique uses a self-advancing 100-year interval that generally crosses a century boundary. This technique determines the century of a two-digit year by comparing the two-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to the system year (generally the current year) that the system sets and maintains. Your applications can access the date that the system sets and automatically advances. This is the main advantage of using a sliding window over the fixed window (where the window is immovable without manually revising the programs each year).

As appropriate to your application environment, you can maintain more than one window. For example, you can set one window to process historical dates, one for mortgage dates, one for birth dates, and so on; and the program adjusts the system date and past and future windows to meet the specific application′s needs.

Consider this specific example. If the dates in your application fall into a range of 35 years in the past and 64 years into the future based on the current year, 1995, your program can accept and accurately deal with dates of 1960 through 2059. Next year, 1996, the window advances and your application accurately deals with dates of 1961 through 2060.

Graphically, Figure 4 on page 52 illustrates this example using the current (1995) 100-year window and that same window when the current system date has progressed to the year 2024.

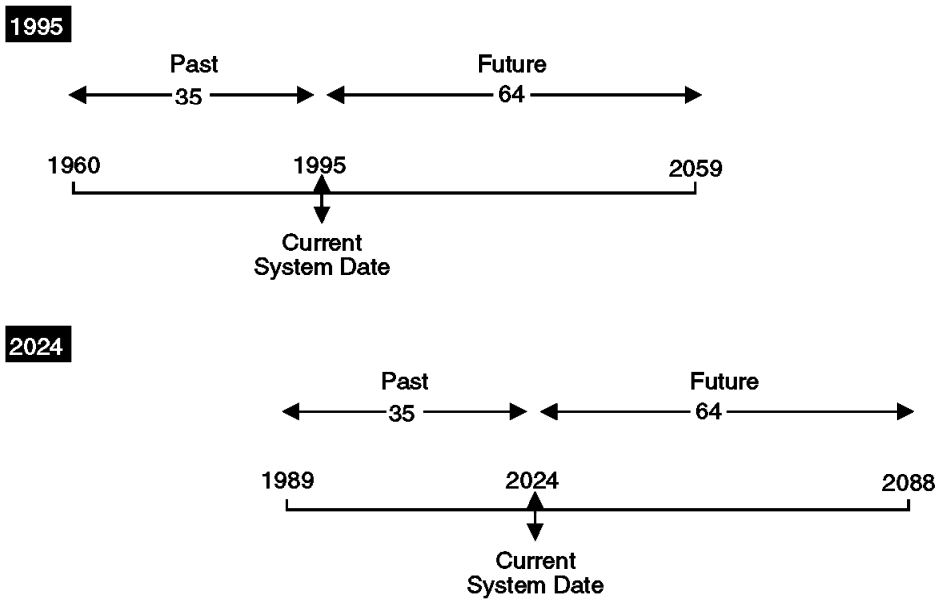


Figure 4. Graphical Representation of the Sliding Window Technique. (This figure uses two current system dates, 1995 and 2024, as an example.)

A sliding window approach requires programming logic to interpret the meaning of all two-digit year data. Such additional programming logic can be packaged into a common data/time service routine, callable from a two-digit year data exploiter. This reduces the programming overhead and impact to the calling programs. The IBM product, Language Environment, provides common date/time service routines with sliding window features. By default, Language Environment uses a window of 80 years in the past and 20 years into the future that automatically adjusts based on the current year date.

Pros and Cons

Pros

1. There is no need to expand the two-digit year format to a four-digit format.
2. It can provide a four-digit year format for data reference.
3. It can distinguish years from different centuries using only a two-digit year format (provided the years being processed are in the range of 100 years at any one time).
4. There is no need to convert the date data to a new date representation scheme.

Cons

1. Potential exposures exist when or if the function of the software application needs to process years beyond the range of 100 years.
2. There is a potential performance impact in direct proportion to the quantity of date processing the particular application handles.
3. All programs that accept output from the sliding window technique must use the same assumptions (current date, past, and future windows).
4. Retaining a two-digit year representation does not provide collating sequence support nor does the use of a sliding window technique provide indexing sequence support when two-digit years are used as index keys in indexed files. You need to provide additional processing to obtain correct collating and indexing sequence output.

5.4 Considerations When Selecting Solutions

The potential two-digit-year exposures can be classified into two categories (no impact and impact). When selecting an appropriate solution (or solutions) for the impact categories, be sure to consider not only the applicability of the solution (or solutions) on the module itself but also the potential impact and adjustments on the external modules that receive data from this module. You have three basic choices; you can:

- Change your application.
- Change your application and the data.
- Invest in a new application (which can also require some date data changes).

Certainly, most IS organizations build their Year 2000-ready system on a combination of these choices. When more than one solution appears feasible, weigh its appropriateness based on:

- Time available
- Resources available (personnel and hardware)
- Project cost (individual application conversions and overall)

As today′s IS environment becomes increasingly more complex and sophisticated, the instances of program and data isolation decreases. Networking, open, and distributed computing allow data to flow from site-to-site, system program-to-application program (or application program-to-system program), and so on. You must ensure that these layers of software ″speak the same language″. As you add in-house code, solution provider-written code, and migrate your operating system, be sure to review that software for date format compatibility.

5.4.1 Solution Applicability

Different combinations of solutions are applicable to different situations. Evaluate solutions based on a ″best-solution combination″ basis when considering both a module itself and other related modules. For example, when applying:

- **Solution #1** (″date″ data type) or **Solution #2** (full four-digit solution) to a certain module, another module that receives data from this module can receive:

  - Two-digit year data as before, provided there is no exposure for itself.
  - Two-digit year data as before and apply Solution #4 (windowing techniques) for its own exposures.
  - Four-digit year data and apply Solution #1 (″date″ data type) or Solution #2 (full four-digit solution) or Solution #3 (one-digit century code) for its own exposure removal.

- **Solution #3** (one-digit century code) to a certain module, another module that receives data from this module can receive:

  - Two-digit year data as before, provided there is no exposure for itself.
  - Two-digit year data as before and apply Solution #4 (windowing techniques) for its own exposures.
  - One-digit century code data and apply Solution #1 (″date″ data type), Solution #2 (full four-digit solution), or Solution #3 (one-digit century code) for its own exposure removal.

- **Solution #4** (windowing techniques) to a certain module, another module that receives data from this module may either receive two-digit year data as before and apply Solution #4 itself or receive four-digit year data and apply Solution #1 (″date″ data type) or Solution #2 (full four-digit solution) or Solution #3 (one-digit century code) for its own exposure removal.

5.4.2 Other Programming Situations

Other programming situations you should consider might include:

- The possibility that a data format has become outdated and does not function correctly beyond December 31, 1999 (or earlier).

  Such data formats might be outdated even earlier and have already been superseded with another method by the Solution Provider.

- When migrating to Year 2000 support, your applications (operations) might support only a two-digit year format, only a four-digit year format, or both formats.

  It is possible that the two-digit values are assumed to be 19xx dates. Therefore, be aware that all of these must be eventually updated or the functions fail or give unpredictable results after December 31, 1999.

- Changes to operations procedures:

  Be sure to educate your operators about command changes so that they know when they must use a full four-digit date (for example, 2000, to avoid implying 1900 if they only enter 00).

- When erroneous data is produced for a limited and known time frame and changes are not justified:

  You might have a situation that is best handled manually to meet a short time period where programming changes simply are not justified. Consider using two-digit year data if a time frame such as a single 24-hour period (December 31, 1999 to January 1, 2000) or a single week (December 25, 1999 through January 1, 2000) is the only time your application does not provide correct results. For example, a program that looks at a sales report to compare the current day′s merchandise movement with the previous seven days. Because there are only eight reports containing both 1999 dates and 2000 dates, you might decide to handle the problem manually rather than changing the code.

Note: Do not fail to use a certain amount of common sense when deciding which applications to change, which to replace, and which to ignore. Do not lose your perspective of your institution's business needs and priorities and the impact and cost a particular application's change might have on attaining those goals.

5.5 Guidelines

While retaining a perspective of any external impact, module currency, and which functions are impaired due to Year 2000 exposures, use the following guidelines when applying Year 2000 solutions.

Note: This is not intended to be an exhaustive guideline, but rather a foundation upon which to start your specific Year 2000 date-data resolution.

1. Establish an in-house "date standard". Conformance to the ISO Standard 8601 in the following list is a valuable starting point. The earlier such a standard is in place, the sooner your IS organization can avoid creating new date issues and the propagation of current ones. You can refer to:
   - ANSI X3.30-1985 (R1991): *Representation for Calendar Date and Ordinal Date for Information Interchange*.
   - ANSI X3.51-1994: *Information Systems -- Representations of Universal Time, Local Time Differentials, and United States Time Zone References for Information Interchange*.
   - ISO 8601:1988: *Data elements and interchange formats -- Information interchange -- Representation of dates and times*.

   These standards are available in the following places:

   ANSI Online Home Page (access to ANSI catalog and standards documents):

   http://www.ansi.org

   OSI Online Home Page (access to ISO catalog and standards documents):

   http://www.iso.ch
2. Minimize potential impact to external references due to incompatible date format changes. For example,
   - Maintain the two-digit year format as an option when a four-digit format is required for an application program interface (API) that provides two-digit year data references.
3. **Avoid** any adhoc solutions; such solutions inevitably require future problem investigation and removal, and should be considered temporary solutions only. For example:
   - Do not determine the century of a two-digit year by comparing the two-digit year against a hard-coded threshold, for example, 60. If the two-digit year is greater than or equal to 60, the year is a 20th century year; otherwise, it is a 21st century year.
   - Do not fix the leap year calculation formula by adding logic to check if the current year is the year 2000. This solution temporarily fixes the leap year calculation problem by singling out the year 2000, but it does not fix the leap year calculation problem for other years in the multiple of 400.
4. A two-digit year format might be acceptable for human-only viewing purposes, for example, display panels, hardcopy reports, and so on.

However, any such data can be, and often is, added to a data set and then read by another program. A ″log″ that is used as input to any program should **not** be considered in this (non-impact, for human viewing only) category.

5. When changing the date format of any ″log″, ensure that all of the contributing programs adopt the new date format as well.
6. Consider the Year 2000 solutions described previously in this document for applicability in the following order:
   - Using a common date/time service routine (a four-digit ″solution″ that can support both two-digit and four-digit formats):
     - This is considered a long-term solution.
     - It is the **recommended solution** for its support of both two-digit and four-digit year formats that provide a long-term solution and no impact to two-digit year data references.
   - Solution #1 or #2 (conversion to a ″date″ data type or a full four-digit year format that externalizes four-digit formats):
     - This is considered a long-term solution.
     - It only supports four-digit year formats that have an impact on two-digit year data reference.
   - Solution #3 (conversion to a one-digit century code) that externalizes three-digit formats):
     - This is considered a reasonably long term solution.
     - It only supports three-digit year formats that have an impact on two-digit year data reference.
   - Solution #4 (windowing techniques that externalize both two-digit and four-digit formats):
     - This is considered a temporary solution and **should only be used when Solution #1, #2, or #3 is not practical**. (This is an arguable issue because there are applications that deal only with years in the range of 100 years. However, there is no guarantee that the functions of the applications will never change in the future and then require four-digit year formats.)
     - It has potential exposures when the function of the program needs to process years beyond the range of 100 years.
     - Use this solution only when:
       - Processing is always limited to the current date data, for example, at the time of IPL or time of job creation.
       - Expanding the date-data field is costly, and the function of the software program is phased out before any exposure occurs.

5.6 Search

A good tool should provide you with an easy way to find and modify your date and year fields. How difficult it is to find the fields depends on the naming standard that is used. If the naming standard has been good and consistent, using SEU may be helpful, even though it may be time consuming. The length of fields may also be used as a search criteria (for example, two-digit or six-digit

fields), or you may be able to search for special edit words. If you are using a tool, it may be necessary to define excludes. Be sure you search for all combinations where year is used, such as date, year, year-week, and year-quarter.

If you use ADS/400, you can use the normal generic search for field names from within a record for all of the files in the application. ADS/400 requires externally described files, and does not find fields defined internally in programs, but it knows in which files the fields are used and which programs use the files, and all objects to re-create if the field is changed.

You should keep a change log for all changes that are made for documentation purposes. You also want to be able to demonstrate that you were not responsible for existing program bugs.

5.6.1 Program Level Analysis

To analyze programs and data at a program level, you can find different tools that do these different tasks:

- ***Analyze data flow:***

  Shows the flow of data among modules in procedures or programs. It determines if a data-flow diagram is complete, consistent, and adheres to those rules that are established that govern flow. This provides both high-level and low-level views of data flow within the system and is used to verify completeness of the program and data changes.

- ***Diagram logic structure:***

  Diagrams how program modules call sub-modules, and what data and control information these program modules share. These tools display multiple program views.

- ***Diagram data structure:***

  Diagrams the representation of appropriate parts of a data model as the structure is used by a database management system structure and relational structures.

- ***Diagram relationships:***

  Illustrates multiple relationships of a program module or data element at the same time. This is useful to understand how data is shared among the programs that have access to it.

- ***Diagram decomposition:***

  Allows a high-level overview specification for a design or data model to be successfully decomposed into smaller entities for further observation and analysis. It facilitates the partitioning of a project that is too large to tackle all at once, such as the Year 2000.

- ***Slice programs:***

  Allows you to view all of the code affecting a given variable or statement. Forward slicing starts with a name or statement, and indicates what that name or statement affects. Backward slicing starts with a name or statement, and indicates all of the parts of the program that *could* affect it.

- ***Analyze logic:***

Inspects the use of control logic within a program, determines if it is proper, and mechanizes the specified design. It is useful for verification and validation of the correct manipulation of time when windowing techniques are used.

5.6.2 Code Editing and Restructuring

To help edit and restructure code for your programs, you can find different tools that do these different tasks:

- ***Power browse:***

  Allows you to scan and inspect code. Scanning can be switched between program (data) structure charts and code. These tools are more powerful than regular text editors. They can provide, for example, syntax checking, sophisticated capability to find data or information, or the ability to edit multiple programs. These types of tools can also include reverse engineering tools or maintenance workbench tools.

- ***Find dates:***

  Locates date-oriented data, variables, declarations, comments, or other information in code for investigation of potential reformatting.

- ***Comparison:***

  Compares two software programs, files, or data sets to identify commonalities and differences. It is extremely useful for the verification of program changes when date reformatting is done by simple field expansion.

- ***Cross reference:***

  Lists where variables, procedures, or other items are located in the code. These tools speed up browsing and provide a limited form of impact analysis.

- ***Expand fields:***

  Automatically expands two-digit year fields into four-digit year fields. It saves editing time tremendously and provides complete coverage of field expansion.

- ***Analyze interfaces:***

  Determines if a range of variables in the programming interfaces is correct as the variables are referenced across the reference boundaries. It can designate a four-digit year format as a standard interface and enforce the standard in the programming interfaces.

- ***Analyze standard/consistency:***

  Determines whether prescribed development standards have been followed. It identifies inconsistency in conventions used in requirements, designs, or programs. These tools can help you introduce standard or more uniform names to date-oriented fields or keywords and improve the consistency and accuracy of data. These tools enforce consistent indentation and alignment.

- ***Trace requirements:***

  Traces how the requirements are realized in the design and code.

- ***Modularize code:***

  Generates modular code and top-down control flow. It reduces the scope of complex programs by creating separate modules. It also identifies routines

that are frequently referenced or changed, for example, date/time services routines, and creates reusable code libraries.

- ***Standard date subroutine:***

  Creates reusable program modules that have correctly implemented date handling. These date subroutines can replace individually developed date subroutines to standardize the use of a date routine. These tools also reduce the chance of error and the cost of development, maintenance, and testing.

5.7 Externally Described Files

If the customer's applications are not using externally described files, all of the changes must be made within the programs. This may be a good time for the customer to introduce externally described files into the application. With externally described files and a field reference file as the base, you have fields defined in one place. In theory, this means that you should only need to change the field in one place and then re-create all of the files and programs that depend on this change. In files and programs, you must always refer back to the field in the field reference file. ADM/400 takes care of all the dependent re-creations if you choose to work with the application under the control of ADM/400.

5.7.1 Field Reference File

Having your applications well defined, this is the most important component to change. All of the fields used in the applications should be defined or refer to a field defined in the field reference file. When you use a field reference file, you can easily find and modify your date and year fields, and then re-create files and programs. Check attributes for format, edit words, and codes.

If century code or year is used as a separate field, you have to introduce a new field for this.

5.7.2 Physical, Logical, and Communication Files

If the definition of these files consists of fields that are referred to in the field reference file, you should not have to make changes to these files.

If the fields used are defined directly in the files, they have to be located and changed. If new fields for century are used, they have to be inserted in the format descriptions.

5.7.3 Display Files

Formats may have the problem of not having room for expanding the date and year fields. It may also be that users do not want to type more than two digits for a year. If the customer chooses to display only two digits for the year, few changes should be necessary, but you should make sure the programs display the correct two digits.

If the customer wants to enter only two digits, but stores the year in a four-digit format in the files, it is necessary to provide special coded routines in input programs to add in the century digits. On the other hand, if the windowing technique is used, special programs must be written to display the correct century digits.

If the customer wants to see all four digits on the display, it may be necessary to individually change each display format. If space is of no concern, it may be changed automatically, but if there is no room, formats must be changed individually.

Edit words and edit codes used with dates and years may be changed. Using date special words must be checked as well as using constants such as UDATE or DATE. Remember UDATE contains only two digits for the year.

Look for constants that contain century digits. You may have to change these to a field and have the program display the correct century.

5.7.4 Printer Files

Look for the same changes as for display files. The changes depend on the customer's choice of using two or four digits to represent the year.

Remember to check for preprinted dates or years. There may be reports with 19 as the preprinted century. The customer may want to change this to print a four-digit year.

5.7.5 Programs

In programs, you need to find where data fields are manipulated and defined, specifically if they are used in communications with other programs, data areas, data queues, or commands. Definitions may be in data structures or work fields.

For programs that do not use externally described files, you have to look for all date and year fields in the input and output definition section. This may be much the same as working on the files. Even programs with externally described files may use a re-definition that needs attention, such as data structures in RPG, the working storage section in COBOL, and declare data in CL programs. In RPG programs or other languages where it is allowed to define fields in places other than the input section, all fields with dates or years must be found and modified. It is important here to find where the fields have been manipulated to see if there is a resulting field that also needs to be changed.

Data areas should be treated as files if they are local, externally, or internally defined.

Using data queues may need to be changed if they contain dates or a year. Check for usage.

If other programs are called, the parameter definitions may need to be changed if they contain dates or years. Keep in mind that the receiving program needs the same changes.

If commands or APIs are used with date or year, the impact of a new definition has to be checked, especially commands such as OPNQRYF. Also, check for imbedded SQL working against date fields.

All calculation programming with dates has to be checked. New date operation codes such as those in ILE RPG4 solve this problem, but there are many date routines used in old programs that need to be modified. If you change the calculations or change to new operation codes, be aware of differences in calculations. You might not get exactly the same results as before. Check for the correct calculation of leap year in year 2000.

If tables are used, check for dates. They may have to be changed to contain century digits.

If your application uses special values in date fields for performing special tasks, this might be a good time to change this. Examples are the use of the value 99 in year, 12 in month, and 31 in day as the indication of the end of something, or the use of the date 99.99.99 or 00.00.00 as values when no date is specified.

If your programs are producing invalid data, this may be a good time to correct this. Since you are converting your data, you should clean up the files and then make sure you are not producing invalid data after the conversion.

Your programs may be changed to accept replications of year fields and date fields (one that is Year 2000 enabled and one that is not). A parameter tells which one to use. This may work if standard data handling routines are used in several programs.

You may want to consider using the function of a trigger program to do checking or conversion of the date. An example is to have a trigger program find the right century digits if only two digits for a year are provided. The digits can be placed in a data area.

5.7.6 Utilities

Utilities such as DFU and Query may have to be changed or re-created, depending on the usage of a date field or if files have changed.

5.7.7 Menu

Check menus for date fields. The display file or the menu may have to be re-created.

5.7.8 Online Information

Online information (Help Text) may need to be changed, but this is not essential for the application to run. You have to scan for dates in the text and look at the description. Panel groups and display files may have to be changed. You may also have to write more online information to explain the new handling of dates.

5.8 Modification

If you are going to use a tool to do the changes, you have to analyze the output to check that the modifications were done correctly. It is unlikely that you can find a tool that is able to find all of the dates that need to be modified. A good tool documents what it is not able to do and give a list of things that have to be checked manually. If you are using an interactive process, make sure manual modifications are not overrun by tool modifications.

