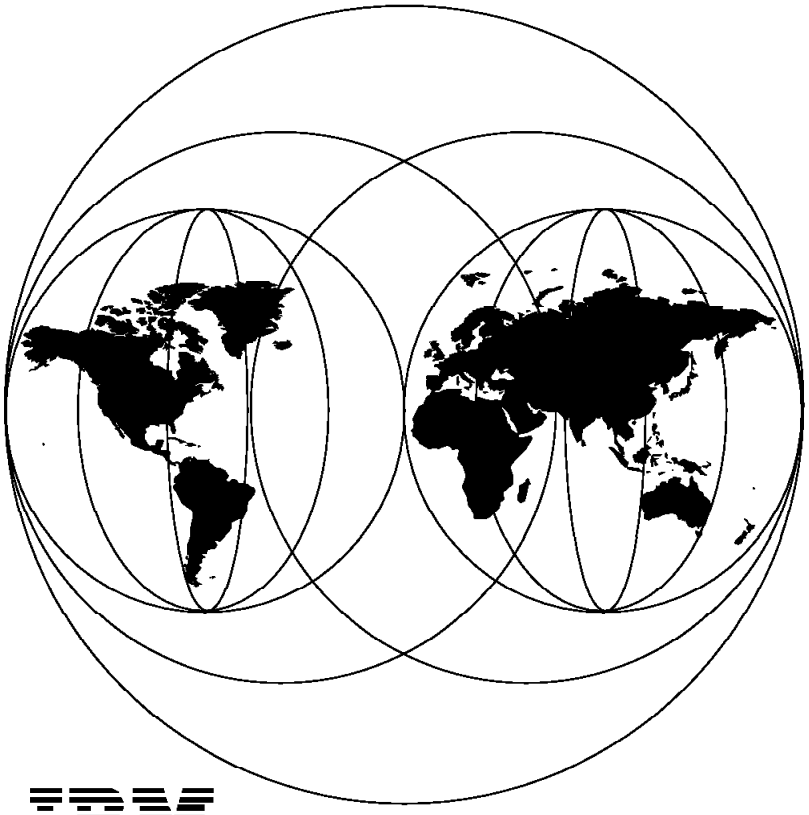
SG24-2042-00

VM/ESA Year 2000 Migration - A Case Study

October 1997



IBM
International Technical Support Organization
Boeblingen Center

International Technical Support Organization

SG24-2042-00

VM/ESA Year 2000 Migration - A Case Study

October 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special Notices” on page 151.

First Edition (October 1997)

This edition applies to Virtual Machine/Enterprise Systems Architecture (VM/ESA) Version 2 Release 2.0, program number 5654-030, and subsequent releases.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. 3222 Building 71032-02
Postfach 1380
71032 Böblingen, Germany

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

Contents

Figures

Tables

Preface

Year2000 migration is considered to be the largest and most daunting project in the history of information technology. This redbook will help you design and implement a plan to migrate your VM/ESA system to a Year2000-ready state. Included is guidance both on migrating the VM/ESA operating system itself to VM/ESA Version 2 Release 2, the Year2000-ready release, from previous levels of VM/ESA, and on identifying and correcting Year2000 exposures in your application programs to permit them to be migrated successfully as well.

The heart of the material in this redbook is a record of real-life experience in VM/ESA Year2000 migration. The project on which the redbook is based was designed as a case study, using partial snapshots of two different IBM customers′ VM/ESA systems to represent various aspects of typical production VM/ESA environments. The three authors planned, implemented, tested, and documented the migration to VM/ESA Version 2 Release 2 and the conversion of a diverse collection of application programs - all in the course of six weeks.

Also included in this redbook is a brief introduction to the Year2000 challenge in general and specifically to the Year2000 support provided by VM/ESA Version 2 Release 2. Throughout the book you will also find a variety of pointers to other sources of information on Year2000 issues. Thus the book can serve as your primary source of reference material on VM and the Year 2000.

This redbook will be of value to VM/ESA system administrators, system programmers, and application programmers who are facing the challenge of ensuring that existing systems and applications function correctly into the next millennium and that new systems and applications are designed and implemented with full Year2000 enablement. It should also be useful to other VM/ESA users who need or want a fuller understanding of VM/ESA-based solutions to Year2000 problems.