5.8.1 Code Generation

To generate code for your programs, you can find different tools that do these different tasks:

- ***Generate database code:***

  Generates database code directly from the data structure diagram.

- ***Paint screen:***

Generates code for the displays of a computer-user dialog or data entry when the displays need an update for reformatting a date.

- ***Generate dialog:***

  Generates dialogs that conform to specified standards, for example, four-digit year input/output standard in any dialog.

- ***Generate reports:***

  Generates code for the structure and layout of a report along with calculations of derived fields in the report.

- ***Generate code:***

  Generates executable code from high-level specifications.

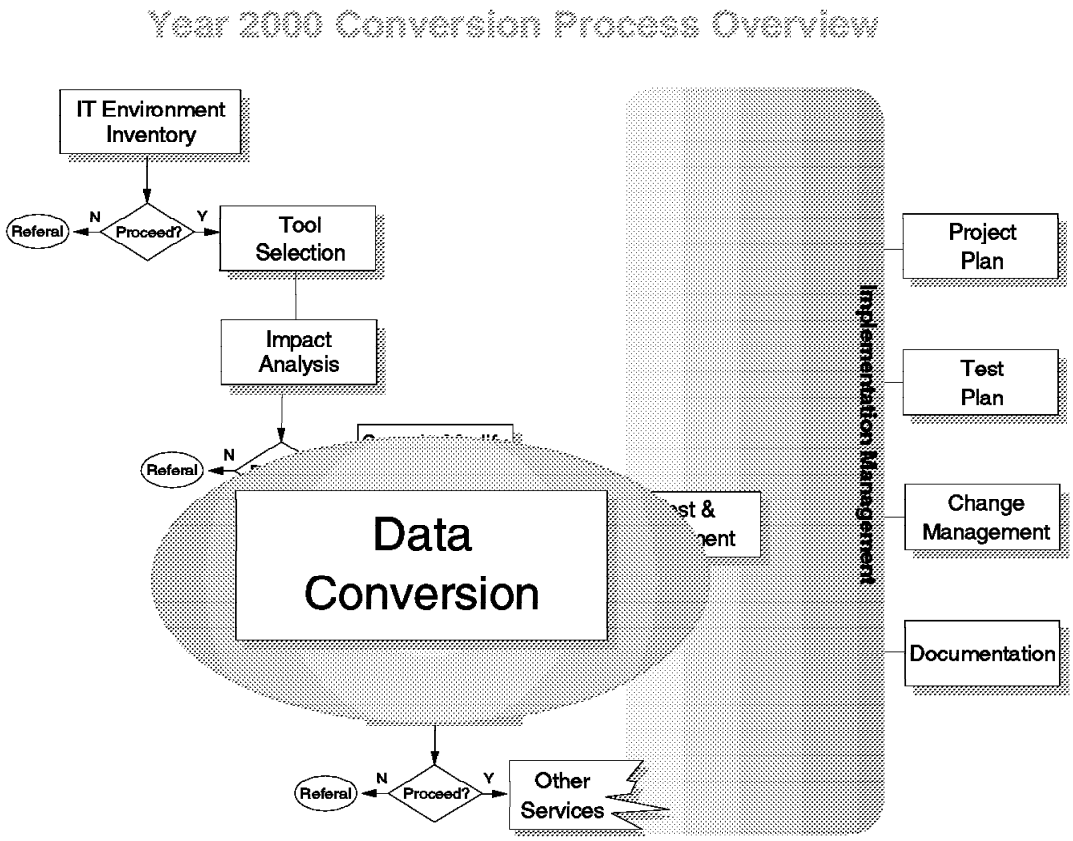
5.9 Build

The build procedure should take care of re-creating objects that have been changed or are dependent on changed objects. Be careful to not mix new and old objects. ADS/400 and ADM/400 may perform these functions.

5.9.1 Security

Keep in mind security concerns when creating objects. Old security may be lost. New objects normally get the same security as the library they are created in. You may use the GRTOBJAUT command to change authority or refer to other objects for reference. If you use a tool for the build procedure, check if it re-creates the security. ADM/400 uses security from old objects if they are replaced with the export function. Refer to the *AS/400 Basic Security Guide*, SC41-3301, or *AS/400 Security - Reference*, SC41-3302, for details.

Chapter 6. Data Conversion



6.1 Inputs and Outputs

Once your programs are enabled for the Year 2000, it is time to convert your data. Converting the data to be Year 2000 enabled requires careful planning. As in the application modification phase, you use the Impact Analysis report as input for the data conversion phase. The output of the data conversion phase is a live database using Year 2000 enabled data. The data represents year and date fields in conformance with the format that the customer has chosen to use for the Year 2000 enabled applications.

6.2 Alternatives

If you are using a windowing technique for Year 2000 enablement, you may not need to convert much of the data. But if you are introducing century year/code to the year field, there are several things to consider:

- You may need additional disc space to handle the expansion of the files using four digits for year fields.
- You may exchange data with other systems that do not do the conversion at the same time.
- If you have partitioned your applications, you need bridge programs.

- The old data may contain invalid or reserved values that need to be corrected. They may have worked before, but do not work with the new code.
- Files may need to be reorganized as new keys are built. Century digits may be included in the key fields. New files (logical) may be needed.

If not all data has to be converted, there must be strict rules for what data should be Year 2000 enabled and what should not. Data that should be Year 2000 enabled has to be converted, while data not enabled must be handled the ″old″ way or with bridging programs. It is important that there is a clear line between data that has to conform to the new format of Year 2000 and data that does not need to.

6.3 Customer Responsibility

The customer is responsible for setting the rules for what data needs to be Year 2000 enabled and what does not (that is, what data must be bridged and what does not need to be considered as part of the conversion). In some cases, the customer may want you to clean up other fields in the database at the same time you are doing the Year 2000 enablement work.

The customer has to provide the environment for conversions to be performed in-house. Additional hardware may be needed.

6.4 Considerations

The following sections highlight some of the key considerations that should be reviewed at a macro level before embarking on a Year 2000 project.

6.4.1 New Hardware

Conversion is a one-time effort for the data being Year 2000 enabled. There may be a need to change the hardware on the system or at least getting more disks. Most customers are on the edge of needing more disk space. If new hardware is brought in, this should be looked on as a normal upgrade and not be mixed with the conversion of data to Year 2000 enabling.

There may be situations where you have to run systems in parallel, either to do parallel testing, or because you are in a phase to change to a new system and your cut-over has to be done in phases. The new system may be in-house or outside. You must consider how you unload/reload data and applications. Synchronization is also an issue.

6.4.2 Cleaning Up Data

The customer′s files may contain invalid data such as blanks in numeric fields. You should use this occasion to clean up the data to avoid problems later.

You might also use this opportunity to get rid of special values. Year 99 may have been used to denote ″no expiration date″. This has to be changed.

You may need to write programs to clean up all of the files so they do not contain any bad data. Be sure to fix the programs that are introducing this data at the same time. Programs are needed to filter the data files by reading

records and then writing correct records or they may just update the record with correct data in the files as they are.

6.4.3 What Not to Convert

You should check for data that does not need to be converted. You may have historic data that does not need to be brought forward to the new century handling. You may keep your old programs for running with this data if necessary.

Other data may be obsolete and is never used again, so there is no need to convert the data. The file description or layout, on the other hand, may have to be changed to be able to accept data with the new year format.

In some cases, it may be necessary to replicate the date field so it is stored in both formats. New logical files may have to be created to point to the correct date field, and a parameter can be used to point to the right file and programs to use. If historical data is stored in this format, a conversion has to be done.

There may be cases where data is only converted when accessed. This means that the conversion takes place over time and only used data is converted. This works when the retention period is finite and a subset of the data is accessed. Archived data coming from image applications or transferred from microfiche may be an example of this.

6.4.4 Conversion Programs

To build conversion programs, you must know the old and the new date format and the consequences that a change has on your files. You must find out if files have to be converted or if bridge programs have to be built.

Files may have to be altered, or new data may have to be included for such things as including the century code or a switch. It may be necessary to create new files or access paths.

Remember also to convert the contents of data areas if necessary.

Some tools used to analyze and modify the source may also create conversion programs for the data files. You may have to write conversion programs from scratch. You may have some from previous work that you can modify. In some cases, you may be able to use utility programs or a copy file command. With SQL, you are able to easily make changes to the entire file.

Special programs for scrubbing or cleaning data may be made or included in the bridge programs.

6.4.5 Bridge Programs

Bridge programs are important. They perform an important function by letting data that is not Year 2000 enabled work with data that is Year 2000 enabled so the goal is to make them transparent to their function and not impact performance. Bridge programs must handle data coming from other systems electronically, from other applications, or from part of the application that is not at the same level of conversion as the one you are working on.

The bridge may be both ways, both from Year 2000 enabled data to data that is not Year 2000 enabled data, and vice versa. The programs should satisfy

performance requirements, maintain sufficient data integrity, and contain sufficient validation and controls. When you implement the bridge alternatives, you should look for ways to avoid the need for a bridge to do online file updates by typing the transaction twice (once for Year 2000 enabled data and once for data that is not Year 2000 enabled). You may allow data records to pass through the bridge and allow separate update programs to do updates.

6.5 Statement of Work

To accomplish a successful conversion and eliminate your current Year 2000 exposures, you need to follow a well-architected conversion plan and prepare to execute that plan prior to actually performing the conversion. This section provides an outline that you can use as a checklist of those steps to help you plan, prepare, and execute your conversion. Note that some steps may not be necessary for your specific environment.

6.5.1 Plan for Conversion

- Plan for conversion:
  - Determine the sequence of steps needed for conversion.
  - Review the conversion procedures with your system administration staff and your end-user community.
  - Determine the resources/time required for conversion.
  - Assign individuals/organizations to each conversion step.
  - Document the conversion sequence and responsibilities.
  - Develop a schedule for converting the new system to reach production mode.
- Examine all changed data that use a new and changed date format:
  - Determine the source of the new data in the existing systems.
  - Determine the data that can be converted automatically.
  - Determine the data that must be converted manually.
- Design bridges and interfaces among packages and reusable modules to maintain compatibility, if needed:
  - Design bridges/interfaces to application packages, if needed.
  - Design bridges/interfaces to reusable application systems, if needed.
  - Design bridges/interfaces to old systems that coexist with the new systems, if needed.
  - Design tests for the verification and validation of these bridge facilities, if needed.
- Design procedures for manual data conversion:
  - Update documents and procedures that are used for manual data entry.
  - Determine checking mechanism for the accuracy and completeness of manually entered data.
  - Design the new displays with a new date format for manual entry of new data and review with your end-user community.

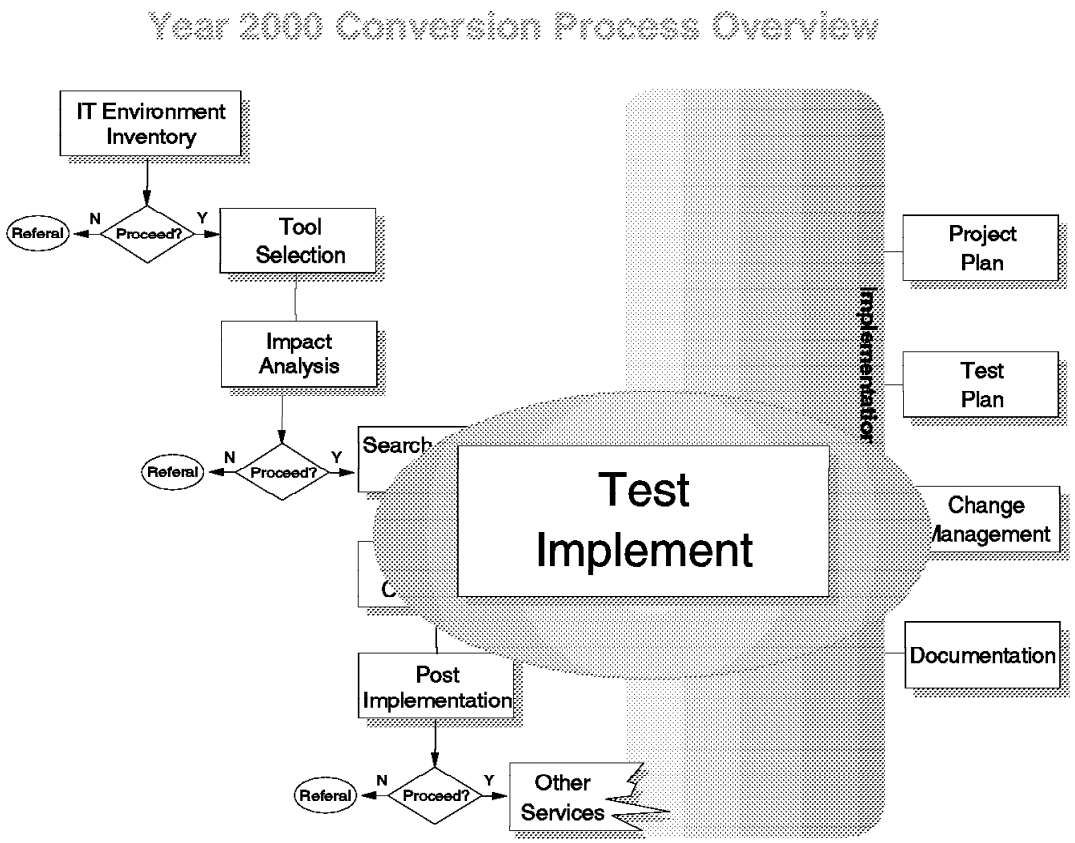
  - Design and update the software to load the manually-prepared data into the new system.
  - Run a rehearsal of the manual data entry and estimate the impact of the new data format on data entry time.
- Design procedures for automated data conversion:
  - Design new software or use automated tools for automated data conversion.
  - Determine a checking mechanism for the accuracy and completeness of automatically converted data.
  - Design recovery procedures for conversion of data errors caused by missing data.
  - Estimate the resources and time for automated data conversion.
- Develop the data conversion systems:
  - Develop subsystems to convert existing data.
  - Develop subsystems for the entry of new data.
  - Develop bridges and interfaces to old systems that remain in production.
  - Develop bridges and interfaces to application packages and reusable modules.
  - Verify and validate the accuracy of the data conversion systems.
- Plan the hardware installation for a new system, if needed.
- Plan for final system testing:
  - Determine the testing strategy.
  - Develop the detailed test plan and schedule.
  - Determine the types of tests to be conducted on the new system.
  - Plan the testing environment:
    - Design the conversion tests for the systems and applications.
    - Determine what testing software or tool (or tools) are used for each type of testing.
    - Determine what testing libraries are used for each specific set of programs and data.
    - Install needed testing software (for example, test data generator, test utilities, debugging utilities, and so on).
    - Build test libraries and test data.
    - Coordinate the testing with your development team and your system administration staff.

6.5.2 Perform Conversion

- Update production procedures, if necessary.
- Install the Year 2000 enabled production system environment:
  - Coordinate with vendors for the hardware installation, if needed.
  - Install the hardware of the new production system, if needed.

  - Coordinate with system programmers/operators for installation of the Year 2000 enabled software.
  - Install the Year 2000 enabled system/vendor/application software on the new production system.
- Perform data conversion process:
  - Load existing data into the new system's databases.
  - Execute the data conversion programs or automated tools for data conversion.
  - Load manually-prepared data through data entry.
  - Integrate the existing and converted data.
  - Test the integrated data to verify data integrity.
- Perform final system/conversion testing:
  - Plan the sequence in which separately developed subsystems are tested and verified in reasonable combinations.
  - Verify that the portions of the system that have no changes still run properly as changes are made to other portions of the system.
  - Verify that the program handles all of its transactions correctly and remains stable for a defined period of time.
  - Verify that the system can accept input from and provide output to other systems with which it interfaces as interfaces change.
  - Verify end-user acceptance of the new system to certify the system as acceptable for production.
- Activate the new system in production:
  - Switch the new system to production mode.
  - Run the new system in parallel with the old system.
  - Phase out the old system as the new system becomes stable.
- Conversion review:
  - Monitor and evaluate system performance, throughput, and reliability.
  - Determine what system tuning is needed based on system status records.
  - Track and evaluate user acceptance of the new system.
  - Determine and document what system and application function enhancements are needed.
  - Plan and schedule the system and application function enhancements.
  - Coordinate system function enhancements with vendors.
  - Design and develop required in-house application function enhancements.
  - Determine when the system and application enhancements are applied.
  - Apply the enhancements, once available, to the new system.

Chapter 7. Test and Implement



7.1 Test Process Overview

It has been estimated that up to 70% of the time it takes to enable an application for the Year 2000 is spent in testing. Therefore, the test process should be well understood.

The test process can be broken down into the following processes. The purpose and deliverables for each of the functions are listed in this chapter. A further breakdown to an activity level and greater detail is done for some of the processes that are specific to Year 2000 issues.

The major processes of test and implementation are to:

1. Develop a master test plan.
2. Develop a detailed test plan.
3. Prepare test data.
4. Run the tests.
5. Report test results.

7.2 Process 1.0: Develop Master Test Plan

Purpose

The master test plan addresses testing from a high-level system viewpoint. It describes all testing activities such as unit, integration, system, user acceptance testing, and operability testing. Its goal is to prevent missed or duplicated testing during the test process and to provide an early definition of the system test environment.

Deliverables

- Test team organization, roles, and responsibilities
- Test schedules
- Test facility requirements
- Test focus
- Test objectives
- Test data
- Test levels, types, and strategy
- Build strategy
- Problem and change management procedures
- Master test plan

7.3 Process 2.0: Develop Detailed Test Plans

Purpose

The purpose of the detailed test plans is to plan the effort for the levels of testing that were included in the master test plan. During this phase, business and structural functions are decomposed and test matrices are developed to assure that test objectives are met. A ″build″ strategy is agreed upon to optimize the development and test efforts. Testing techniques and tools are decided upon. Staffing assignments are planned. Administrative procedures and controls are established. Configuration management procedures applicable to the level of testing are decided upon. In general, the detailed plans resolve all questions relating to exactly which tests are going to be done, by whom, and when.

Deliverables

- Business functions
- Structural functions
- Test matrices
- Administration procedures and controls
- Integration approach
- Test facility and environment setup plans
- Detailed test plans

7.4 Process 3.0: Prepare For Tests

Purpose

This is the step where plans are turned into actions. Test material must be created containing both valid and invalid test data. Existing test data may be used, or sometimes the data must be created from scratch. Temporary

programs may be needed to create or receive data from the specific programs under test. Special environments may need to be designed and built to simulate the production environment, or to interface with applications outside the conversion scope. Instructions may be written to guide the testers in how to conduct the tests.

Deliverables

- Testing techniques and tools
- Test conditions and cases
- Test run procedures
- Test facility and environment setup

7.5 Process 4.0: Run the Tests

Purpose

Run the test cases in the different environments, namely, development, system test, user-acceptance test, and operations. Log test results (success, failures, or discrepancies) and any incidents found.

Deliverables

- Unit/integration test results
- System test results
- Operability test results
- Incident reports (for every level of test)

7.6 Process 5.0: Report Test Results

Purpose

The purpose of this activity is to report the test results and learn from the experience in order to reduce the defect rate in the future. Analyze the defects looking for patterns. Discuss what worked and what did not work. Gather overall defect statistics and assess how closely you have come to meeting the projects goal.

Deliverables

For every level of test:

- Verified results
- Variance reports
- Test reports
- Defects statistics
- Test scripts/data
- Project experiences

7.7 Testing Techniques

Testing can be formalized into a four-phase process as follows:

Test Type	Used to Test
Unit testing	A single program module
Integration testing	A related group of program modules
System testing	The entire software application
Acceptance testing	The entire software application with live data for production readiness

Ideally, these phases should be completed sequentially. However, when development work is done in parallel, module coding, unit testing, and integration testing are commonly integrated followed by system testing and acceptance testing.

During the process of testing, apply a combination of verification and validation techniques. Unit and integration testing are primarily used for program verification. These two forms of testing comprise structural testing, which is used to uncover errors injected during program coding. System and acceptance testing are used for program validation, and these two forms of testing comprise functional testing, which is used to uncover errors that occurred when implementing requirements or design specifications. The following sections cover some useful testing techniques and scenarios for Year 2000 testing.

7.7.1 Structural Testing Techniques

Structural testing ensures sufficient testing of a function′s implementation and helps determine that all structures of the system are integrated to form a cohesive unit.