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, Böblingen Center.

The authors of this redbook are:

Stuart Byrne works for IBM New Zealand. He has spent 12 of the last 15 years working for IBM in a variety of VM related roles.

Perry Ruiter works as Senior Technical Analyst in the VM group at the Government of the Province of British Columbia, Canada. He has worked exclusively with VM for the last 12 years helping users exploit its capabilities. He can be reached at: Perry.Ruiter@gems1.gov.bc.ca

Roger Thibault works in VM Support at the IBM Support Centre, in Montreal, Canada. Of his 25 years with IBM, he has spent the last 22 supporting VM and most of its associated Program Products.

The residency that produced this redbook was coordinated by:

Stephen Record
International Technical Support Organization, Böblingen Center.

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

Pamela Christina
IBM Endicott

Tracy Dean
IBM Santa Teresa

John Franciscovich
IBM Endicott

Mary Stefos
IBM Endicott

Melinda Varian
Princeton University

Jens Völker
IBM Böblingen

Gudrun Wiedemann
International Technical Support Organization, Böblingen Center.

We would also like to express our appreciation to the following software vendors for permitting us to make use of one or more of their products on the case study systems during the residency:

MiraSoft, Inc.
Boston, Massachusetts

Sterling Software
Reston, Virginia

Syncsort Incorporated
Woodcliff Lake, New Jersey

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 175 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

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

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. Understanding the Year2000 Challenge

1.1 The Year2000 Challenge

The scope of the Year2000 challenge spans the entire Information Technology industry.

It affects all computer platforms, it is not unique to IBM processors, and it does affect customers with the VM operating system.

Historically many computer programs made use of a two-digit date format (95 rather than 1995), and this causes programs (both system and application) that perform arithmetic operations, comparisons or sorting of date fields to give incorrect results when working with dates outside the range of 1900-1999.

Although generally referred to as the Year2000 problem, it is really a two-digit year problem.

1.2 Planning Aspects

You need to plan for and resolve the date problem well before January 2000; otherwise, your computing environment will not function as expected, because the systems or application programs will not process dates later than December 31st 1999 properly.

Planning for and implementing a strategy to meet the Year2000 challenge is not a trivial project.

It is clear that to provide solutions to Year2000 problems, changes must be made both to the operating system environment and also to application programs. The problems cannot be ignored.

Plans must be put into place to ensure that both the operating system itself and the application programs address the Year2000 challenge.

To address date sensitivities within the VM operating environment, you should plan to migrate to VM/ESA Version 2 Release 2, which will incorporate all requirements necessary to handle dates beyond 1999.

To address Year2000 sensitivities within application systems and application programs requires decisions on the adoption of either a long-term or a short-term solution, and detailed planning for resolution. When selecting a proposed Year2000 solution, you should evaluate the following factors:

- **What is the external impact due to incompatible date format changes?**

  What other programs or what output will be affected and to what extent will those programs require change if this solution is implemented for this particular application?

- **How current are the program modules that reference the date format externalized by the exposures?**

Are there any plans to either eliminate or replace this particular program or routine, the programs that provide input to the program, or those receiving output from the program?

- **What functions will be impaired due to Year2000 exposures?**

  Will any mission critical function within your business be compromised due to not reworking or replacing a particular program?

You should also have a plan in place to quantify the objectives before making the decision for either a short-term or long-term solution.

You may consider the following quantifiers:

- **Identify and communicate the organization goal**

  The goal is to have the function and operation of an organization Year2000 ready before any disruption caused by two-digit year data occurs.

  The term ***Year2000 ready*** means the capability of a product, when used in accordance with its associated documentation, to correctly process, provide and/or receive date data within and between the 20th and 21st centuries, provided that all products (for example, hardware, software, and firmware) used with the product properly exchange accurate date data with it.

- **Identify the deliverables and associated schedules for the following:**
  - *Hardware*
    - Is the current processor Year2000 ready and capable of running the Year2000-ready versions of all necessary software?
    - Does the current processor allow for additional processing overhead and the test workload?
    - If I order a new processor, when will it arrive? When will it be installed?
    - Are there enough disk volumes for a new release of VM/ESA or for an expansion of files and databases to cater for more bytes in each date field?
    - Are there enough disk volumes for test data?
    - Does my current configuration cater for additional DASD? Do I need additional controllers as well? If I order now, when will they be installed?
    - Do I have enough terminals to cater for additional workloads for my people? Do they require workstations?
  - *Software*
    - I know I have to migrate to VM/ESA Version 2 Release 2. When do I schedule this migration? What impact will it have on my production system?
  - *Systems Software Vendors*
    - I know that I have to upgrade or migrate to compatible versions of non-IBM vendor software. When do I install these products? Should I install now if possible? Will they run on my current version of VM, so that I may take advantage of new functions now? What PTFs do I need to apply?

- Are these products Year2000 enabled? Contact vendors for a commitment for Year2000-ready products.
- Some vendor products check for the date for entitlement logic. How will your vendor handle that for doing the test?

– *Data Interchange*

- How is your environment organized? Are you communicating with other systems and locations within your company? Are you interchanging data with suppliers or customers? In these cases you need complete and thorough coordination.
- Are particular date fields or date-sensitive data items used in multiple applications? If so, conversion efforts need to be planned so that all the applications concerned are converted at the same time.

– *Documentation*

- Just how good is my documentation? Do I need to update my system diagrams to reflect new hardware and software? When will this be done?
- When are the new DASD layouts going to be available? Has my system programmer caught up with the documentation from the last upgrade?

– *Training*

- There are new operator messages. What are they? Where are they documented?
- When is education scheduled for the new releases of my optional products? Do I need to send all my applications people on courses? When?

– *Maintenance*

- Are all my devices at the correct hardware EC level? Does my hardware engineer have to order new microcode? When will the upgrade be performed?

– *Operations and Administration*

- Are there extra programs to run in the batch window because of changes to cater for Year2000-enabled programs? Will my batch window now be too small?
- Will my on-line system take longer to initialize?
- Are there changes to the network?
- Are there changes to operational procedures?

– *Acceptance criteria for all deliverables*

- Has all hardware been tested and in production? Has the microcode been placed in operation or in test mode? Is it reversible?
- Has the new system software been tested thoroughly? Is there any impact on my production system? Can I revert back to my old system? How long will it take?

• **Analyze job assignments**

– Identify the responsible person or organization. For example:

- Customers

Identify the person responsible for changes of data format on reports or documents shipped with goods. Determine people in customer sites who will be dealing with data interchange. Coordinate changes made internally and externally. Make the customers aware of changes being made.

- Management
  - Chief Executive Officer

    Determine how the Chief Executive will be informed of progress and who will prepare documentation for the Board of Directors for approval.
  - Chief Financial Officer

    Determine who is responsible for additional expenditure for hardware and software, if required.
  - Software Development Managers

    Determine who is going to manage the Year2000 project and under which criteria or priority. Determine which projects may be deferred by giving priority to the Year2000 project. Determine Team Leadership and project ownership.
  - Operations and Administration Managers

    Determine who will manage and own the Year2000 project. Determine the best testing techniques and who will manage the resources.
  - Budget and Finance Managers

    Determine whom to consult about budget over-runs, requirements for additional staff, who pays for what and when.
- Vendors and Service Providers
- Systems and Operations personnel
- End users
- Auditors and quality assurance people

– Measure the estimated completion time

– Identify precedences and dependencies, resources and skills, schedule

– Measure efforts, costs

– Analyze potential benefits
  - Return on investment
  - Achievement of business goals
  - Potential quality and acceptance of the approach
  - Your business keeps running

– Analyze risk factors
  - Complexity of the task
  - Resource and time constraints
  - Length of project
  - Critical development skills

- **Measure the criticality of each task and set priorities**

  Evaluate and determine how critical the functions of each entity are to the business success of the organization, and set a priority for providing Year2000-ready solutions. The factors contributing to how critical a task may be might include pressure of demand from end users, legal issues, financial issues, or political issues.

  – *Impact on the Business*

    To determine the impact on your business, consider:

    - Is it critical to the operation of the business (legal compliance)?
    - Is it critical to the uninterrupted operation of the business (such as payroll)?
    - Is it required to support the business (such as management or financial reports)?
    - Is it desirable, but not absolutely required to support the business?

  – *Impact on Operations*

    To assess the severity of the impact on your operations, you could use categories such as:

    Fatal Operations will ABEND or terminate.