7.7.1.1 Operations Testing

Apply operations testing to determine whether the system is ready for normal system (production) operations. In contrast, recovery processing (discussed in Section 7.7.1.3, “Recovery Testing” on page 73) is intended for abnormal system operations. Considering the potential scope and magnitude of your Year 2000 transition, every aspect of the normal operation might be impacted to some extent as you revise programs and data for Year 2000 readiness. Operations testing ensures that prior to production, your IS staff can properly administer the applications using the new support mechanisms, documentation, procedures, and training as you complete your Year 2000 transition.

7.7.1.2 Stress Testing

Apply stress testing to determine if the system can function when transaction volumes are larger than normally expected. The typical areas that are stressed include disk space, transaction speeds, output generation, computer capacity, and interaction with people. When testing Year 2000 changes, it is essential to verify that the existing resources can handle the normal and abnormal volumes of transactions after the restructuring of the code and the possible expansion of the data fields. For example, apply stress tests to determine:

- If existing CPU capacity is sufficient to meet expected user turnaround time when a particular solution is applied that uses more CPU cycles and processing time for code conversion.

- If existing disk capacity is sufficient to accommodate the additional disk space and provide acceptable disk access time when a particular solution is applied that expands the year data field.

7.7.1.3 Recovery Testing

Apply recovery testing to ensure that the system can restart processing after losing system integrity. This is essential for systems in which the continuity of operation is critical to end users. Recovery processing normally involves the ability to go back to the last checkpoint and reprocess up to the point of failure. The success of the recovery depends heavily on complete backup data and checkpointing. Any data integrity or unresolved exposures that lead to inconsistent data or code after you have implemented appropriate Year 2000 solutions affects the completeness of backup data. On the other hand, checkpointing is time-oriented and sensitive. Any mishandling of the time-related data might invalidate system checkpointing. The recovery testing is critical in a Year 2000 testing environment. It can also involve manual functions (such as hardware or operating system failure), loss of database integrity, operator error, or loss of input capability. Recovery testing should include all aspects of the recovery processing.

7.7.2 Functional Testing Techniques

Functional testing is designed to ensure that the system and end-user requirements and specifications are achieved. Functional testing focuses on the results of processing rather than how processing is implemented. To accomplish this, create test cases to evaluate the functional correctness of the system and programs. Functional testing techniques are discussed in the following sections.

7.7.2.1 Requirements Testing

Apply requirements testing to verify that the system performs its function correctly and that it remains functional over a continuous period of time. Functional checklists such as user requirements, design specifications, and the compliance of an organization′s polices and procedures are used to create test cases to ensure that these requirements are still satisfied following your Year 2000 transition. Note that if the Year 2000 solutions are merely restructuring code and reformatting data without major redesign of the applications or systems, most requirements testing can be covered by another method, regression testing.

7.7.2.2 Regression Testing

Apply regression testing to ensure that all aspects of a system remain functionally correct after changes have been made to a program in the system. Because the potential exists for a tremendous amount of data and programs to be involved in your Year 2000 transition, any change to an existing program in the system can have a snowballing or cascading effect on other areas in the system. A change that introduces new data or parameters, or an incorrectly implemented change, can cause a problem in previously tested parts of the system, simply because of the way data can be shared between software entities.

Regardless of how an error was introduced or propagated, regression testing needs to be conducted to retest even unchanged parts or programs of the system. Normally, tests that have been previously run are reused to ensure that

the same results are achieved. In most cases, regression testing is automated because the test cases and the results are already known.

7.7.2.3 Error Handling Testing

A normal error-handling cycle is an iterative process that either prevents errors from occurring, or recognizes and corrects errors that have occurred. Error-handling testing is necessary to determine the ability of the system to properly process incorrect transactions that can be reasonably expected as types of error conditions. For example, programs that accept only four-digit year, data, or entry format need to provide error messages for data entry in two-digit year format, and vice versa for programs that accept only two-digit year, data, or entry format. When changing from a two-digit year format to a four-digit year format, you need to apply error-handling testing to verify the appropriate error-handling functions.

7.7.2.4 Manual Support Testing

Apply manual support testing to evaluate the adequacy of the processes used by people (end users) who must handle the new data generated from the automated applications with Year 2000 support. Types of data from these applications include data entry and report generation. Any new data format should be easy to understand and not ambiguous. This method includes testing the interfaces (for example, displays, procedures, operation manuals, and online information (Help) displays) between end users and the application program. End users should be trained and use procedures provided by the system personnel. Testing should be conducted without any other assistance.

7.7.2.5 Intersystem Testing

Applications are frequently connected with other applications to provide a higher or deeper level of functionally. Data may be shared between applications or systems. Multiple applications or systems may be involved in such an environment. This is the typical environment for Year 2000 projects. Intersystem testing is required to ensure that the connection functions properly between the applications. This test determines that the proper parameters and data are correctly passed between applications, and proper coordination and timing of each function exists between applications.

7.7.2.6 Parallel Testing

Parallel testing is used to determine whether the processing and results of a new version of an application are consistent with the processing and results of the previous version of the application. It should be applied when the old and new versions of the application are similar. For Year 2000 solutions without any major function redesign, this is the ideal technique. Parallel testing requires that the same input data be run through the two versions of the application. However, if the new application changes data formats, such as reformatting the year-date notation to four-digit format, you must modify test input data before testing.

The efficiency and effectiveness of parallel processing is highly dependent on the degree of difficulty encountered in verifying output results and preparing common input. It may be difficult to automatically verify the results of processing by comparing the results on a tape or disk file. Some automated test tools or customized solutions can be used to prepare input and verify output more quickly.

7.7.3 How to Change Date and Time for Testing

By nature, Year 2000 exposures are time-sensitive and time-driven. Basic Year 2000 testing requires that you set the system date and time to a point where Year 2000 exposures can be detected and then removed.

Attention: Be cautious before resetting the system timer. Some system resources and functions are time-sensitive and may be activated or de-activated when you reset the system clock. Such effects can occur when you either set the system clock forward or backward. Without careful planning, you can cause the loss of these system resources and functions, some of which might prove difficult and time-consuming to recover. Ensure that you do not contaminate your production system or production databases when running various test scenarios. Consider:

- Providing a separate test system and storage device (or devices).
- Providing a separate set of test data.

The most vulnerable resources and functions subject to expiration include:

- User IDs
- Passwords
- Data files and databases
- Authorization/protection
- Licences/services
- Network access
- Automation functions (as well as unexpected activation)
- Hierarchical storage management

You can change the system date and time as follows:

- **On an AS/400 system:** use the Change System Value (CHGSYSVAL) command with the QDATE parameter. For example,

  ```
  CHGSYSVAL QDATE('101300')
  ```

  changes the system date to October 13, 2000 on a system using MMDDYY date formats (QDATFMT). To make sure all jobs on the AS/400 system are using this new date, you should then switch off power on the AS/400 system using the Power Down System (PWRDWNSYS) command and conduct your testing after the subsequent IPL.

7.7.4 Current Year Testing

This is the first task you have to perform. Run the test on the modified application using current year test data and compare the result against the baseline. Record any discrepancies and analyze these with the customer. (This might take you back to square one.)

When the test proves that no new bugs have been introduced, you should get the customer's approval and obtain a **Sign-off** of the current year test results.

Showing that the revised application still works is, however, not the ultimate goal of the entire exercise. Now you have to turn the clock forward and repeat the test using 21st century dates.

7.7.5 Year 2000 Testing

This is your next challenge. You have to review the test data and shift the dates to reflect Year 2000 before you kick off the test.

These are the steps you must perform:

1. Review current year test data for Year 2000 testing.
2. Execute routines to shift input data files and databases to Year 2000 dates.
3. Revise system parameters and system date for Year 2000.
4. Run the revised programs using Year 2000 data.
5. Analyze the result and compare to baseline.
6. Isolate and record any discrepancies.
7. Schedule and perform changes in components.
8. Retest discrepancies.
9. Document and review with customer.
10. Obtain sign-off of test result.

The only thing remaining before you transfer to production is to ensure that any maintenance done to the running application has been documented and implemented in your Year 2000 enabled code.

7.7.5.1 Basic Testing Scenarios

The scenarios for Year 2000 testing depend heavily on the system environment and applications. Some basic Year 2000 testing scenarios that are common for most installations are suggested here:

- Set the clock to test process cycles and automatic functions that are activated on a regular basis. These scenarios can be used to identify Year 2000 exposures that need to be fixed as well as to validate programs after applying Year 2000 solutions.
  - Daily
  - Weekly
  - Semi-monthly
  - Monthly
  - Bi-monthly
  - Quarterly
  - Semi-annually
  - Annual
  - Automatic archiving
  - Automatic restart/restore
  - On demand
- Test the setting and display of special dates, including:
  - 1900/2/29 should fail; the year 1900 is not a leap year.
  - 1996/2/29 should succeed; the year 1996 is a leap year.
  - 2000/2/29 should succeed; the year 2000 is a leap year.
  - 00/01/01 should display an unambiguous four-digit year date, the value of which depends on the application (for example, 1900/01/01, 2000/01/01, and so on).

- 1999/12/31 should be able to distinguish between a regular end-of-year 1999 date and a special meaning date (for example, a never-expiring date indicator).

- Test the processing of time-sensitive data with different combinations of data and time:

  1. Use the current system clock and test data with dates:
     - Before 2000/01/01
     - After 2000/01/01
  2. Set the system clock before the year 2000 (for example, 1999/12/31), and test data with dates:
     - Before 2000/01/01
     - After 2000/01/01
  3. Set the system clock after 2000/01/01 and test data with dates:
     - Before 2000/01/01
     - After 2000/01/01

7.8 Look Out For

There are a lot of pitfalls you might stumble into, and the tool you select may not pick them all up. In fact, the tools we have tested in most cases do not spot any of the pitfalls. Be prepared to use standard CL commands to manually examine the application and system values to identify them.

7.8.1 Client/Server Applications

Many AS/400 installations today are participating in some kind of networking environment. You certainly find sites where the AS/400 system interacts or communicates with the outer world in some way or another. Finding these connections and determining whether they are affected by the changes you are about to inflict on the system are of utmost importance.

Be aware that the system you are about to convert can act both as a server and as a client. Examine the configuration of the communication controllers and look for:

Host systems
S/390, AS/400 system, and other...

Client Systems
AS/400 system, PC′s, and other....

Most PC based client/server applications are run on the PC, and use the AS/400 system primarily as a database server. These applications probably are affected by the changes you make to the date fields in the database. Finding these applications can be rather difficult. Most of them, however, use remote SQL or ODBC to access the data. Looking for objects of type ***SQLPKG** in library QGPL may help you in this task.

*SQLPKGs contain the access plan for an SQL query or program and they are given the same name as the program containing the SQL statements. If you find an *SQLPKG with no corresponding *PGM elsewhere in the system, chances are that the package has been generated by a client application accessing the AS/400 database.

Analyze the content of the *SQLPKG using the PRTSQLINF command to determine whether any files containing modified date fields are processed. If so, delete the *SQLPKG , make any changes that are required to the query, and resubmit it to have the *SQLPKG re-created.

7.8.2 Remote Applications and Communications

Distributed applications present another challenge. You may find that your customer exchanges information over the network in a number of different ways. These applications can use a number of different mechanisms and standards to communicate, and the nature of the data is, of course, diversified. We do not get into the area of office applications here, since e-mail and exchanging office documents normally is not affected by the changes you make to date fields in the database.

There are, however, other types of applications that you must look out for. One is the automated distribution of standardized documents such as order acknowledgements, invoices, and so on. These documents often contain date fields such as delivery date, due date, and so on. One other important type is electronic funds transfer.

Although difficult to find using automated tools, there are some types of objects you can scan for in order to identify these applications:

Integrated Communications Files (ICF)
Distribution queues
Tape files
Diskette files
PC file transfer
Other.....

Since there is a great probability that these remote applications are not converted synchronously with your local applications, you must create some type of bridging function to be able to keep these applications running.

7.8.3 Vendor Applications

If your customer runs standard applications purchased from a software vendor, you probably do not have access to the source code, and, thus, cannot modify the application to be Year 2000 compliant. You and your customer must decide whether the purchased application is affected by the year 2000 problem or not. If it is, a new version must be purchased and installed. The next thing to determine is whether the purchased application in any way interacts with or utilizes data from any application or component that is Year 2000 enabled. If so, you may have to write some kind of bridging function, even if the purchased software is replaced, since it may not be feasible to synchronize this with the conversions you make.

7.8.4 Bridge Programs

The applications that you are enabling for Year 2000 probably are exchanging information with other applications and systems. These can be run either locally on your AS/400 system or remotely on another AS/400 system, S/390, or other platform. All of these interconnected applications are probably not Year 2000 enabled at the same point in time; in fact, some of them might not need to be updated at all.

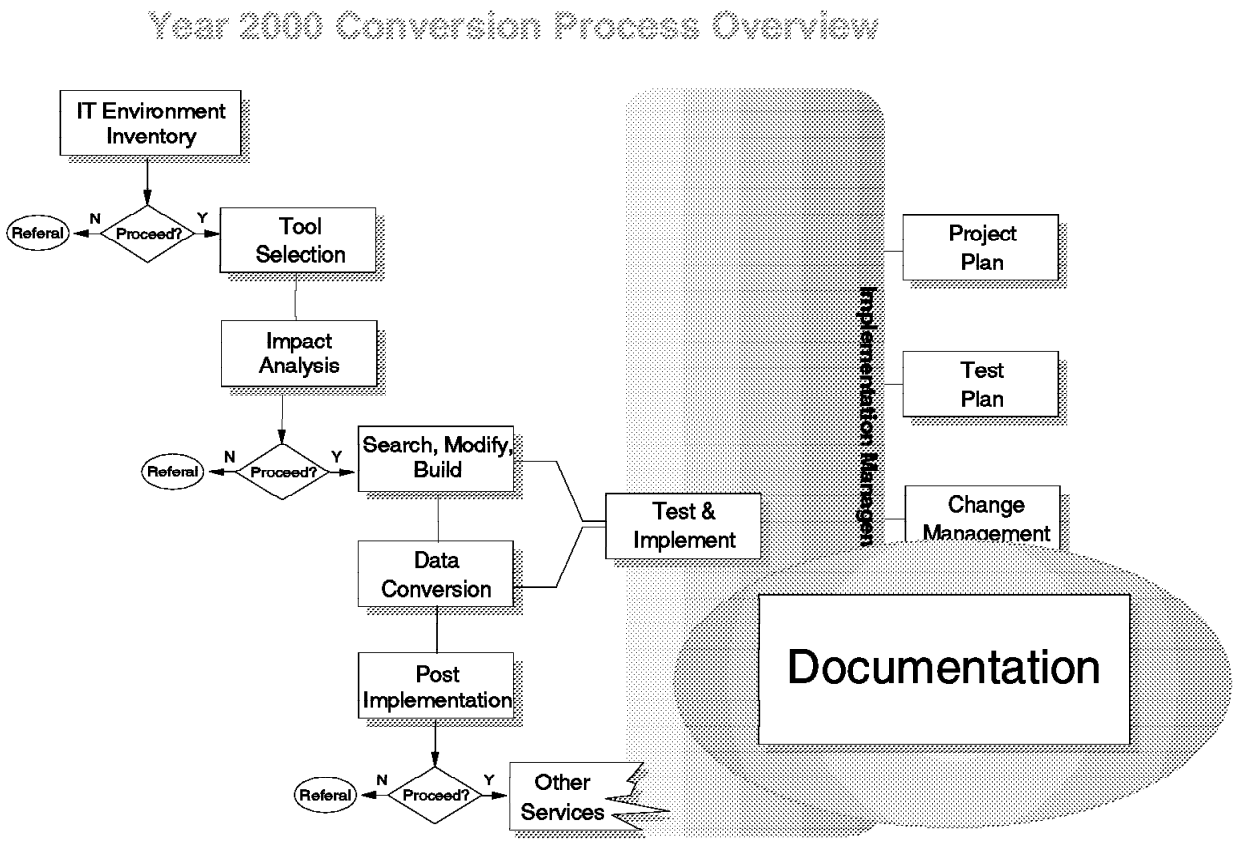
You can also possibly find instances where your customer wants to make exceptions to the rule. Single applications or displays might not allow the expansion to four-digit years, so for this particular instance, you have to add the century digits within the logic of the program.

This makes it necessary to create ″bridges″ or ″filters″ to translate between ″two-digit year″ and ″four-digit year″ information that is exchanged between the applications.

Please consider the performance implications when choosing what bridging method you choose. Using external calls to a bridging program can be CPU-consuming versus using Trigger programs or the RCVJRNE command to duplicate data in the desired format, which fills your available disk capacity.

Whatever method you choose for bridging, you must plan for additional hardware, or consider even moving the application to ILE. Recoding your application in an ILE language allows you to call bridge programs ″bound″, which causes significantly less overhead in terms of CPU usage than the traditional dynamic calls used in the OPM environment.

Chapter 8. Documentation



In order to help the customer fully understand the scope of the changes you have made to enable the application (or applications) for Year 2000, you must produce comprehensive documentation for all of the changes you have made. This documentation should include:

Documentation	Purpose
Change log	Shows all changes to the application.
End user training	Makes the end users familiar with the changes to the application.
Data dictionary	Facilitates future maintenance of the application.

8.1 Change Log

The change log is the most important part of the documentation. It enables your customer to verify that all of the items identified during the Impact Analysis have been taken care of, and can aid you in resolving any errors that, in spite of testing, might have been introduced during the conversion. It also makes it clear that problems that existed before the Year 2000 conversion were not the result of your changes.

The change log should cover any alterations made to:

- Files, data queues, and data areas
- Programs
- User interfaces
- Policies and rules

8.1.1 Changed Files, Data Queues, and Data Areas

All objects containing date type information that has been changed during the conversion should be listed and the changes should be highlighted. Most of the basic changes are probably included in the field reference files and propagated through other object types. Nevertheless, you should include all changed descriptions, making the listing as complete as possible, including:

- Field reference files
- Physical files
- Logical files
- Printer files
- Display files
- Integrated communications files
- Data queues
- Data areas

8.1.2 Changed Programs

We suggest that you only include in the documentation the changes you have made to the programs rather than a complete program listing. Remember that the purpose of the documentation is for the customer to understand the changes you have made, and not to document their application as such.

Retain the original code in the source members and comment it out so that you easily can identify both ″before″ and ″after″ images of the altered code. The easiest way of organizing the information is probably to arrange by source code file, giving you a listing structured by programming language and execution environment. Include all affected source files such as:

- QBASSRC, (BASIC)
- QRPGLESRC, QRPGSRC, QSQLRPGSRC, (RPG)
- QCBLLESRC, QCBLSRC, QLBLLESRC, QLBLSRC (COBOL)
- QCSRC, (C)
- QCLLESRC, QCLSRC, (CLP)
- QCMDSRC, (CMD)
- Queries
- QMF formats
- Other languages ...

The preceding list is not complete, but serves merely as an example of sources containing information about changed programs.

8.1.3 Changed User Interfaces

Use print screen to document any changes made to display files, menus, panel groups, and commands. This also aids in producing educational material for the end users. Include samples of the reports that have been changed and highlight the areas that were affected.

8.1.4 Changed Policies and Rules

Policies and rules regarding date computation may have been altered to facilitate Year 2000 enablement. These changes were probably agreed upon during the initiation of the conversion project and documented there. Regardless of this, you still need to document these changes in the documentation of the finished project.

Examples of policies and rules that may have changed during the conversion are:

- Using new date functions and APIs versus old methods
- Changed date formats and date presentation in:
  - Database files
  - Display files
  - Printer files
  - Related documents and forms

8.2 End User Training

Some of the changes you have made and documented in the change log can affect the way users interact with the system. Just producing documentation may not be sufficient to make the users comfortable with their updated system.

You should at least produce documentation targeted for end users and describe how your changes affect their work. Preferably, you should also produce material to perform lectures and training sessions for the users in order to make them fully familiar with the updated system.