    Critical Operations will produce an incorrect result; for example, expiration dates for food items are calculated as over 100 years old, not one or two days old.

    Marginal Operations will cause minor inconvenience, annoyance, or irritation; for example, inventory reports collate dates 00 prior to 99.

- **Establish a "critical event horizon"**

  Business environments are unique, and thus the initial date at which Year2000 challenges will arise varies from business to business. If, for example, you prepare your business forecasts on a three year cycle, the fourth quarter of 1996 might be your critical event horizon. It is unlikely in any business cycle that some Year2000 issues will not surface prior to 1999 or 2000.

- **Provide data administration**

  – Identify the responsibility and scope of migrating the affected data

    - *Exclusive*
      - The data object is created and processed independent of any other business area. There is no data or file integration with other applications.
    - *Primary responsibility*
      - The application defines and creates the application data, and these definitions and data are passed to other business areas which should use them as defined.
    - *Secondary responsibility*
      - The application receives and accepts the data and data definitions from a primary application.
    - *External exposures*

        - The definitions and data are defined within the business and exported outside the enterprise and beyond the scope of the enterprise.
        - Data is created outside your enterprise and then imported into it for use.
    - Determine formats of the data dictionary
    - Determine procedures for changing and entering data definitions
    - Determine procedures for data sharing and use
- **Decide technical and management approaches**
    - Date conversion strategy
        - Decide on a general strategy, or make individual decisions for each date field:
            - Convert to four-digit year?
            - Use windowing technique?
            - Other conversion strategy?
    - Programming standards
    - Hardware and software
    - Development and test procedures
    - Prototyping and parallel development
    - Process and data modelling
    - Data dictionary
    - Documentation structure, layout and standards
    - Reviews
    - Quality assurance procedures
    - Testing methods
    - Automated tools
    - Migration of and bridging to existing Year2000-ready systems
    - Estimated cost of future maintenance
- **Identify project constraints, interfaces and dependencies**
    - End users and customers
        - Availability of test and other data
        - Availability of facilities and services
        - Responsibility for end user tests
        - Other actions
            - Interfaces and dependencies with other projects
            - Supporting services and facilities required
            - Solution Developer automated tools
            - Risks and alternatives
            - Other assumptions

1.3 IBM Assistance

IBM has made available to everyone a comprehensive Year2000 resource guide at no charge. The guide includes a compilation of IBM′s Year2000 findings, recommended approaches and product listings. Also included is a bibliography of other Year2000 publications available throughout the industries. This guide also contains a list of tools that are currently available from both IBM and non-IBM software vendors to assist in identifying potential exposures.

The document, entitled *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, is available in softcopy form on the World Wide Web at URL `http://www.software.ibm.com/year2000/resource.html` or may be ordered through IBM.

1.4 IBM′s Year2000 Offering

Note

You should consult with your local IBM representative for up-to-date information on IBM′s Year2000 Offering.

In developing offerings and support, IBM has taken a customer view of the Year2000 challenge. Every customer needs to take certain steps to achieve Year2000-ready status, depending on the complexity of their IT environment. IBM′s offerings and support are positioned against this customer *needs* framework, shown in Figure 1 on page 8.

Because customer systems are unique, what is required to make a successful transition to Year2000 readiness will be different in every case. There are no *silver bullets*, no shortcuts to take, and no magic formulas to follow. It is IBM′s goal to understand each customer′s environment, and identify where IBM can provide assistance.

Customer needs might include all or some of the following:

- **Project Management**
  - Overall management support for your needs throughout the Year2000 transition process from assessment to execution and follow-on support.
- **Assessment**
  - Awareness of the Year2000 issue
  - Assessing the *ready* status of your IT infrastructure
  - Assessment of the effort required and options available including evaluation, cost, time and resource requirements
  - Development of a strategy to address Year2000 non-ready issues
- **Plan Development**
  - Development of a detailed implementation plan to address all aspects of the environment requiring transition including hardware, software, applications and management processes
- **Execution: Ready Hardware, Ready Software, Ready Applications and Integration Testing**