The educational material should cover all of the areas requiring user interaction such as:

- Data entry
- Queries
- Reports
- Forms and procedures

8.3 Data Dictionary

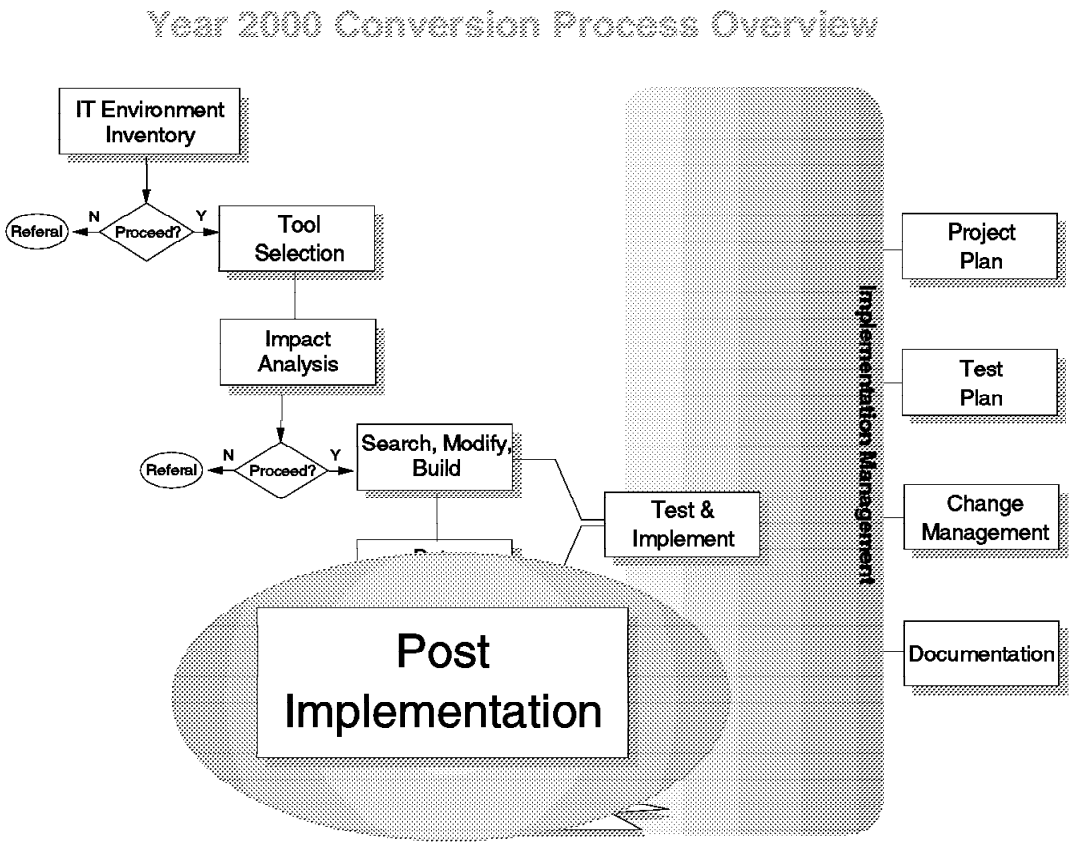
In order to facilitate future maintenance of the application, we strongly recommend that you establish a data dictionary, documenting the system as it has been updated. Consider using a tool such as Application Dictionary Services (ADS), a feature of Application Development ToolSet/400 (5763PW1).

8.4 Additional Documentation

During the conversion, you probably have identified report layouts that have been altered. In some cases, these reports may be external (for example, invoices, statement of accounts, and so on).

It is recommended that you specifically highlight these to your customer. This enables them to inform the recipient of these documents about the changes that are made, and how these changes should be interpreted.

Chapter 9. Post Implementation Services



This chapter describes the support you may provide to the customer (such as warranty services) once the conversion and testing phases are complete. You need to be clear on the scope and duration of this service. Some customers want it for a short duration, while others may want you to provide continuing support. It is important that you clearly state your warranty intentions in the statement of work so that it is easy to identify the end of the project and, hence, the end of your relationship with the customer. Be sure to make the appropriate adjustments to the cost of the overall project based on the duration of the warranty.

If the post-implementation services are not a part of the initial proposal, you may need to submit a new financial proposal to the customer spanning the explicit services that the service provider wants to provide.

9.1 Application Warranty

During this service, you are obligated to fix any problems encountered during the warranty period that were introduced as part of your Year 2000 enablement work. A good change management system is important to help you determine whether the problem was introduced because of your changes, or whether it was there even before Year 2000 enablement.

Remember!

You may have to modify the code, build some part of the application, and do some testing as well so be sure to put the charges for these services in the statement of work.

9.2 Application Support

The ending of the warranty period finalizes the Year 2000 enablement process. At this time, the responsibility of managing the application is returned to the customer. Of course, you may choose to offer continuing support to the customer if you determine you want to make it a part of the services your company provides, but it is advisable to outline the provisions for continuing support in a separate contract.

The following list contains some of the elements a warranty or service should offer:

- The change management process through which the problems are evaluated including:
  - Designing forms for formal submission of a change request and their approval.

Change Request Form

Part I - Customer Section

Description of change or problem ecnountered:
(Include details of input to the application)

Problem Encountered		Problem Reported	
Date	Time	Date	Time
_____	_____	_____	_____

Name of the person reporting problem or requesting change:

_____

Figure 5. Change Request Form - Part I

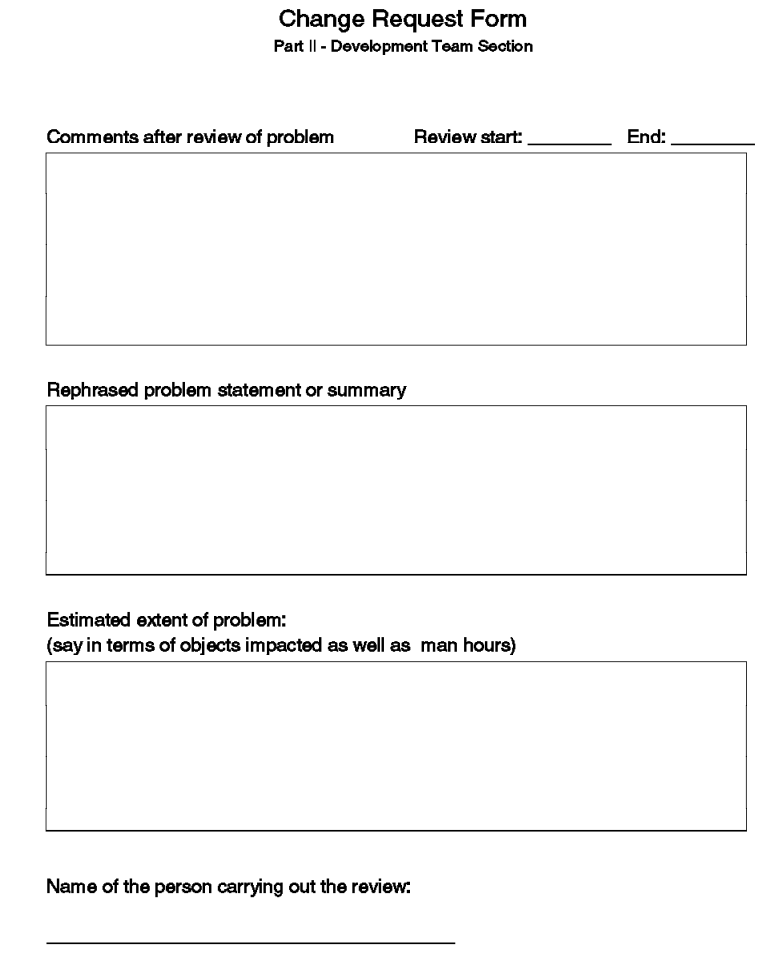


Figure 6. Change Request Form - Part II

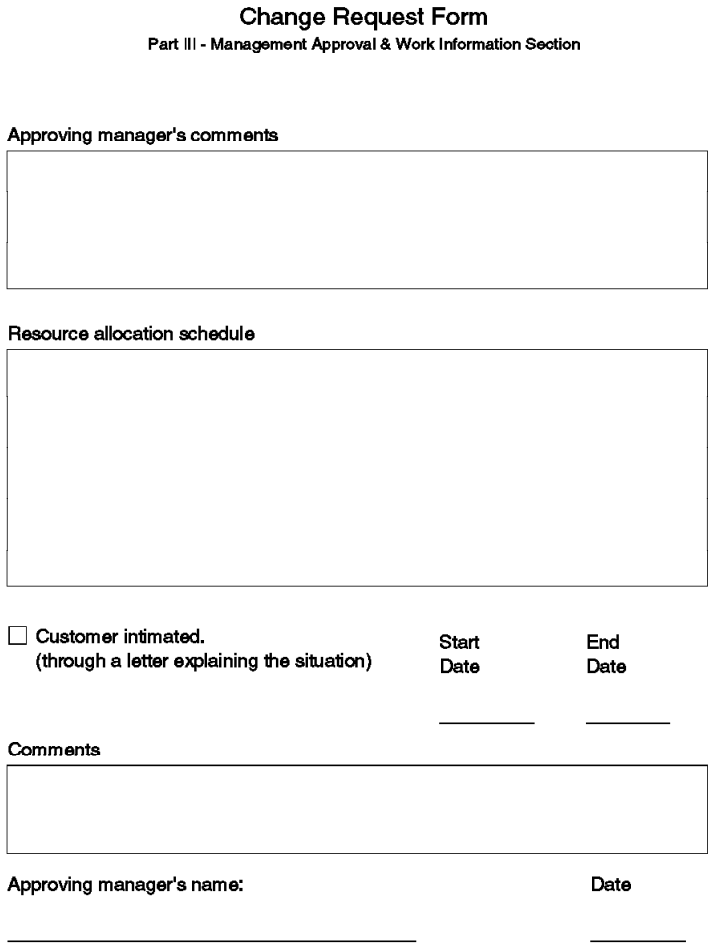


Figure 7. Change Request Form - Part III

  - Define a set of rules to evaluate problems.
  - Determine if it was an operational problem by checking the operator input.
  - If the problem was operational, evaluate the job log.
  - Look at the source code as well as the call stack.
  - Understand and re-create the events that caused the problem to occur.
  - Be sure the problem is not corrupting data as quickly as possible.
- Estimate and allocate resources.
- Include a statement of limitation of scope and liability in the contract. Be sure you have a way of charging for work that is outside the scope of the contract or work that is caused by misinformation from the customer.
- The duration of the service offering depends upon the following factors:
  - Size of the application software
  - Difficulty of making the software Year 2000 enabled
  - Lines of code provided

  - Any special customer requests, such as warranty limit extensions
  - Relationship with your customer
  - Your own preferences in providing such a service
- Resource requirements projected
- Financial proposal
- Exact description of start and end dates of the service

During the support period, the service provider may recommend that the customer modernize the application based on the client/server or network centric model. You may give the customer some kind of an approximate estimate of the resource required and put a separate contract in place for this work. This is a good time to point out current trends in the industry and the limitations of the current application. At a minimum, you may want to suggest that the customer start using ADS/400, ADM/400, and CODE/400 for more rapid development and better manageability of source code.

9.3 Service Offerings Out of Post Implementation Relationship

Since the service provider may be in a long term relationship with the customer (even after the Year 2000 enablement process is over) due to the *application warranty*, the customer might want you to perform other services such as:

- Revalidating and revamping the security system
- Establishing proper backup procedures
- Standardizing the applications
- Functionally enhancing the application software
- Changing to a GUI environment
- Network installation, setup, running, and trouble-shooting
- Performance tuning
- Capacity planning
- Availability enhancements
- Creating a comprehensive testing methodology
- Finding another vendor that can help the customer with problems, or even searching for an alternate solution

There are many types of services that you may not choose to perform. It is good to be aware of other service providers that specialize in these areas so that you can refer the customer to them.

Appendix A. Year 2000 Enablement - The Big Picture

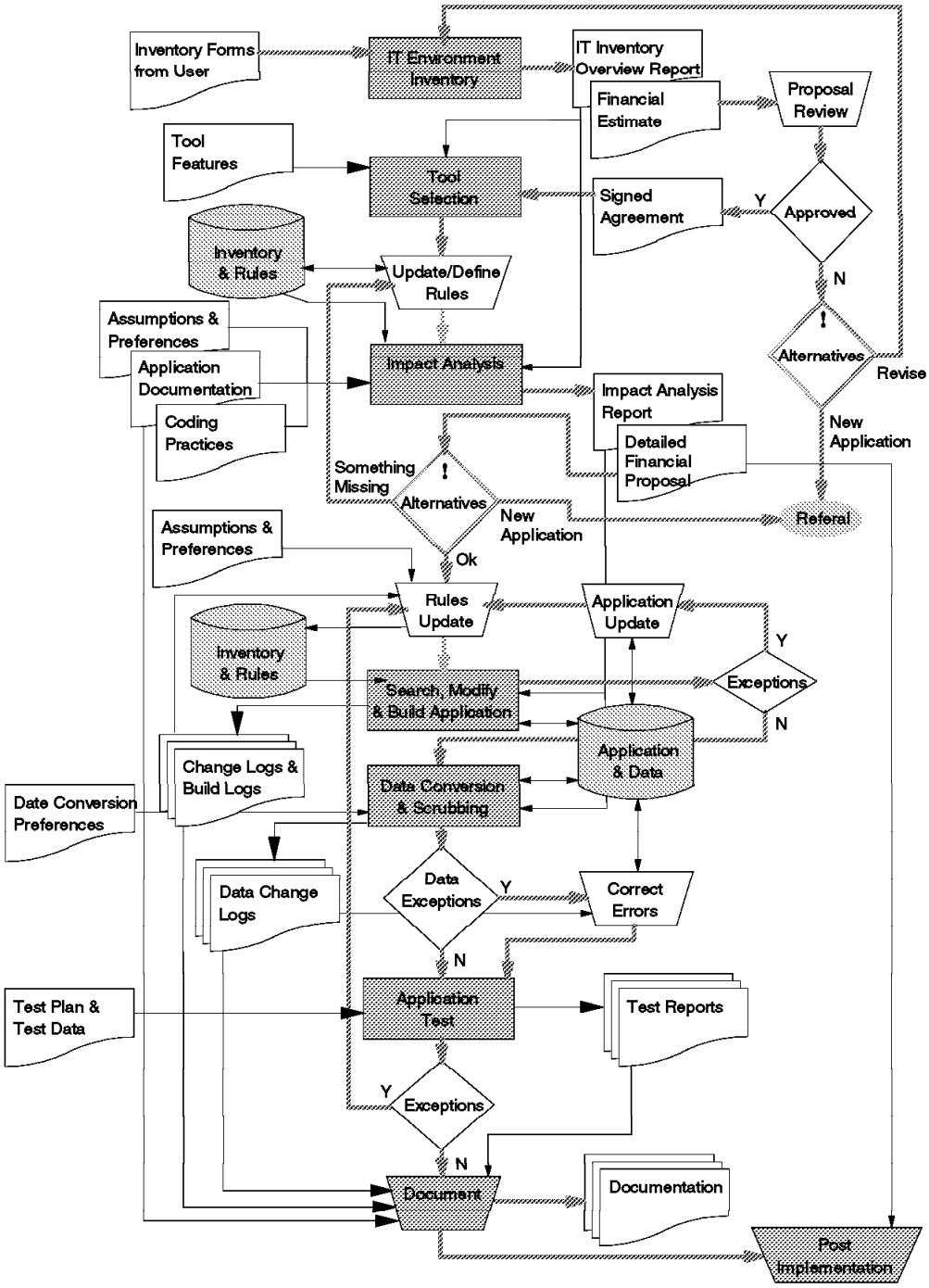


Figure 8. Detailed Diagram of Year 2000 Enablement Process

The diagram shown in Figure 8 gives a complete overview of the entire Year 2000 enablement. While it appears complicated on first viewing, it is more easily digested by firstly looking at the shaded rectangles. These represent the basic process building blocks and there is a chapter for each in this book. The shaded

database symbols represent information that must be kept while performing a Year 2000 enablement project. The remaining symbols mainly represent reports, decision points, and activities.

Appendix B. Application Dictionary Services (ADS/400)

ADS/400 integrates a dictionary database program with the AS/400 application development tools. It stores descriptions of application objects and their relationships in a data dictionary, thereby maintaining a complete inventory of all components. As you make changes to an object, the information in the dictionary is automatically updated.

ADS/400 can help you design and maintain your applications by enabling you to do the following:

- To determine where fields are used in the objects that constitute your application and to determine what files or programs are affected by a change to a given field. This enables you to assess the impact of a change before it is made.
- To ensure that any change to a field is reflected in all objects that refer to that field, and if necessary, to recompile files or programs that are affected by this change.
- To identify certain physical files as field reference files.
- To search for strings in a source member.
- To scan for externally described field names in RPG programs to determine if a change to that field has real impact.
- To determine which programs are calling a specific program or which programs are being called by a specific program.
- To create physical files based on field reference files.
- To build SQL table files based on field reference files and build SQL index files based on the SQL table files and physical files.

An ADS/400 application dictionary documents information about every object in your application′s libraries and saves it in a set of database files. These files are kept up-to-date automatically and serve as a means of navigating through the application. When you create a dictionary, it can document up to 99 libraries depending on their size. Note that the dictionary only contains information describing the objects; the objects themselves remain in the AS/400 libraries in which they are created.

Please refer to *Application Dictionary Services/400 User′s Guide* SC09-1860-00, for the complete write-ups.

Once you have created an application dictionary for your applications, you can search for a particular field/file/program, make changes to the source member, and compile the object and its related objects, all through the PDM-like menus. Or, this application dictionary can be separately used for your own application so that you can write a query program for scan/search purposes.

During an earlier residency, the ADS/400 tool was used to analyze the impact of the date fields in an Order Entry application to size the Year 2000 problem. The following brief write-up shows what ADS/400 was able to contribute to this activity. The first step was to create a dictionary for the application using the create dictionary option.

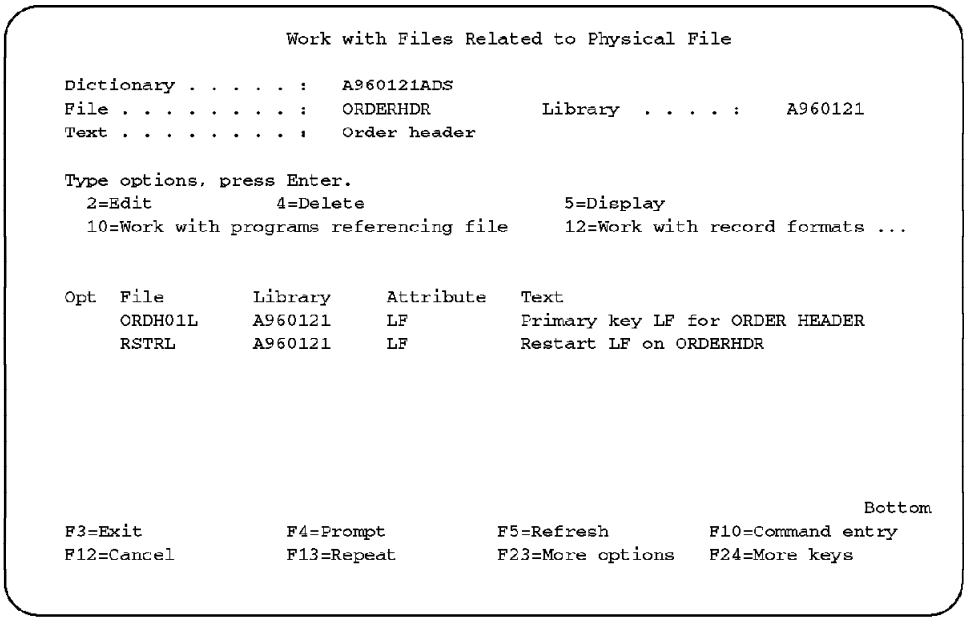


Figure 9. ADS/400 Dictionary Created

The next step was to attempt to search for possible date fields. A simple program was written for this purpose and was linked to option 55 of the ADS Main Menu. The objective of this program was to filter the content of the ADS field reference file to isolate possible date fields. The program also produced a summary report that showed all possible date fields found within the application. The sample report is shown in the following figure:

```
                              Possible Date Fields found using ADS/400
                              ========================================
    match criteria:  numeric field with 2, 3, 4, 6 digits
                    field that contains selected text or substring

                              ========================================
                                                                    Number
                     File        Record      External     Field     of     Field Text
Library    File      Type        Format      Field Name   Type    Digits  Description
---------  --------- ----------  ----------  -----------  ------  ------------------------------
A960121    DAT_DES   PF-DTA      DFTRCD      DTA_DES      A            0  DATA DESCRIPTORS
                                               -----------
             Inspected field for file DAT_DES:          1

           ODET      DSPF        SFL01C      REC1         S            4
             Inspected field for file ODET:             1

           OHRD      DSPF        ORDER       *IN25        A            0
                     DSPF        ORDER       ORHDLY       S            6
                     DSPF        ORDER       ORHDTE       S            6
             Inspected field for file OHRD:             3

           ORDERHDR  PF-DTA      ORDERHDR    ORHDLY       P            6  DELIVERY DATE <----
                     PF-DTA      ORDERHDR    ORHDTA       P            6  ORDER DATE    <----
                                               -----------
             Inspected field for file ORDERHDR:         2

           ORDH01L   LF          ORDERHDR    ORHDLY       P            6  DELIVERY DATE <----
                     LF          ORDERHDR    ORHDTA       P            6  ORDER DATE    <----
                                               -----------
             Inspected field for file ORDH01L:          2

           STOCK     PF-DTA      STOCK       PRDEXP       P            6  EXPIRE DATE   <----
                                               -----------
             Inspected field for file STOCK:            1

           STOCK01L  LF          STOCK       PRDEXP       P            6  EXPIRE DATE   <----
                                               -----------
             Inspected field for file STOCK01L:         1

           STOCK02L  LF          STOCK       PRDEXP       P            6  EXPIRE DATE   <----
                                               -----------
             Inspected field for file STOCK02L:         1
```

Figure 10. ADS/400 Customized Year 2000 Impact Analysis

Using ADS/400 to filter the reference file provided a good start to the analysis of the Year 2000 issue. Should it become necessary to change the size of a field, the report provides some feel for the number of fields and files requiring the change. The next step was to identify the files with the programs. Figure 11 on page 96 and Figure 12 on page 96 show the objects used within programs, physical files, logical files, and libraries.