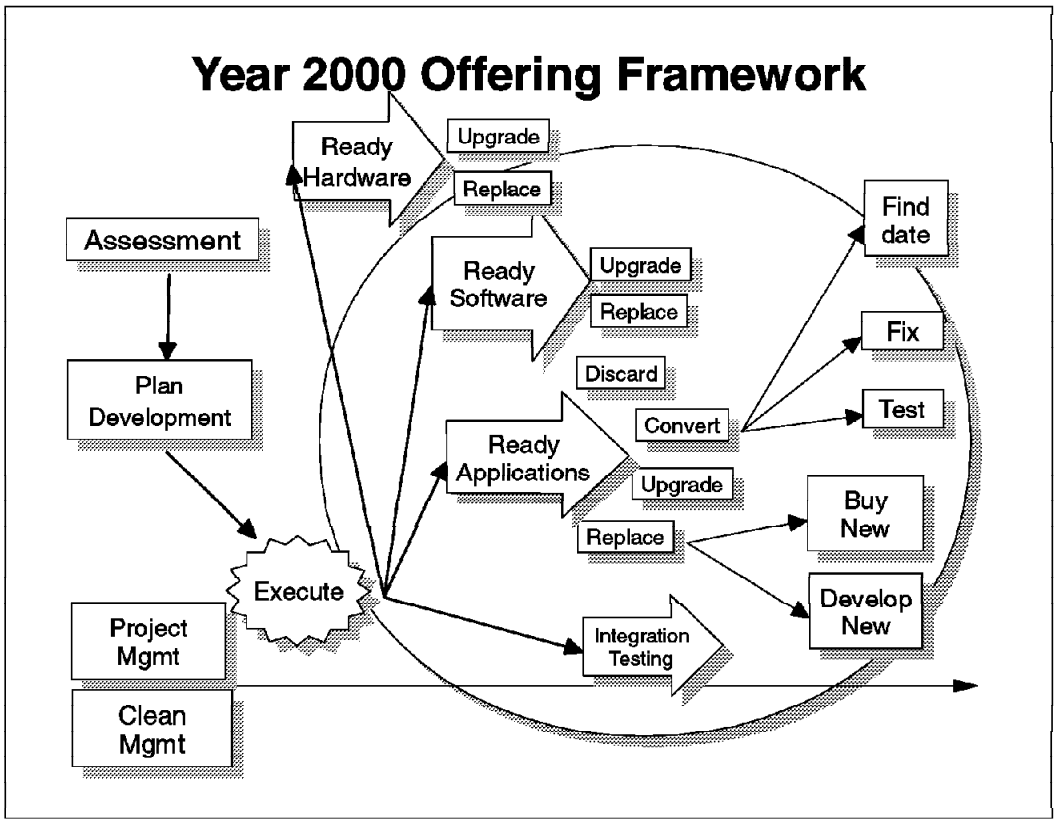


Figure 1. IBM Year2000 Offering Framework

  - Implementation of offerings and support required to address your needs including replacing, upgrading and/or conversion of customer developed software
  - Testing of integrated hardware, software, and application *ready* environment

- **Clean Management**
  - Clean management to prevent the reintroduction of problems once a computing system is Year2000 ready

Additional descriptions of each of the potential needs follow.

Project Management. Given the breadth of the Year2000 challenge, and the possible complexity of the environment, project management of varying degrees may be needed. This may range from guiding only a specific set of implementation tasks, to orchestrating the entire Year2000 transition from assessment through all planning, analysis, implementation and change process activities.

Assessment. It is estimated that **90%** of existing computer systems have Year2000 exposures of some sort. In addition to helping you become aware of the possible extent of the challenge, IBM can assist in assessing your entire systems environment. Date problems may exist at any level of the system including the hardware, the operating system and middleware, and the applications and data.

All IBM hardware products marketed today are Year2000 ready. However, there are older systems no longer being marketed that will not function correctly after the year 2000. You may need assistance to inventory the ready or non-ready status of installed systems hardware as well as any peripherals.

Even though your hardware may be Year2000 ready, you must also assess your operating systems and middleware as well as your applications and data. For a list of IBM Year2000-ready hardware and software products, refer to the *IBM Year 2000 Product Readiness Database*, available through IBM's Year2000 home page (`http://www.ibm.com/year2000/`). The database allows you to select software and hardware products by platform, operating system, and industry, and limit the search to a specific product name or machine type/model or product number. You can request and receive a readiness report on the status of products that match your search criteria. "Readiness Reports" provide information such as the Year2000 readiness status of specific products, the principal technique used to achieve readiness in a given Year2000-ready product, recommended upgrades or replacements for not ready products, and other useful information. This report can be mailed to your e-mail address, often within minutes. If you do **not** have World Wide Web access and have IBM product-specific Year2000 readiness questions, you can get help in obtaining information or a "Readiness Report" by contacting an IBM Technical Support Center.