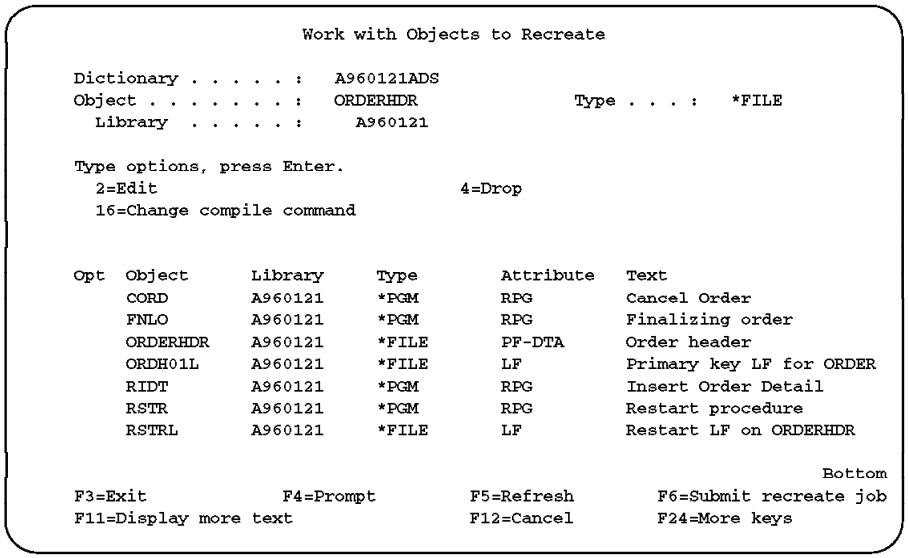


Figure 11. ADS/400 Object Impact Analysis

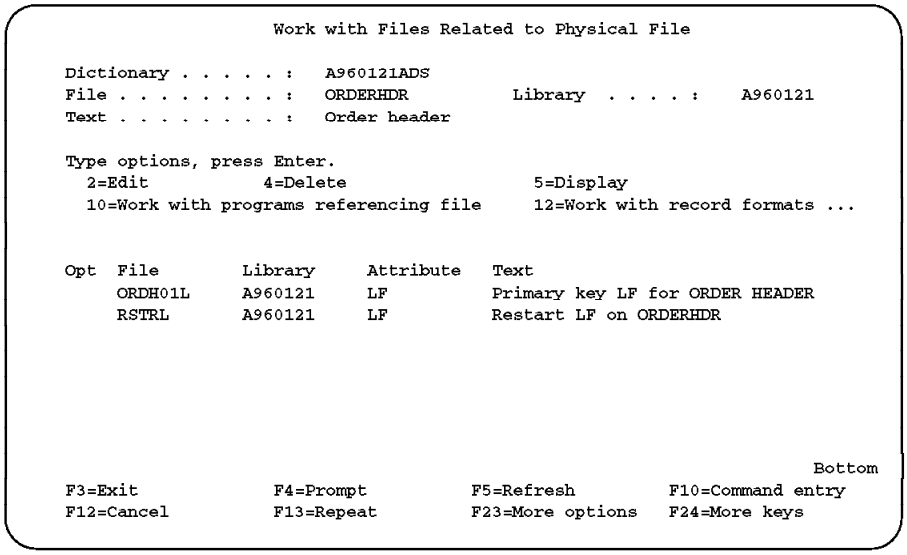


Figure 12. ADS/400 Related File with Date

The two simple steps using ADS/400 paved the way for impact analysis and additional activities towards the Year 2000 conversion effort.

Appendix C. OS/400 Products

This appendix reviews the Year 2000 support, the different releases of OS/400, and the many associated OS/400 Licensed Program Products.

- OS/400 V3R1 is fundamentally Year 2000 safe:
  - Date/Time is internally tracked as milliseconds since August 23, 1928 at 12:03:06.315.
  - Command-to-command processing program (CPP) interface is CYYMMDD.
  - Outfiles are date qualified (CYY or YYYY).
  - APIs are date qualified (CYY ot YYYY).
  - Some fixes are needed to specific routines.
- Most other OS/400 LPs, LPOs, and PRPQs are in a similar situation.
- An OS/400 product list can be found in *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251-02.

:

C.1.1.1 OS/400 V3R1/V3R6 Support

- Date retrieval is currently available:
  - RPG/400 and RPG IV *Year and *DATE four-digit year support
  - RPG II and RPGIII UYEAR and UDATE two-digit year support
  - C library functions (time, ctime, and so on) $-digit support
  - MI date instructions four-digit year support
  - CL and callable API support for two-digit and four-digit year support
  - SQL four-digit year support for RPG/400, RPG IV, COBOL, C, FORTRAN, and PL/1
- Date arithmetic:
  - Lilian/ISO: add and subtract operations of RPG, COBOL, C, and so on
  - Date data type: SQL, OPNQRYF, and RPG IV
- Day of week:
  - ILE Command Execution Environment (CEE) services
- Date format conversions (both two-digit and four-digit formats supported):
  - Convert Date (CVTDAT) CL command
  - Convert Date (QWCCVTDT) API
  - ILE CEE services
- Relative date windows:
  - ILE CEE sliding window services
  - MI window definition services

C.1.1.2 Phased Year 2000 Enhancements

Stage 1 - Base Enablers and Operations:

- Enablers:
  - Cl commands date data type expansion to four-digit year
  - Century system value accessible through API from RPG, COBOL, C, and so on
  - Century job value accessible through API from RPG, COBOL, C, and so on
  - Ability to access Century and YYYY information from SSP 7.5 based applications (OCL, RPG II, COBOL)
  - Expand current 1940 - 2039 fixed window scheme for OS/400
- Consistent operational use of four-digit years:
  - Storing of power-up time
  - Job date on some displays
- Update documentation.
- Enablers available starting with V3R1 and V3R6, operational changes are reflected in releases available by year-end 1996.

C.1.1.3 Current Year 2000 Support Summary

- OS/400:
  - V2R2 and V3Rx - Operationally safe with application enablers available in base release.
  - V3R1 and V3R6 - Application enablers available through PTF, 99% operationally safe but if you use that 1%....
  - Pre-V3R1 - System is not fully operational safe, limited enablers are available (no ILE RPG, COBOL, CL; no unambiguous system date, and so on).
- SSP:
  - Advanced 36 Model 436 - Operationally safe with documented limitations (Value-Added Software Package (VASP)), application enablers for century and four-digit year currently available through PTF (C6107075).
  - Advance 36 Model 236 - Operationally safe documented limitations (VASP), application enablers for century and four-digit year to be available 2Q1996.
  - S/36 - Operationally safe with documented limitations (VASP).
  - S/34 - Not known, no plans to test.
- CPF:
  - S/38 - Not operationally safe, only limited enablers are available.

C.1.1.4 Work in Process for Year 2000

- DDS enhancements:
  - Date, time, and time-stamp data types in DDS:
    - Display file (*DSPF)
    - Printer file (*PRTF)
  - Edit code support of four-digit years
  - Special values for four-digit years

Appendix D. IBM System Features for AS/400 System

You can use the IBM products and tools in the following list to help you address the changes required as you proceed with your Year 2000 transition. The tools listed, however, represent only a subset of the AS/400 oriented offerings available to you for application development. A more complete list of available tools can be found in the *AS/400 Application Development Handbook*, G325-6249.

D.1 The IBM RPG Family

For application code written in RPG, IBM has developed several RPG language compilers. These compilers target the host application development environment.

IBM's RPG compilers for the AS/400 system are:

- Integrated Language Environment (ILE) RPG IV
- Original Program Model (OPM) RPG/400
- System/36 compatible compiler

All three RPG language compilers are available with either the Integrated Language Environment RPG for OS/400 Version 3 product (5716-RG1) or the IBM Integrated Language Environment RPG/400 Version 3 product (5763-RG1).

Using ILE RPG IV Date Support: The ILE RPG IV compiler has the following support for dates:

- Retrieval of the job date through special words UDATE and UYEAR. This level of support provides for a two-digit year.
- Retrieval of the job date through special words *DATE and *YEAR. This level of support provides for a four-digit year.
- Retrieval of the system date through operation TIME. This level of support provides for both two-digit and four-digit years.
- Date, time, and time-stamp data type support. This level of support provides for date operations such as subtraction, addition, extraction, testing, comparison, and move.
- Calling OPM System APIs. This level of support provides for both two-digit and four-digit years.
- Calling ILE Date APIs. This level of support provides for both two-digit and four-digit years.
- SQL date, time, and time-stamp data type support. This level of support provides for four-digit years.

This section provides information on how to use RPG date/time fields. Specifically, it covers:

- The date/time data types and their formats
- User date special words
- How to edit date-time fields
- Date-time keywords
- Date-time operations and how to use them

D.2 Date-Time Data Types and Formats

Date, time, and time-stamp fields have an internal format that is independent of the external format. The **internal format** is the way the data is stored in the program. The **external format** is the way the data is stored in files.

You need to be aware of the internal format when you are:

- Passing parameters
- Overlaying subfields in data structures

There is a default internal format for the date and time fields. In general, to change the internal format for a specific field, you must define the field and specify its internal format on a definition specification. Similarly, to change (or specify) the external format for program-described fields, you specify the format on the corresponding input or output specification.

For fields in an externally described file, the external data format is specified in the data description specifications in position 35. You cannot change the external format of externally described date and time fields.

For subfields in externally described data structures, the data formats specified in the external description are used as the internal formats of the subfields by the compiler. The reason for this difference is that a data structure, even if externally described, exists only when a program is running.

Internal Format: The default format for date/time fields is *ISO. In general, it is recommended that you use the default ISO internal format, especially if you have a mixture of external format types.

For date/time fields, you can change the default format in two ways. You can use the DATFMT and TIMFMT keywords on the Control specification to change the default internal format, if desired, for *all* of the date and time fields in the program. In addition, you can use the definition specification to:

- Override the default internal format by using the DATFMT and TIMFMT keywords.
- Specify an initial value for a date, time, or time-stamp field that is different than the default by using the INZ keyword.

Specifying an External Format for a Date or Time Field: If you have date or time fields in program-described files, you *must* specify their external format. You can specify a default external format for all date or time fields in a program-described file by using the DATFMT and TIMFMT keywords on a file description specification. You can specify an external format for a particular field as well. Specify the desired format in position 31 to position 34 on an input specification. Specify the appropriate keyword and format in position 53 to position 80 on an output specification.

For more information on each format type, see the appropriate section in the remainder of this chapter.

Date Data Type: Date fields have a predetermined size and format. They can be defined on the definition specification. Leading and trailing zeros are required for all date data.

Date constants or variables used in comparisons or assignments do not have to be in the same format or use the same separators. Also, dates used for I/O operations such as input fields, output fields, or key fields are also converted (if required) to the necessary format for the operation.

The default internal format for date variables is *ISO. This default internal format can be overridden globally by the control specification keyword DATFMT and individually by the definition specification keyword DATFMT.

The hierarchy used when determining the internal date format and separator for a date field is:

1. From the DATFMT keyword specified on the definition specification
2. From the DATFMT keyword specified on the control specification
3. *ISO

There are two kinds of date data formats: two-digit and four-digit year formats. For two-digit year formats, years in the range 1940 to 2039 can be represented. This leads to the possibility of a date overflow condition occurring when converting from a four-digit year format to a two-digit year format.

The following table lists the formats for date data:

Table 1. Date Formats for Date Data Type

Format Name	Description	Format	Length	Example
*MDY	Month/Day/Year	MM/DD/YY	8	01/15/91
*DMY	Day/Month/Year	DD/MM/YY	8	15/01/91
*YMD	Year/Month/Day	YY/MM/DD	8	91/01/15
*JUL	Julian	YY/DDD	6	91/015
*ISO	International Standards Organization	YYYY-MM-DD	10	1991-01-15
*USA	IBM USA Standard	MM/DD/YYYY	10	01/15/1991
*EUR	IBM European Standard	DD.MM.YYYY	10	15.01.1991
*JIS	Japanese Industrial Standard Christian Era	YYYY-MM-DD	10	1991-01-15

The following table lists the *LOVAL, *HIVAL, and default values for all of the date formats.

Table 2. Date Values				
Format Name	**Description**	***LOVAL**	***HIVAL**	**Default Value**
*MDY	Month/Day/Year	01/01/40	12/31/39	01/01/01
*DMY	Day/Month/Year	01/01/40	31/12/39	01/01/01
*YMD	Year/Month/Day	40/01/01	39/12/31	01/01/01
*JUL	Julian	40/001	39/365	01/001
*ISO	International Standards Organization	0001-01-01	9999-12-31	0001-01-01
*USA	IBM USA Standard	01/01/0001	12/31/9999	01/01/0001
*EUR	IBM European Standard	01.01.0001	31.12.9999	01.01.0001
*JIS	Japanese Industrial Standard Christian Era	0001-01-01	9999-12-31	0001-01-01

Time Data Type: Time fields have a predetermined size and format. They can be defined on the definition specification. Leading and trailing zeros are required for all time data.

Time constants or variables used in comparisons or assignments do not have to be in the same format or use the same separators. Also, times used for I/O operations such as input fields, output fields, or key fields are also converted (if required) to the necessary format for the operation.

The default internal format for time variables is *ISO. This default internal format can be overridden globally by the control specification keyword TIMFMT and individually by the definition specification keyword TIMFMT.

The hierarchy used when determining the internal time format and separator for a time field is:

1. From the TIMFMT keyword specified on the definition specification
2. From the TIMFMT keyword specified on the control specification
3. *ISO

The following table lists the formats for time data.

Table 3. Time Formats for Time Data Type				
Format Name	**Description**	**Format**	**Length**	**Example**
*HMS	Hours:Minutes:Seconds	HH:MM:SS	8	14:00:00
*ISO	International Standards Organization	HH.MM.SS	8	14.00.00
*USA	IBM USA Standard. AM and PM can be any mix of upper and lower case.	HH:MM AM or HH:MM PM	8	02:00 PM
*EUR	IBM European Standard	HH.MM.SS	8	14.00.00
*JIS	Japanese Industrial Standard Christian Era	HH:MM:SS	8	14:00:00

The following table lists the *LOVAL, *HIVAL, and default values for all of the time formats.

Table 4. Time Values				
Format Name	**Description**	***LOVAL**	***HIVAL**	**Default Value**
*HMS	Hours:Minutes:Seconds	00:00:00	24:00:00	00:00:00
*ISO	International Standards Organization	00.00.00	24.00.00	00.00.00
*USA	IBM USA Standard. AM and PM can be any mix of upper and lower case.	00:00 AM	12:00 AM	00:00 AM
*EUR	IBM European Standard	00.00.00	24.00.00	00.00.00
*JIS	Japanese Industrial Standard Christian Era	00:00:00	24:00:00	00:00:00

D.2.1 Time-Stamp Data Type

Time-stamp fields have a predetermined size and format. They can be defined on the definition specification. Time-stamp data must be in the format:

`yyyy-mm-dd-hh.mm.ss.mmmmmm (length 26).`

Microseconds (.mmmmmm) are optional for time-stamp literals and if not provided, are padded on the right with zeros. Leading zeros are required for all time-stamp data.

The default initialization value for a time stamp is midnight of January 1, 0001 (0001-01-01-00.00.00.000000). The *HIVAL value for a time stamp is 9999-12-31-24.00.00.000000. Similarly, the *LOVAL value for time stamp is 0001-01-01-00.00.00.000000.

D.3 User Date Special Words

The user date special words (UDATE, *DATE, UMONTH, *MONTH, UDAY, *DAY, UYEAR, and *YEAR) allow the programmer to supply a date for the program at run time. The user date special words access the job date that is specified in the job description. The user dates can be written out at output time; UDATE and *DATE can be written out using the Y edit code in the format specified by the control specification. (For a description of the job date, see *AS/400 Work Management*, SC41-4306.)

Rules for User Date: Remember the following rules when using the user date:

- UDATE, when specified in position 30 through position 43 of the output specifications, prints a 6-character numeric date field. *DATE, when similarly specified, prints an 8-character (four-digit year portion) numeric date field. These special words can be used in three different date formats:

  Month/day/year
  Year/month/day
  Day/month/year

  Use the DATEDIT keyword on the control specification to specify the editing to be done. If this keyword is not specified, the default is *MDY.

- For an interactive program, the user date special words are set when the job starts running. For a batch program, they are set when the job is sent to the job queue. In neither case are they updated when the program runs over midnight or when the job date changes. Use the TIME operation code to obtain the time and date while the program is running.

- UMONTH, *MONTH, UDAY, *DAY, and UYEAR, when specified in position 30 through position 43 of the output specifications, print a 2-position numeric date field. *YEAR can be used to print a 4-position numeric date field. Use UMONTH or *MONTH to print the month only, UDAY or *DAY to print the day only, and UYEAR or *YEAR to print the year only.
- UDATE and *DATE can be edited when they are written if the Y edit code is specified in position 44 of the output specifications. The DATEDIT keyword on the control specification determines the format and the separator character to be inserted (for example, 12/31/88, 31.12.88., and 12/31/1988).
- UMONTH, *MONTH, UDAY, *DAY, UYEAR, and *YEAR cannot be edited by the Y edit code in position 44 of the output specifications.
- The user date fields cannot be modified. This means they cannot be used:
  - In the result field of calculations
  - As factor 1 of PARM operations
  - As the factor 2 index of LOOKUP operations
  - With blank after in output specifications
  - As input fields
- The user date special words can be used in factor 1 or factor 2 of the calculation specifications for operation codes that use numeric fields.
- User date fields are not date data type fields but are numeric fields.

D.4 Editing Date Fields

The Y edit code is normally used to edit a three-digit to nine-digit date field. It suppresses the leftmost zeros of date fields up to but not including the digit preceding the first separator. Slashes are inserted to separate the day, month, and year. The DATEDIT and DECEDIT keywords on the control specification can be used to alter edit formats.

Note: The Y edit code is not valid for *YEAR, *MONTH, and *DAY.

The Z edit code removes the sign (plus or minus) from and suppresses the leading zeros of a numeric field. The decimal point is not placed in the field and is not printed.

The Y edit code suppresses the leftmost zeros of date fields up to but not including the digit preceding the first separator. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

```
   nn/n
   nn/nn
   nn/nn/n
   nn/nn/nn
  nnn/nn/nn
   nn/nn/nnnn  Format used with M, D or blank in position 19
  nnn/nn/nnnn  Format used with M, D or blank in position 19
 nnnn/nn/nn    Format used with Y in position 19
nnnnn/nn/nn    Format used with Y in position 19
```

D.5 Date Operations

Date operations allow you to perform date and time arithmetic, extract portions of a date, time, or time-stamp field, or test for valid fields. They operate on date, time, and time-stamp fields and character and numeric fields representing dates, times, and time stamps. The date operations are:

- ADDDUR
- EXTRACT
- SUBDUR
- TEST

With ADDDUR, you can add a duration to a date or time. With SUBDUR, you can subtract a duration from a date or time, or calculate the duration between two dates, times, or time stamps. With EXTRACT, you can extract part of a date, time, or time stamp. With TEST, you can test for a valid date, time, or time-stamp field. The valid duration codes (and their short forms) are:

- *YEARS for the year (*Y)
- *MONTHS for the month (*M)
- *DAYS for the day (*D)
- *HOURS for the hours (*H)
- *MINUTES for the minutes (*MN)
- *SECONDS for the seconds (*S)
- *MSECONDS for the microseconds (*MS).

Adding or Subtracting Dates: When adding (or subtracting) a duration in months to (or from) a date, the general rule is that the month portion is increased (or decreased) by the number of months in the duration, and the day portion is unchanged. The exception to this is when the resulting day portion exceeds the actual number of days in the resulting month. In this case, the resulting day portion is adjusted to the actual month end date.

For example, adding one month to ″95/05/30″ (*YMD format) results in ″95/06/30″, as expected. The resulting month portion has been increased by 1; the day portion is unchanged. On the other hand, adding one month to ″95/05/31″ results in ″95/06/30″. The resulting month portion has been increased by 1 and the resulting day portion has been adjusted because June has only 30 days.

Subtracting one month from ″95/03/30″ yields ″95/02/28″. In this case, the resulting month portion is decreased by 1 and the resulting day portion adjusted because February has only 28 days (in non-leap years).

Similar results occur when adding or subtracting a year duration. For example, adding one year to ″92/02/29″ results in ″93/02/28″, an adjusted value since the resulting year is not a leap year.

Calculating Durations Between Dates: The SUBDUR operation can be used to calculate a duration by subtracting two dates, times, or time stamps. The result of the calculation is a complete unit; any rounding that is done is downwards. The calculation of durations includes microseconds.

For example, if the actual duration is 384 days, and the result is requested in years, the result is one complete year because there are 1.05 years in 384 days. A duration of 59 minutes requested in hours results in 0 hours. Here are some additional examples.

```
Duration in    between                    and                        is
============  ==========           ============              ===========
 Months         1994-02-28           1994-03-28                 1 month
                1994-03-15           1995-03-14                11 months
                1994-03-15           1995-03-15                12 months

 Years          1994-03-15           1995-03-14                 0 years
                1994-03-31           1995-03-31                 1 year

         1970-03-14-23.00.00.000000  1970-03-14-22.00.00.000001  0 years

 Hours  1990-03-14-12.34.45.123456  1989-03-14-12.34.45.123457  0 years
```

Unexpected Results: If adjustment takes place on a date or time addition or subtraction, a subsequent duration calculation probably results in a different duration than the one originally added or subtracted. This is because the calculated duration no longer contains a complete unit, and so, rounding down, yields one unit less than expected. This is shown in example 1 and example 2.

A second unexpected result is shown in example 3 and example 4. Different initial dates give the same result after adding one month. When subtracting one month from the result, it is impossible to arrive at both initial dates.

1. ″95/05/31″ ADDDUR 1:*MONTH gives ″95/06/30″
   ″95/06/30″ SUBDUR ″95/05/31″ gives 0 months

You might expect the result of the SUBDUR to be one month.

2. ″95/06/30″ ADDDUR 1:*MONTH gives ″95/07/30″
   ″95/07/30″ SUBDUR ″95/06/30″ gives 1 month

This is the ″expected″ result.

3. ″95/01/31″ ADDDUR 1:*MONTH gives ″95/02/28″
   ″95/01/28″ ADDDUR 1:*MONTH gives ″95/02/28″

Two different dates yield the same date due to adjustment.

   ″95/02/28″ SUBDUR 1:*MONTH gives ″95/01/28″

″Reversing″ the addition does not result in the original dates.

D.6 Using OPM RPG/400 Date Support

The OPM RPG 400 compiler has the following support for dates:

- Retrieval of the job date through special words UDATE and UYEAR. This level of support provides for a two-digit year.
- Retrieval of the job date through special words *DATE and *YEAR. This level of support provides for a four-digit year.
- Retrieval of the system date through operation TIME. This level of support provides for both two-digit and four-digit years.
- Calling OPM System APIs. This level of support provides for both two-digit and four-digit years.
- SQL date, time, and time-stamp data type support. This level of support provides for four-digit years.

D.7 Using System/36 Compatible Date Support

The System/36 compatible compiler has the following support for dates:

- Retrieval of the job date through special words UDATE and UYEAR. This level of support provides for a two-digit year.
- Retrieval of the system date through operation TIME. This level of support provides for a two-digit year.
- Calling OPM System APIs. This level of support provides for both two-digit and four-digit years.

D.8 The IBM COBOL Family for AS/400 System

For application code written in COBOL, IBM has developed several COBOL language compilers. These compilers target the host application development environment.

The IBM COBOL compilers for the AS/400 system are:

- Integrated Language Environment (ILE) COBOL/400
- Original Program Model (OPM) COBOL/400
- System/36 compatible COBOL compiler

All three COBOL language compilers are available with either the Integrated Language Environment COBOL for OS/400 Version 3 product (5716-CB1) or the IBM Integrated Language Environment COBOL/400 Version 3 product (5763-CB1).

D.8.1.1 Using COBOL Date Support

The COBOL ACCEPT statement is supported by the three compilers previously listed. This level of support provides for a two-digit year when referencing a DATE or DAY.

Calling OPM System APIs is supported by the three compilers previously listed. This level of support provides for both two-digit and four-digit years.

The COBOL WHEN-COMPILED special register is supported in both the OPM and ILE COBOL/400 compilers. This level of support provides for a two-digit year.

Calling ILE Date APIs is supported by the ILE COBOL/400 compiler. This level of support provides for both two-digit and four-digit years.

SQL is supported by both the OPM and ILE COBOL compilers. This level of support provides for four-digit years.

D.9 The IBM C Family for AS/400 System

For application code based on C, IBM has developed language compilers for both C and C++.

The IBM C-based compilers for the AS/400 system are:

- Integrated Language Environment (ILE) C for OS/400
- VisualAge C++ for OS/400

D.9.1 Using C and C++ Date Support

The C library, available to both compilers, provides many functions related to date and time retrieval, manipulation, and formatting. This level of support provides for four-digit years.

Calling OPM System APIs is supported by both compilers. This level of support provides for both two-digit and four-digit years.

Calling ILE Date APIs is supported by both compilers. This level of support provides for both two-digit and four-digit years.

SQL is supported by the ILE C for OS/400 compiler. This level of support provides for four-digit years.

D.10 Integrated Language Environment for OS/400

Integrated Language Environment (ILE) for OS/400 is IBM′s common run-time environment for enterprise applications written in the ILE languages of RPG, COBOL, C, and CL. ILE is designed to provide defined calling conventions, enhanced inter-language communication, and callable services in areas such as date and time, math, storage management, and exception handling. ILE services are standard in OS/400 starting with Version 2 Release 3.

ILE provides a number of date functions that include:

- Ability to parse dates in an infinite number of formats by using a picture string as a parsing guide.
- Retrieve current date.
- Convert a date character string to a Lilian (integer) format, thereby enabling easy date arithmetic operations.
- Convert a Lilian date to a date character string.
- Century sliding window.

The ILE century sliding window technique may provide a short term solution for some applications. If you are unable to change all of your applications and data at the same time, the century window allows two-digit years to be interpreted in a 100-year window. Any 100-year window can be selected. You pass a two-digit year to ILE and ILE returns a four-digit year based on the 100-year window.

The advantage to the century window is that you need to change only the application code and not the databases with two-digit years. This allows you to change the application programs one at a time or groups at a time without affecting your databases. Note, this only works for dates that range less than 100 years. For example, dates of birth may not be appropriate for this solution.

D.11 DB2/400 SQL

DB2/400, a standard part of OS/400, provides the run-time support for SQL on the AS/400 system. SQL provides support for the database data types of date, time, and time stamp and operations such as add, subtract, assignment, and compare. The DB2 Query Manager and SQL Development Kit product (5716-ST1 or 5763-ST1) provides SQL precompilers for AS/400 programming languages such as RPG, C, and COBOL. SQL support provides for four-digit year support.

D.11.1 Using Date, Time, and Time Stamp Support

Date, time, and time stamp are data types represented in an internal form not seen by the SQL user. Date, time, and time stamp can be represented by character string values and assigned to character string variables. The database manager recognizes the following values as date, time, and time-stamp values:

- A value returned by the DATE, TIME, or TIMESTAMP scalar functions
- A value returned by the CURRENT DATE, CURRENT TIME, or CURRENT TIMESTAMP special registers
- A character string when it is an operand of an arithmetic expression or a comparison *and*; the other operand is a date, time, or time stamp. For example, in the predicate:

  ```
  ... WHERE HIREDATE < '1950-01-01'
  ```

  If HIREDATE is a date column, the character string '1950-01-01' is interpreted as a date.
- A character string variable or constant used to set a date, time, or time stamp column in either the SET clause of an UPDATE statement, or the VALUES clause of an INSERT statement

For more information on character string formats of date, time, and time-stamp values, refer to *DB2/400 SQL Reference*, SC41-3612.

Specifying Current Date and Time Values: You can specify a current date, time, or time stamp in an expression by specifying one of three special registers: CURRENT DATE, CURRENT TIME, or CURRENT TIME STAMP. The value of each is based on a time-of-day clock reading obtained during the running of the statement. Multiple references to CURRENT DATE, CURRENT TIME, or CURRENT TIME STAMP within the same SQL statement use the same value. The following statement returns the age (in years) of each employee in the EMPLOYEE table when the statement is run:

```
SELECT YEAR(CURRENT DATE - BIRTHDATE)
FROM CORPDATA.EMPLOYEE
```

The CURRENT TIMEZONE special register allows a local time to be converted to Universal Coordinated Time. For example, if you have a table named DATETIME containing a time column type with a name of STARTT, and you want to convert STARTT to Universal Coordinated Time, you can use the following statement:

```
SELECT STARTT - CURRENT TIMEZONE
FROM DATETIME
```

Date and Time Operands and Durations: Date and time values can be incremented, decremented, and subtracted. These operations may involve decimal numbers called *durations*. A *duration* is a positive or negative number representing an interval of time. There are four types of durations:

Labeled durations

A *labeled duration* represents a specific unit of time as expressed by a number (which can be the result of an expression) followed by one of the seven duration keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES,

SECONDS, or MICROSECONDS.[4] The number specified is converted as if it were assigned to a DECIMAL(15,0) number. A labeled duration can only be used as an operand of an arithmetic operator in which the other operand is a value of data type DATE, TIME, or TIME STAMP. Thus, the expression HIREDATE + 2 MONTHS + 14 DAYS is valid whereas the expression HIREDATE + (2 MONTHS + 14 DAYS) is not. In both of these expressions, the labeled durations are 2 MONTHS and 14 DAYS.

Date Duration

A *date duration* represents a number of years, months, and days expressed as a DECIMAL(8,0) number. To be properly interpreted, the number must have the format *yyyymmdd* where *yyyy* represents the number of years, *mm* the number of months, and *dd* the number of days. The result of subtracting one date value from another, as in the expression HIREDATE - BRTHDATE, is a date duration.

Time Duration

A *time duration* represents a number of hours, minutes, and seconds expressed as a DECIMAL(6,0) number. To be properly interpreted, the number must have the format *hhmmss* where *hh* represents the number of hours, *mm* the number of minutes, and *ss* the number of seconds. The result of subtracting one time value from another is a time duration.

Time-stamp duration

A *time-stamp duration* represents a number of years, months, days, hours, minutes, seconds, and microseconds expressed as a DECIMAL(20,6) number. To be properly interpreted, the number must have the format *yyyymmddhhmmsszzzzzz* where *yyyy, mm, dd, hh, mm, ss,* and *zzzzzz* represent, respectively, the number of years, months, days, hours, minutes, seconds, and microseconds. The result of subtracting one time-stamp value from another is a time-stamp duration.

Date and Time Arithmetic in SQL: The only arithmetic operations that can be performed on date and time values are addition and subtraction. If a date or time value is the operand of addition, the other operand must be a duration. The specific rules governing the use of the addition operator with date and time values follow:

- If one operand is a date, the other operand must be a date duration or labeled duration of years, months, or days.
- If one operand is a time, the other operand must be a time duration or a labeled duration of hours, minutes, or seconds.
- If one operand is a time stamp, the other operand must be a duration. Any type of duration is valid.
- Neither operand of the addition operator can be a parameter marker.

The rules for the use of the subtraction operator on date and time values are not the same as those for addition because a date or time value cannot be subtracted from a duration, and because the operation of subtracting two date or time values is not the same as the operation of subtracting a duration from a date or time value. The specific rules governing the use of the subtraction operator with date and time values follow:

---

[4] Note that the singular form of these keywords is also acceptable: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MICROSECOND.

- If the first operand is a date, the second operand must be a date, a date duration, a string representation of a date, or a labeled duration of years, months, or days.
- If the second operand is a date, the first operand must be a date or a string representation of a date.
- If the first operand is a time, the second operand must be a time, a time duration, a string representation of a time, or a labeled duration of hours, minutes, or seconds.
- If the second operand is a time, the first operand must be a time or string representation of a time.
- If the first operand is a time stamp, the second operand must be a time stamp, a string representation of a time stamp, or a duration.
- If the second operand is a time stamp, the first operand must be a time stamp or a string representation of a time stamp.
- Neither operand of the subtraction operator can be a parameter marker.

Date Arithmetic: Dates can be subtracted, incremented, or decremented.

Subtracting Dates: The result of subtracting one date (DATE2) from another (DATE1) is a date duration that specifies the number of years, months, and days between the two dates. The data type of the result is DECIMAL(8,0). If DATE1 is greater than or equal to DATE2, DATE2 is subtracted from DATE1. If DATE1 is less than DATE2, however, DATE1 is subtracted from DATE2, and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = DATE1 - DATE2.

```
If DAY(DATE2) <= DAY(DATE1)
   then DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

If DAY(DATE2) > DAY(DATE1)
   then DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
     where N = the last day of MONTH(DATE2).
   MONTH(DATE2) is then incremented by 1.

If MONTH(DATE2) <= MONTH(DATE1)
   then MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2).

If MONTH(DATE2) > MONTH(DATE1)
   then MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2).
   YEAR(DATE2) is then incremented by 1.

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

For example, the result of DATE('3/15/2000') - DATE('12/31/1999') is 215 (or, a duration of 0 years, 2 months, and 15 days).

Incrementing and Decrementing Dates: The result of adding a duration to a date or of subtracting a duration from a date is itself a date. (For the purposes of this operation, a month denotes the equivalent of a calendar page. Adding months to a date is the same as turning the pages of a calendar, starting with the page on which the date appears.) The result must fall between the dates January 1, 0001 and December 31, 9999 inclusive. If a duration of years is added or subtracted, only the year portion of the date is affected. The month is unchanged, as is the day unless the result is February 29 of a non-leap-year. In

this case, the day is changed to 28, and SQLWARN6 in the SQLCA is set to ″W″ to indicate the end-of-month adjustment.

Similarly, if a duration of months is added or subtracted, only months and, if necessary, years are affected. The day portion of the date is unchanged unless the result is invalid (September 31, for example). In this case, the day is set to the last day of the month, and SQLWARN6 in the SQLCA is set to ″W″ to indicate the end-of-month adjustment.

Adding or subtracting a duration of days affects the day portion of the date, and potentially the month and year. Adding a labeled duration of DAYS does not cause an end-of-month adjustment.

Date durations, whether positive or negative, may also be added to and subtracted from dates. As with labeled durations, the result is a valid date, and a warning indicator is set in the SQLCA whenever an end-of-month adjustment is necessary.

When a positive date duration is added to a date, or a negative date duration is subtracted from a date, the date is incremented by the specified number of years, months, and days, in that order. Thus DATE1 + X, where X is a positive DECIMAL(8,0) number, is equivalent to the expression:

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS
```

When a positive date duration is subtracted from a date, or a negative date duration is added to a date, the date is decremented by the specified number of days, months, and years, in that order. Thus, DATE1 - X, where X is a positive DECIMAL(8,0) number, is equivalent to the expression:

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS
```

When adding durations to dates, adding one month to a given date gives the same date one month later *unless* that date does not exist in the later month. In that case, the date is set to that of the last day of the later month. For example, January 28 plus one month gives February 28; and one month added to January 29, 30, or 31 results in either February 28 or, for a leap year, February 29.

Note: If one or more months is added to a given date and the same number of months is subtracted from the result, the final date is not necessarily the same as the original date.

Time Arithmetic: Times can be subtracted, incremented, or decremented.

Subtracting Times: The result of subtracting one time (TIME2) from another (TIME1) is a time duration that specifies the number of hours, minutes, and seconds between the two times. The data type of the result is DECIMAL(6,0). If TIME1 is greater than or equal to TIME2, TIME2 is subtracted from TIME1. If TIME1 is less than TIME2, however, TIME1 is subtracted from TIME2, and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = TIME1 - TIME2.

If SECOND(TIME2) <= SECOND(TIME1)
    then SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2).

If SECOND(TIME2) > SECOND(TIME1)
    then SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2).
    MINUTE(TIME2) is then incremented by 1.

If MINUTE(TIME2) <= MINUTE(TIME1)
then MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2).

If MINUTE(TIME2) > MINUTE(TIME1)
then MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2).
HOUR(TIME2) is then incremented by 1.

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).

For example, the result of TIME('11:02:26') - TIME('00:32:56') is 102930 (a duration of 10 hours, 29 minutes, and 30 seconds).

Incrementing and Decrementing Times: The result of adding a duration to a time or of subtracting a duration from a time is itself a time. Any overflow or underflow of hours is discarded, thereby ensuring that the result is always a time. If a duration of hours is added or subtracted, only the hours portion of the time is affected. The minutes and seconds are unchanged.

Similarly, if a duration of minutes is added or subtracted, only minutes and, if necessary, hours are affected. The seconds portion of the time is unchanged.

Adding or subtracting a duration of seconds affects the seconds portion of the time, and potentially the minutes and hours.

Time durations, whether positive or negative, also can be added to and subtracted from times. The result is a time that has been incremented or decremented by the specified number of hours, minutes, and seconds, in that order. TIME1 + X, where "X" is a DECIMAL(6,0) number, is equivalent to the expression:

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

Time Stamp Arithmetic: Time stamps can be subtracted, incremented, or decremented.

Subtracting Time Stamps: The result of subtracting one time stamp (TS2) from another (TS1) is a time-stamp duration that specifies the number of years, months, days, hours, minutes, seconds, and microseconds between the two time stamps. The data type of the result is DECIMAL(20,6). If TS1 is greater than or equal to TS2, TS2 is subtracted from TS1. If TS1 is less than TS2, however, TS1 is subtracted from TS2 and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = TS1 - TS2.

If MICROSECOND(TS2) <= MICROSECOND(TS1)
then MICROSECOND(RESULT) = MICROSECOND(TS1) -
MICROSECOND(TS2).

If MICROSECOND(TS2) > MICROSECOND(TS1)
then MICROSECOND(RESULT) = 1000000 +
MICROSECOND(TS1) - MICROSECOND(TS2)
and SECOND(TS2) is incremented by 1.

The seconds and minutes part of the time stamps are subtracted as specified in the rules for subtracting times.

If HOUR(TS2) <= HOUR(TS1)
then HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).

If HOUR(TS2) > HOUR(TS1)

then HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
and DAY(TS2) is incremented by 1.

The date part of the time stamps is subtracted as specified in the rules for subtracting dates.

Incrementing and Decrementing Time Stamps: The result of adding a duration to a time stamp or of subtracting a duration from a time stamp is itself a time stamp. Date and time arithmetic is performed as previously defined, except that an overflow or underflow of hours is carried into the date part of the result, which must be within the range of valid dates. Microseconds overflow into seconds.

D.12 OS/400 CL Commands

The following sections show the Year 2000 four-digit date support for selected OS/400 CL Commands.

D.12.1 CVTDAT (Convert Date)

The Convert Date (CVTDAT) command converts a date value from one format to another without changing its value. CVTDAT provides for both two-digit and four-digit year formats.

D.12.2 OPNQRYF (Open Query File)

The Open Query File (OPNQRYF) command opens a file that contains a set of database records that satisfies a database query request. OPNQRYF provides for both two-digit and four-digit years.

Date, Time, and Time Stamp Comparisons Using the OPNQRYF Command: A date, time, or time-stamp value can be compared either with another value of the same data type or with a string representation of that data type. All comparisons are chronological, which means the farther a time is from January 1, 0001, the *greater* the value of that time.

Comparisons involving time values and string representations of time values always include seconds. If the string representation omits seconds, zero seconds are implied.

Comparisons involving time-stamp values are chronological without regard to representations that might be considered equivalent. Thus, the following predicate is true:

```
TIMESTAMP('1990-02-23-00.00.00') > TIMESTAMP('1990-02-22-24.00.00')
```

When a character, DBCS-open, or DBCS-either field or constant is represented as a date, time, or time stamp, the following rules apply:

Date: The length of the field or literal must be at least eight if the date format is *ISO, *USA, *EUR, *JIS, *YMD, *MDY, or *DMY. If the date format is *JUL (yyddd), the length of the variable must be at least six (includes the separator between yy and ddd). The field or literal may be padded with blanks.

Time: For all of the time formats (*USA, *ISO, *EUR, *JIS, and *HMS), the length of the field or literal must be at least four. The field or literal may be padded with blanks.

Time Stamp: For the time-stamp format (yyyy-mm-dd-hh.mm.ss.uuuuuu), the length of the field or literal must be at least 16. The field or literal may be padded with blanks.

Date, Time, and Time-Stamp Arithmetic Using OPNQRYF CL Command: Date, time, and time-stamp values can be incremented, decremented, and subtracted. These operations may involve decimal numbers called *durations*. The following definition includes durations and a specification of the rules for performing arithmetic operations on date, time, and time-stamp values.

D.12.2.1 Durations

A **duration** is a number representing an interval of time. The four types of durations are:

Labeled Duration

A **labeled duration** represents a specific unit of time as expressed by a number (which can be the result of an expression) used as an operand for one of the seven duration built-in functions: %DURYEAR, %DURMONTH, %DURDAY, %DURHOUR, %DURMINUTE, %DURSEC, or %DURMICSEC. The functions are for the duration of year, month, day, hour, minute, second, and microsecond, respectively. The number specified is converted as if it was assigned to a DECIMAL(15,0) number. A labeled duration can only be used as an operand of an arithmetic operator when the other operand is a value of data type *DATE, *TIME, or *TIMESTP. Thus, the expression HIREDATE + %DURMONTH(2) + %DURDAY(14) is valid, whereas the expression HIREDATE + (%DURMONTH(2) + %DURDAY(14)) is not. In both of these expressions, the labeled durations are %DURMONTH(2) and %DURDAY(14).

Date Duration

A **date duration** represents a number of years, months, and days expressed as a DECIMAL(8,0) number. To be properly interpreted, the number must have the format *yyyymmdd* where *yyyy* represents the number of years, *mm* the number of months, and *dd* the number of days. The result of subtracting one date value from another, as in the expression HIREDATE - BRTHDATE, is a date duration.

Time Duration

A **time duration** represents a number of hours, minutes, and seconds expressed as a DECIMAL(6,0) number. To be properly interpreted, the number must have the format *hhmmss* where *hh* represents the number of hours, *mm* the number of minutes, and *ss* the number of seconds. The result of subtracting one time value from another is a time duration.

Time-Stamp Duration

A **time-stamp duration** represents a number of years, months, days, hours, minutes, seconds, and microseconds expressed as a DECIMAL(20,6) number. To be properly interpreted, the number must have the format *yyyymmddhhmmsszzzzzz* where *yyyy, mm, dd, hh, mm, ss,* and *zzzzzz* represent, respectively, the number of years, months, days, hours, minutes, seconds, and microseconds. The result of subtracting one time-stamp value from another is a timestamp duration.

Rules for Date, Time, and Time-Stamp Arithmetic: The only arithmetic operations that can be performed on date and time values are addition and subtraction. If a date or time value is the operand of addition, the other operand must be a duration. The specific rules governing the use of the addition operator with date and time values follow:

- If one operand is a date, the other operand must be a date duration or a labeled duration of years, months, or days.
- If one operand is a time, the other operand must be a time duration or a labeled duration of hours, minutes, or seconds.
- If one operand is a time stamp, the other operand must be a duration. Any type of duration is valid.

The rules for the use of the subtraction operator on date and time values are not the same as those for addition because a date or time value cannot be subtracted from a duration, and because the operation of subtracting two date and time values is not the same as the operation of subtracting a duration from a date or time value. The specific rules governing the use of the subtraction operator with date and time values follow:

- If the first operand is a date, the second operand must be a date, a date duration, a string representation of a date, or a labeled duration of years, months, or days.
- If the second operand is a date, the first operand must be a date or a string representation of a date.
- If the first operand is a time, the second operand must be a time, a time duration, a string representation of a time, or a labeled duration of hours, minutes, or seconds.
- If the second operand is a time, the first operand must be a time or string representation of a time.
- If the first operand is a time stamp, the second operand must be a time stamp, a string representation of a time stamp, or a duration.
- If the second operand is a time stamp, the first operand must be a time stamp or a string representation of a time stamp.

Date Arithmetic: Dates can be subtracted, incremented, or decremented.

Subtracting Dates: The result of subtracting one date (DATE2) from another (DATE1) is a date duration that specifies the number of years, months, and days between the two dates. The data type of the result is DECIMAL(8,0). If DATE1 is greater than or equal to DATE2, DATE2 is subtracted from DATE1. If DATE1 is less than DATE2, however, DATE1 is subtracted from DATE2, and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = DATE1 - DATE2.

If %DAY(DATE2) <= %DAY(DATE1)
  then %DAY(RESULT) = %DAY(DATE1) - %DAY(DATE2).

If %DAY(DATE2) > %DAY(DATE1)
  then %DAY(RESULT) = N + %DAY(DATE1) - %DAY(DATE2)
   where N = the last day of %MONTH(DATE2).
  %MONTH(DATE2) is then incremented by 1.

If %MONTH(DATE2) <= %MONTH(DATE1)
  then %MONTH(RESULT) = %MONTH(DATE1) - %MONTH(DATE2).

If %MONTH(DATE2) > %MONTH(DATE1)
then %MONTH(RESULT) = 12 + %MONTH(DATE1) - %MONTH(DATE2).
%YEAR(DATE2) is then incremented by 1.

%YEAR(RESULT) = %YEAR(DATE1) - %YEAR(DATE2).

For example, the result of %DATE('3/15/2000') - %DATE('12/31/1999') is 215 (or, a duration of 0 years, 2 months, and 15 days).

Incrementing and Decrementing Dates: The result of adding a duration to a date or of subtracting a duration from a date is itself a date. (For the purposes of this operation, a month denotes the equivalent of a calendar page. Adding months to a date is the same as turning the pages of a calendar, starting with the page on which the date appears.) The result must fall between the dates January 1, 0001, and December 31, 9999, inclusive. If a duration of years is added or subtracted, only the year portion of the date is affected. The month is unchanged as is the day unless the result is February 29 of a year that is not a leap year. In this case, the day is changed to 28.

Similarly, if a duration of months is added or subtracted, only months and, if necessary, years are affected. The day portion of the date is unchanged unless the result is not valid (September 31, for example). In this case, the day is set to the last day of the month.

Adding or subtracting a duration of days affects the day portion of the date, and potentially the month and year.

Date durations, whether positive or negative, may also be added to and subtracted from dates. As with labeled durations, the result is a valid date.

When a positive date duration is added to a date or a negative date duration is subtracted from a date, the date is incremented by the specified number of years, months, and days, in that order. Thus, DATE1 + X, where X is a positive DECIMAL(8,0) number, is equivalent to the expression:

```
DATE1 + %DURYEAR(%YEAR(X)) + %DURMONTH(%MONTH(X)) + %DURDAY(%DAY(X))
```

When a positive date duration is subtracted from a date or a negative date duration is added to a date, the date is decremented by the specified number of days, months, and years, in that order. Thus, DATE1 - X, where X is a positive DECIMAL(8,0) number, is equivalent to the expression:

```
DATE1 - %DURDAY(%DAY(X)) - %DURMONTH(%MONTH(X)) - %DURYEAR(%YEAR(X))
```

When adding durations to dates, adding one month to a given date gives the same date one month later *unless* that date does not exist in the later month. In that case, the date is set to that of the last day of the later month. For example, January 28 plus one month gives February 28; and one month added to January 29, 30, or 31 results in either February 28 or, for a leap year, February 29.

Note: If one or more months are added to a given date and the same number of months is subtracted from the result, the final date is not necessarily the same as the original date.

Time Arithmetic: Times can be subtracted, incremented, or decremented.

Subtracting Times: The result of subtracting one time (TIME2) from another (TIME1) is a time duration that specifies the number of hours, minutes, and seconds between the two times. The data type of the result is DECIMAL(6,0). If

TIME1 is greater than or equal to TIME2, TIME2 is subtracted from TIME1. If TIME1 is less than TIME2, however, TIME1 is subtracted from TIME2, and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = TIME1 - TIME2.

If %SECOND(TIME2) <= %SECOND(TIME1)
then %SECOND(RESULT) = %SECOND(TIME1) - %SECOND(TIME2).

If %SECOND(TIME2) > %SECOND(TIME1)
then %SECOND(RESULT) = 60 + %SECOND(TIME1) - %SECOND(TIME2).
%MINUTE(TIME2) is then incremented by 1.

If %MINUTE(TIME2) <= %MINUTE(TIME1)
then %MINUTE(RESULT) = %MINUTE(TIME1) - %MINUTE(TIME2).

If %MINUTE(TIME2) > %MINUTE(TIME1)
then %MINUTE(RESULT) = 60 + %MINUTE(TIME1) - %MINUTE(TIME2).
%HOUR(TIME2) is then incremented by 1.

%HOUR(RESULT) = %HOUR(TIME1) - %HOUR(TIME2).

For example, the result of %TIME('11:02:26') - %TIME('00:32:56') is 102930 (a duration of 10 hours, 29 minutes, and 30 seconds).

Incrementing and Decrementing Times: The result of adding a duration to a time or of subtracting a duration from a time is itself a time. Any overflow or underflow of hours is discarded, thereby ensuring that the result is always a time. If a duration of hours is added or subtracted, only the hours portion of the time is affected. The minutes and seconds are unchanged.

Similarly, if a duration of minutes is added or subtracted, only minutes and, if necessary, hours are affected. The seconds portion of the time is unchanged.

Adding or subtracting a duration of seconds affects the seconds portion of the time, and potentially the minutes and hours.

Time durations, whether positive or negative, also can be added to and subtracted from times. The result is a time that has been incremented or decremented by the specified number of hours, minutes, and seconds, in that order. `TIME1 + X` where `X` is a DECIMAL(6,0) number is equivalent to the expression:

```
TIME1 + %DURHOUR(%HOUR(X)) + %DURMINUTE(%MINUTE(X)) + %DURSEC(%SECOND(X))
```

Time-Stamp Arithmetic: Time stamps can be subtracted, incremented, or decremented.

Subtracting Time Stamps: The result of subtracting one time stamp (TS2) from another (TS1) is a time-stamp duration that specifies the number of years, months, days, hours, minutes, seconds, and microseconds between the two time stamps. The data type of the result is DECIMAL(20,6). If TS1 is greater than or equal to TS2, TS2 is subtracted from TS1. If TS1 is less than TS2, however, TS1 is subtracted from TS2 and the sign of the result is made negative. The following procedural description clarifies the steps involved in the operation RESULT = TS1 - TS2:

If %MICSEC(TS2) <= %MICSEC(TS1)
then %MICSEC(RESULT) = %MICSEC(TS1) -
%MICSEC(TS2).

If %MICSEC(TS2) > %MICSEC(TS1)
then %MICSEC(RESULT) = 1000000 +
%MICSEC(TS1) - %MICSEC(TS2)
and %SECOND(TS2) is incremented by 1.

The seconds and minutes part of the time stamps are subtracted as specified in the rules for subtracting times:

If %HOUR(TS2) <= %HOUR(TS1)
then %HOUR(RESULT) = %HOUR(TS1) - %HOUR(TS2).

If %HOUR(TS2) > %HOUR(TS1)
then %HOUR(RESULT) = 24 + %HOUR(TS1) - %HOUR(TS2)
and %DAY(TS2) is incremented by 1.

The date part of the time stamp is subtracted as specified in the rules for subtracting dates.

Incrementing and Decrementing Time Stamps: The result of adding a duration to a time stamp or of subtracting a duration from a time stamp is itself a time stamp. Date and time arithmetic is performed as previously defined, except that an overflow or underflow of hours is carried into the date part of the result, which must be within the range of valid dates. Microseconds overflow into seconds.

Appendix E. Design and Coding Tips

Here are some tips that might be worth considering when you make changes. For a more extensive list and more details about design and coding, see *AS/400 Performance Management V3R6*, GS24-4735.

- Use odd length packed fields for numeric data.
- Move passed parameters to local work fields before you use them.
- Avoid decimal data errors.
- Minimize activation groups within a job (ILE support):
  - It is recommended to consider changing the default of CRTPGM ACTGRP(*NEW) to another value such as ACTGRP(*CALLER). This minimizes the chance of causing a large number of activation groups to be created within a job, which creates a large overhead.
- Use ILE RPG IV built-in functions.
- Use ILE RPG IV call bound support.
- In RPG programs, check the control specification format (H) for the date format. It may have been changed.
- Reduce the number of file opens and closes.
- Do not use CANCEL/FREE with SHARE(*NO).
- Communicate between jobs with data areas and data queues.
- Consider using DB2/400 Date/Time support:
  - Avoid multiplying the MMDDYYYY format by 10000.0001 to get the format YYYYMMDD.
- Consider proper error handling within programs.
- Share access paths.
- Create a work file in library QTEMP once per job.
- Consider when to use Open Query File:
  - In batch report job streams, run OPNQRYF to subset the number of records that needs to be processed. If the stream has multiple reports in the same sequence with different records needed, do one OPNQRYF and select the records needed for all reports. In the specific report program, select only the records needed for that particular report.

    Sort may be faster if no index is available and several (5000 or more) records are selected.
- Consider sorting records for high performance batch processing.
- Minimize the number of logical files.
- Minimize the key size in logical files.
- Consider record locking and waiting in the multi-user environment.
- Consider trigger support for DB2/400.
- Consider Database Referential Integrity support for DB2/400.
- Consider journaling.

- Consider commitment control:
  - Commitment control can now be used or not used in an RPG program depending on user's choice. There is a parameter that can be set on or of on the COMMIT keyword for the file.
- Minimize data sent to or received from a display:
  - Check the use of the display parameter RSTDSP(*YES).
  - Minimize the use of break messages during peak workloads.
  - Use the DDS window support.
- Minimize the amount of workstation specific data processing:
  - Minimize the use of 5250 "numeric only" field types ("Y" type fields).
  - Use DDS keyword for data checking and validations.
  - Minimize the number of subfile records written to the subfile.
  - Minimize the number of subfile records processed on input.
- Define attention keys rather than command function keys.

Appendix F. Year 2000 Certification Steps for Solution Providers

For a solution provider who wants to certify the solutions from IBM with respect to Year 2000 enablement, the provider has to present the following guidelines:

- The solution provider must submit an *Impact Analysis Comparison Report* to IBM. The report compares the Impact Analysis reports produced before and after the Year 2000 enablement of the application software. These impact analyses are carried out using a tool provided by IBM. Actual reports produced by the tool must also be included with the comparison report.
- Solution providers must be able to demonstrate that they have tested all of the representative cases of data that can be fed into the application software. Moreover, if service providers have some kind of a pre-built, pretested testing material available that they have used to test the application, they must prove that the testing material is valid to-date.
- The solution provider must submit the actual screen prints and reports produced by the application software before and after the Year 2000 enablement of the software.
- The solution provider must also submit documentation of the types of decisions used during the conversion process. This includes any cosmetic adjustments made, decisions about date formats used, as well as the variable representation techniques used and any improvements done to the software. The report should also cover if any date fields have been left intact such as accessing the date fields using bridge programs.

Appendix G. Testing Handbook

The general idea, terminology, and forms examples used in this appendix were inspired and copied from the unpublished *Full Lifecycle Testing Handbook*. The authors of this document want to express their thanks to Rajesh Sivaraman for his cooperation and willingness to let us use his material.

Disclaimer
While the testing procedure that is presented in this section is quite rigorous in its approach, IBM does not warrant nor guarantee that if it is followed, that all problems with the code will be found. It is the responsibility of the customer to satisfy themselves that the application is in good working order. IBM cannot be held responsible for application modifications that are made beyond IBM's control.

G.1 Develop the Master Test Plan

For each Year 2000 enablement project, there must be sufficient time allowed for testing. Even with the best conversion tools, some obscure date conversion may elude even the best programmers or automatic conversion tools. Therefore, a master test plan must be developed to make sure that the risk of any undetected date problems causing significant damage when the code is returned to production. This section documents the steps that are used as a sensible approach to testing.

G.1.1 Organize the Test Team

You must assign the following responsibilities to specific individuals:

- Test planning
- Test preparation
- Running the tests
- Test reporting

G.1.1.1 Entry Criteria

- Project plans are available.
- Initial statements of requirements are available.

G.1.1.2 Process Details

1. Assign the test manager or coordinator during the early part of the requirements phase. Assign responsibilities for each major test activity such as test planning, test preparation, and so on.
2. Select team members and assign responsibilities as the project progresses at the different test levels.
3. Identify testing activities to be performed.
4. Identify individuals who can perform (or be trained to perform) those activities.
5. Estimate when and how much time is required for test team members to performs their activities.

6. Obtain management commitment to make staff available when required for the test team.

G.1.1.3 Activities

Test planning and coordinating activities are:

- Co-ordinate test planning (master test plan, detailed test plan).
- Schedule the tests.
- Co-ordinate test environment requirements and setup.
- Interface with interested groups (users, product acceptance, ...).
- Organize the verification of test results.
- Co-ordinate preparation and reporting of test results.
- Co-ordinate review and inspection of test plans and test deliverables.

Test preparation activities are:

- Co-ordinate setup of test environments (hardware, terminals, communication lines, ...).
- Analyze test data needs.
- Develop test databases.
- Develop test specifications and procedures.
- Develop test cases.

Running the Test activities are:

- Manage problems and control change.
- Co-ordinate test execution activities.
- Liaison with Technical Library.
- Manage the test libraries.
- Running the tests.
- Resolve technical problems (environmental).
- Route and deliver test output.
- Verify test results.

Test Reporting activities are:

- Prepare test documentation.
- Prepare test reports.

G.1.1.4 Responsibilities

Supervision

This is the role of the Test Leader. Normally you assign this role to someone on the service provider's staff, but it can be assigned to the customer, or even someone from a third party.

Test Material

The task of producing the test material should be assigned to someone from the customer staff; only they can determine what is significant data and make sure that all occurrences of date-related processing are accounted for during the test.

Test Tools

If the tools you have selected for the conversion do not provide a testing function, someone should be assigned the task of selecting a suitable test tool. Even though this might seem to be of minor importance, especially if you are converting a small system, we suggest you evaluate possible tools to help you perform the test as safely as possible. This task is best assigned to someone from the service provider's staff.

Verification

This task should be carried out by someone from the customer's staff. To perform this task, the verifier uses documents and reports produced during the testing phase, corresponding baseline material and verifying that everything conforms to the success criteria stated in the statement Of work.

G.1.2 Determine Test Focus Areas

Description

Select the test focus areas that are important for the Year 2000 project. For reference, test focus areas are those aspects of a system, such as auditability or reliability, that may require special testing.

Purpose

Ensure that critical or sensitive attributes of the system receive extra focus so that the overall quality of the system is as high as possible.

G.1.3 Evaluate Requirements for Testability

Description

Verify that each major requirement (business function) can be objectively tested.

Purpose

Ensure that a **measurable** test scenario can be created for each major business function in order to verify that the requirements have been met.

G.1.4 Develop Test Objectives

Description

Define major test objectives and evaluation criteria for each of the business and structural functions to be tested.

Purpose

- Establish test objectives for determining what testing is to establish.
- Define evaluation criteria to know when testing is complete.

Details

1. Use test focus areas to help define the test objectives.
2. Use brainstorming techniques to define objectives.
3. For each business or structural function, define the test objectives to be achieved. Relate them either to input (event triggered) or to output (function driven).
4. Prioritize the objectives.
5. Describe the evaluation criteria to determine achievement of objectives.

6. Verify that the objectives meet the SMART (Specific, Measurable, Attainable, Realistic,and Timely) criteria.

G.1.5 Develop Test Strategy

Description

Determine the testing strategy to address the testing concerns and focus the test effort on areas with high and medium risk.

Purpose

Outline the strategy to include:

- Determine the approach to achieve the test objectives.
- Identify areas to focus the testing effort.
- Select levels/types of testing.
- Identify the individuals to do the tests.
- Identify any exposures.

G.1.6 Develop Test Schedules

Description

Establish work schedules for testing activities.

Purpose

Ensure that:

- Test resources (personnel, hardware, and software) are available as needed.
- Any assumptions or constraints on the schedule have been identified.
- Testing resources are used effectively.

Details

1. List all test activities to be performed.
2. Estimate the time required to perform the test activities and record these using the selected project management tool.
3. Determine any critical paths that may impact testing.
4. Review test team availability to ensure that all resources are available as required.
5. Identify any training requirements for test team members and include these in the project plan and schedule.
6. Schedule test activities to ensure appropriate focus on high priority items.
7. Prepare an outline draft of the procedures that are used to monitor and control test activities and to handle contingencies.
8. Set up frequent check-points (suggest weekly) during the test cycle to review status with users.
9. Ensure that the schedule is signed off by the project manager and the user manager responsible for accepting the revised system.

G.1.7 Define Problem/Change Management

Description
Define the procedures to be used during the project to:

- Record problems (variances).
- Track problems from identification to resolution.
- Track the progress of fixes.

Purpose
Ensure that:

- High priority problems receive immediate attention.
- Problems do not fall through the cracks.
- Problems are not closed without approval.
- Exact problem status is known (found, to be fixed, fixed).
- Data is available for causal analysis.

Details
If the customer has a problem/change management process (such as Infoman), use it. If not:

- Develop methods for documenting problems.
- Determine the method for reporting problems and variances and assigning priorities.
- Establish procedures for handling problems based on priority.
- Establish responsibility for analyzing variances, evaluation of causes, and possible solutions.
- Establish standards for each level of testing to retest fixes.
- Establish criteria for the problems to be closed.
- Develop procedures to track the progress of problems from problem identification through closing of the problem ″record″.

G.1.8 Define Facility Requirements

Description
Define and document the facility (environment) requirements for testing.

Purpose
Ensure that the hardware, software, and organizational resources necessary to run the various types and levels of tests are committed to the project in the time frame required to meet the test schedule.

Details
Identify the test environment and list the various facilities that are required for the testing activities:

- Test facility requirements - acceptance and operability testing environments, availability schedules, and so on.
- Hardware requirements - CPU, DASD, and so on.
- Special software tool requirements (test tools).

G.1.9 Document Master Test Plan

Description

The master test plan is a high level ″plan of attack″ explaining the overall strategy. It documents:

- System objectives
- Test team members
- Test objectives
- Resource requirements
- Test strategy
- Tools to be used
- Testing milestones
- Methods to be used
- High-level test schedule
- Procedures to be used

Purpose

Answer questions such as:

- What will be tested?
- How will testing be performed?
- What resources are needed, and when?
- What are the acceptance criteria?

Details

1. Assemble the master test plan using relevant sections form this chapter.
2. Match any risk against test strategy.
3. Ensure that unmatched risks are recorded and assigned for resolution.
4. Verify that skills required to perform the levels and types of testing are available when needed.
5. Verify that necessary procedures to handle problem management are in place.
6. Review the document and obtain necessary sign off.

G.2 Develop Detailed Test Plan

Description

This activity is done for **every level** of testing. The detailed test plan is a detailed plan that describes a specific level of testing. It documents:

- Administrative plan
- Test objectives
- Types of test
- Techniques and tools
- Configuration management procedures & controls
- Entry and exit criteria

Purpose
The purpose of the detailed test plan is very similar to that of a master test plan except that it refers to a specific level of testing and contains more detailed information. It answers such questions as:

- What will be tested?
- How will the testing be performed?
- What resources are needed, and when?

G.3 Prepare for the Tests

This section gives suggestions and guidance on the activities which should be done prior in preparation to testing.

G.3.1 Select Techniques and Tools

Description
Select the testing techniques and tools as appropriate for each development phase and testing level.

Purpose
Evaluate the different techniques available for use and choose the ones that are most effective for the particular testing level and types of tests performed at that level.

Details
Use the following guidelines for selection of appropriate techniques:

- Unit level testing - select the applicable white box testing techniques.
- Integration level testing - a combination of white box for testing module interfaces and black box for testing functions.
- System level testing - white box techniques when testing system interfaces from a technical point of view; black box techniques when testing functions to other application interfaces.
- User acceptance testing - black box techniques for testing compliance to functional requirements.

G.3.2 Build and Modify Test Data

Description
Analyze data from the following sources in order to determine the data that is required to test the application:

- Existing production data
- Data from previous testing

Prioritize test conditions and cases created in the previous steps. Build test data by modifying existing data and supplementing them with additional data as required.

Purpose

- Determine the potential sources for test data.
- Identify methods to extract data.

- Build test data and verify that all test cases are covered.
- Document the test data for future use.

Details

1. Analyze data from production data to determine the type, frequency, and characteristics of data used by the application being tested.
2. Determine the appropriate sources of data to use.
3. Determine the appropriate files and sizes to use.
4. Determine the appropriate methods to extract test data.
5. Prioritize test conditions and cases to be used. Ranking should be a function of risk and test focus area. Ranking need not be a formal ″number crunching″ exercise.
6. Use the determined methods to extract test data.
7. Customize test data to assure adequate coverage.
8. Document the expected results.
9. Use detailed test matrices to validate coverage.
10. Document test data:
    - What is being tested?
    - How is it being tested (procedures, and so on)?
    - Where to find test data.
    - Procedures to re-run.
    - Expected results.

Considerations

The more complete test you perform, the greater the likelihood that you can find and eliminate whatever bugs might be remaining in the application. The ideal situation should, therefore, be that you perform a complete test with copies of the total, complete database using actual transactions with only dates changed between the before and after test.

Depending on resources available, however, you may have to settle for a testing environment where you extract a significant subset from both the database and the transactions. The selection or extraction of such data should be the responsibility of the customer who is more familiar with the database, the transactions, and their content.

Test Material Selection Criteria

In selecting the test material, your customer may require some assistance. There are a lot of issues that have to be resolved before the test material actually is produced. Some are dependant on resources available, whether it is feasible to run a full-scale parallel test, or if you have to extract data from the database files and the transactions so only a significant subset is used for the tests.

You have to consider several scenarios and parameters when designing the test data:

Interactive processing

How are you going to reproduce data into the application? Do you have enough operators and terminals to do it manually? Can you control the quality of the entered data in respect to both the correctness, volume, and speed? Since every test after a rewrite or modification of an application should be benchmarked against the same test run before the change, you can consider using the following methods to capture and replay keystrokes:

Most PC based terminal emulators have the option to let you record and playback macros. Use this feature to create a test that can be repeated several times, starting with the original, unchanged, version of the application.

After recording the keystrokes, you must edit the macros produced in order to insert the updated user interface into the macro. You also must alter the actual information entered through the macro so that it reflects transactions within the date ranges around the shift of the century that you are testing.

Batch Batch processing contains several types of programs or functions that you must test. These are both programs that produce reports, as well as programs that just performs calculations and updates to the database. When looking for these, you must consider:

- Periodically run functions, such as:

  End of day processing
  End of week Processing
  End of month Processing
  End of quarter Processing
  End of year processing
  Others...

- Adhoc run functions:

  These might be harder to identify since they are normally not identified in the schedules, but might just be documented on the user′s desktop or personalized menu. Look for items such as:

  Query/400 queries
  QMF queries
  Batch programs that are referenced from personalized menus only, or not referenced elsewhere at all.

G.3.3 Develop Test Run Procedures

Description

Develop step-by-step descriptions of tests to be performed including data reduction and analysis for setting up regression test data and define evaluation criteria for tests.

Purpose

- Define the sequence and contents of tests.
- Establish pass/fail criteria for individual tests.
- Establish entry/exit criteria for each test.

Details

1. Determine which tests are run and in what order. Include test schedules and prerequisites.
2. Establish pass/fail criteria for tests.
3. Assign responsibility for conducting the test.
4. Identify requirement for data analysis and data reduction for regression testing.
5. Establish entry/exit criteria for each individual test.
6. Identify end-of-test close activities that need to be performed.
7. Identify any restore/backup activities that need to be performed at the start and end of tests.
8. Include detailed instructions to restore/backup files.
9. Identify any special test environment setup or prerequisites for each test.
10. Include instructions to handle unexpected error situations.
11. Document all execution procedures.

G.4 Run the Tests

This section describes the various types of tests which should be run prior to handing the application over the production.

G.4.1 Run Unit Tests

Description

Unit level testing is the initial testing of new and changed code in a program module. It validates the internal logic.

Purpose

It assures that:

- Adequate statement, condition, decision, and path coverage are provided (dynamic validation of specifications).
- Implementation at module level is defect free.
- Error recovery procedures work correctly at module level.

Details

1. Execute all test cases in the unit test plan.
2. Verify successful execution of all new, changed, or affected paths.
3. Maintain a log of unit test cases executed.
4. Fix problems and re-test.
5. Put an action plan in place for unresolved problems. Some problems may result in change requests that have to be processed through the normal change management steps.
6. Update test documentation as necessary.

G.4.2 Run Integration Tests

Description

Integration tests verify the proper execution of components within an application.

Purpose

The purpose of the integration testing is to:

- Exercise new, changed, and affected application functions and verify that they are operationally acceptable and in accordance with internal design.
- Exercise new, changed, or affected interfaces between modules and subsystems and verify that they are functioning correctly.

Details

1. Verify successful execution of all integration test cases as per integration test plan.
2. If problems are found, fix and re-test in a unit test environment.
3. Run regression tests in the integrated environment.
4. Put an action plan in place to fix outstanding problems. Some problems may result in change requests that have to be processed through the normal change management steps.
5. Update the documentation as necessary.

G.4.3 Run User Acceptance Tests

Description

User acceptance tests verify that the system meets user requirements as specified. The details of this activity are replaced by the test execution plan activities when the user acceptance test is conducted in an acceptance test environment.

Purpose

The purpose of user acceptance test level test is to:

- Run the application in a production-like environment.
- Verify that the ″right″ system is built and that it meets functional, reliability, documentation, and security considerations.

Details

1. Verify successful execution of all user acceptance test level test cases as per the user acceptance test plan.
2. If problems are found, fix and re-test them in unit, integration, and system test environments.
3. Promote fixes as per configuration management procedures.
4. Re-test fixes in the user acceptance test environment.
5. All problems encountered should be resolved or deferred based upon user decision.
6. Complete regression tests as necessary to cover typical and key business scenarios.
7. Update test documentation as necessary.

8. Obtain sign off from users that the system is ready to be implemented into production or promoted to operability testing.

G.4.4 Run Operability Tests

Description

Operability tests verify that the application can operate in the production environment. Operability tests are performed after or concurrent with the user acceptance test.

Purpose

The purpose of the operability level test is to:

- Verify that the application can provide an acceptable level of performance as defined.
- Ensure that the integrity of the application can be maintained and that the application can be recovered and re-started after failure.
- Ensure that computing resources are properly utilized.
- Ensure that the application can coexist with other applications.
- Ensure problem-free operation.
- Verify compliance to operability standards.

G.5 Report Test Results

This section gives detail of what should be documented and reported when after test results have been analyzed.

G.5.1 Analyze Test Results

Description

Actual results are compared to expected results. Variances are identified and handled.

Purpose

Identify those tests that have succeeded and, for those tests that have failed:

- Identify the problem.
- Resolve the problem.
- Document the resolution.
- Re-test if required.

G.5.2 Catalogue/Document Tests

Description

Analyze the test scripts and test data and document those that can be saved and re-used.

Purpose

- Identify regression testing requirements.
- Identify, document, and save test items for re-use.

Details

1. Well-documented tests serve as system documentation for future testing. It is, therefore, essential that all test conditions, test scripts, and data created during tests are described and stored such that they can be easily retrieved and re-used. Establish a suitable controlled and secure method for cataloging the test cases.
2. Cross-reference test matrices and test conditions to test cases and test data.
3. When test conditions are no longer valid, deletions can be cascaded to test cases and then to test data. This eliminates examining the data every time to determine whether or not they still are required.
4. Identify test conditions and cases that should be included as part of regression testing. This reduces the effort required to examine every test condition and case later on to set up regression test data.
5. Always include any conditions and data that tested previous production problems.
6. Use simple meaningful conventions and procedures to document tests and test data.

G.6 Work Sheets

The following sections contain examples of work sheets that you can either copy or modify to fit your requirements.

G.7 Inspection Cover WS201

Project Name:_____ Inspection Date: _____

Inspection Phase:_____ Material: _____

Moderator's Name:_____ Phone : _____

Author's Name :_____ Dept. : _____

Inspector's Name:_____ Phone : _____

Preparation Time:_____._____ Hours Prep. Completion Date: _____

General Comments on Material

G.8 Inspection Minor Defect List WS202

Project Name:_____ Inspection Date: _____

Inspector's Name:_____ Phone : _____

Page #	Line #	Defect Description, Text of Suggestion or Issue

G.9 Inspection Major Defect List WS203

Project Name:_____ Inspection Date: _____

Inspector's Name:_____ Phone : _____

Page #	Line #	Defect, Issue or Suggestion	1	2	3	Fix Date

1 - Category	M = Missing	W = Wrong	E = Extra
	I = Issue	S = Suggestions	
2 - Defect Type	D = Design	L = Logic	S = Standards
	I = Interface	R = Return Code	M = Messages
	C = Comment	X = Other	A = Data
3 - Defect Source	P = Project Def'n	E = External Design	C = Construction
	R = Requirements	I = Internal Design	I = Implementation
	O = Operations	X = Other	

G.10 Inspection Summary Sheet WS204

Project Name:_____ Inspection Date: _____

Proj. Mgr's Name:_____ Phone: _____

Inspection Phase:_____ Material: _____

Moderator's Name:_____ Phone: _____

Author's Name:_____ Dept.: _____

Inspectors' Names:_____ Prep. Time: _____._____

:_____ Prep. Time: _____._____

:_____ Prep. Time: _____._____

:_____ Prep. Time: _____._____

:_____ Prep. Time: _____._____

:_____ Prep. Time: _____._____

Meeting Length :_____._____ Hours Total Prep. Time: _____._____

If Re-inspection was done, date of first inspection was: _____/_____/_____

Total Material Inspected:

Pages New: _____ Modified: _____

Lines of Code New: _____ Modified: _____ Language: _____

I certify that all defects discovered during htis inspection will be corrected or resolved, and that no known defects will remain in the inspected material when this material is ready for the next stage of development.

Author: _____ Date: _____

Inspection Completed: _____ Approved By: _____

Test Team Utilization WS302

Project Name:_____

Person(s) Responsible:_____

Name	Department	Skills	Time Requirement	Responsibility

G.11 Test Focus/Types WS305

Project Name:_____

Person(s) Responsible: _____

Test Types		Functional Testing											Structural Testing				
	Priority	Audit & Control	Conversion	Documentation	Error Handling	Function	Interface	Installation	Parallel	Regression	Trans. Flow	Useability	Backup & Recovery	Oper./Job Stream	Performance	Security	Stress/Volume
Accountability		O				O							O				
Continuity of Processing													O	O			O
Correctness		O	O	O	O	O	O	O	O	O	O	O					
Maintainability				O													
Operability				O	O									O			
Performance												O		O	O		O
Portability								O						O			
Reliability					O	O						O	O				
Security		O				O										O	
Useability				O	O	O						O					

G.12 Business Functions WS306

Project Name:_____

Person(s) Responsible: _____

Ref #	Business Function	Objectives	Evaluation Criteria

G.13 Structural Functions WS307

Project Name:_____

Person(s) Responsible: _____

Ref #	Structural Function	Objectives	Evaluation Criteria

Test Strategy WS308

Project Name:_____

Person(s) Responsible:_____

Ref #	Function or Risk	Priority	Test Strategy	Action/Person(s) Responsible

G.14 Levels/Types of Tests WS309

Project Name:_____

Person(s) Responsible: _____

TYPES / Levels	Unit	Integration	Sytem	U.A.T.	Operability
Audit & Control		o	o		
Conversion	o		o		o
Documemtation & Procedure			o	o	o
Error-handling	o	o	o	o	o
Function	o	o	o	o	
Installation			o		o
Interface/ Inter-system			o	o	o
Parallel			o	o	o
Regression	o	o	o	o	o
Transaction Flow			o	o	
Useability			o	o	
Backup & Recovery	o		o		o
Contingency					o
Jobstream			o	o	o
Operational			o		o
Performance			o		o
Security	o		o		o
Stress/Volume			o		

G.15 Build Strategy WS310

Project Name:_____

Person(s) Responsible: _____

Ref #	Build Ref # Function Description	1	2	3	4	5	6

G.16 Functions Matrix WS311

Project Name:_____

Person(s) Responsible: _____

Build Ref#: _____

WS311 Ref#	Sub-Function Description	WS316 Ref#

G.17 Test Set Matrix WS316

Project Name:_____

Person(s) Responsible: _____

Sub-function Ref # (from WS311):_____

Ref #	Test Set # / Condition Description	1	2	3	4	5	6	7	8	9	10

G.18 Test Conditions Matrix WS317

Project Name:_____

Person(s) Responsible: _____

Condition 1	Condition 2	Condition 3	Condition 4	Expected Result

G.19 Test Cases Matrix WS318

Project Name:_____

Person(s) Responsible: _____

Test Set Ref: _____

Test Case# Condition Ref#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

G.20 Variance Recording and Analysis WS319

Project Name:_____

Person(s) Responsible: _____

Var.#:		Severity:
Test #:	Test Name:	
Status:	Updated by:	Date:
Desc. of Problem:	Sub. by:	Date:
Analysis:	Analyzed by:	Date:
Affected Modules:		
Estimate:	Est. by:	Date:
Recommentation	Rec. by:	Date:

G.21 Test Variance Impact Statistics WS320

Project Name:_____

Report Date: _____

Person(s) Responsible: _____

Phase: _____

	Reported		Closed		Open	
Variances Impact	No.	%	No.	%	No.	%
Severity 1						
Severity 2						
Severity 3						
Severity 4						
Total						

G.22 Test Case Specification WS321

Project Name:_____
Person(s) Responsible: _____

Written By:	Executed By:
Date:	Date:
Conditions To Be Tested	
Test Preparation	
Execution Procedure	
Expected Results	
Action Taken	

Pass		Fail	

G.23 Testing Incident Report WS325

Project:_____ Test Level: _____

Date & Time Observed:_____ Observed By: _____

Test case #'s involved (where applicable): _____

Incident Description:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Resolution Status:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Quality Category: _____ Assigned to: _____

Keyword: _____ Severity: _____ Incident #: _____

Appendix H. Special Notices

This redbook is a collection of information needed by service providers to assemble a service offering to assist customers and solution providers in the Year 2000 enablement process.

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 information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. 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 document 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, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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

IBM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Other trademarks are trademarks of their respective companies.

Appendix I. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

I.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 163.

- *AS/400 Applications: Moving to the 21st Century*, SG24-4790
- *Software Life Cycle Management with Application Development Manager/400 Systemview Manager/400*, SG24-4187
- *Moving to ILE for RPG IV*, GG24-4358
- *DB2/400 Advanced Database Functions*, GG24-4249
- *An Implementation Guide for AS/400 Security and Auditing*, GG24-4200

I.2 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
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

I.3 Other Publications

These publications are also relevant as further information sources:

- *AS/400 Application Development ToolSet/400: Application Development Manager/400 User's Guide*, SC09-2133.
- *AS/400 Application Development ToolSet/400: Application Dictionary Services/400 User's Guide*, SC09-2087.
- *AS/400 Basic Security Guide*, SC41-3301
- *AS/400 Security - Reference*, SC41-3302
- *ILE Concepts*, SC41-3606
- *ILE Application Development Example*, SC41-3602.
- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251.

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 URL 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 lists of redbooks:

  ```
  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:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page 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.

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** (Do not send credit card information over the Internet) — 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) 415 855 43 29 (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 Home Page	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).

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.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

ADM/400	Application Development Manager/400
ADS/400	Application Dictionary Services/400
ADTS/400	Application Development Tool Set/400
APA	all points addressable
CL	Command Language
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
PROFS	Professional Office System

Index

Special Characters

A

B

C

D

E

G

I

L

O

P

R

S

T

U

Printed in U.S.A.