The final and most difficult part of assessment involves examining your application portfolio. You must first determine if the applications you use were developed in-house or purchased. An inventory tool may be useful at this stage. It will help you understand whether you have matching source and object code, and will identify applications that have not been used for an extended period of time. Many customers are finding that they have code that can simply be discarded rather than converted or upgraded.

If you have source code, the next step will be to do a quick scan of the code to determine how often dates are used. A Year2000 impact analysis tool may be used to help identify date-related variables in program logic and date fields in database files. Many editors and other application development tools also have search capabilities and may be helpful in finding references to dates in programs and data.

Your environment should also be inventoried for non-IT systems that are date sensitive. Environmental control systems, badge readers, elevators and so on may fail if not corrected.

There is another important Year2000 consideration. You should contact your suppliers to make sure they are also preparing adequately. As an example, if your business is dependent upon parts from a particular supplier, you should make sure this supplier will be operational when the date rolls over to the year 2000.

Finally, you may need assistance in understanding your options. Factors to consider include such things as cost, time and resources required.

Plan Development. A detailed plan identifying needed actions, specific tasks, roles and responsibilities, key dependencies and so on, needs to be developed to implement your Year2000 strategy. You must consider your entire IT environment. The detailed plan will address the options to help resolve each unique Year2000 issue. Examples follow.

- If your hardware is not Year2000 ready, you have the option of either upgrading to a newer version of the same platform or purchasing a completely new system.
- If your operating system or middleware is not Year2000 ready, the solution will be to upgrade to a Year2000-ready release. This may be a relatively simple task, or may require significant time and resources, depending on the platform. You will also need to work on the application portfolio. You may choose to convert or rewrite custom applications, upgrade purchased applications to a Year2000-ready version, or replace packaged applications with new solutions. Many customers will combine these options.
- If you have purchased applications from a software vendor, you will need to contact the vendor to determine if the currently installed package is Year2000 ready. If it is not, you must upgrade to a newer, Year2000-ready version of the application, assuming one exists. If the application provider is no longer in business, or if the provider does not have a Year2000-ready version of the application, you will have to replace the application. If you are not satisfied with the functionality of your existing applications, you might also choose to select a new application solution rather than to upgrade the current application.
- Code conversion is an option if you have both the source code and the rights to modify the code (for example, if you originally developed the code in-house, or purchased the rights to the source). You might need to choose this option if you have very specialized application requirements. It may be an expensive option, however, and resources to perform conversions will be limited. Even if you previously used custom code, you may find that replacing it with a packaged solution is a better alternative.

Execution, including ready Hardware, ready Software, ready Applications, and Integration Testing. The execution of the plan involves fulfillment of your needs to get to ready hardware, ready software and ready applications, and to test these in an integrated environment. This may involve replacing or upgrading hardware or system software, making your applications ready by converting, replacing or upgrading them, and finally integrated testing of all components.

IBM has a portfolio of offerings which include products, tools, services, education and financing. Details of these can be obtained from your local IBM representative.

A key issue for the customer in execution will be locating the skills and resources required to complete the work. If you have in-house programmers, the work can be assigned. However, this may mean that Year2000 work will be completed at the expense of other planned projects and, as a result, you may not invest in other new technologies.

If you do not have an IT staff, you will need the assistance of a service provider and/or business partners. Services exist today to help you with a variety of tasks ranging from simple version-to-version hardware or software upgrades, to assistance in locating new software solutions, to customized application and data conversions. The cost associated with these services varies widely and will have to be considered in determining your overall Year2000 strategy.

An additional concern in the implementation of your transition plans is to ensure adequate time for testing once other execution steps have been completed. Testing should be performed at two levels. Individual modules or combinations

of modules that make up applications must be tested to ensure that they function correctly. When this is complete, the total, integrated IT environment must be tested to make sure there are no errors in the interfaces.

You should be sure to allow sufficient time for testing. It may be necessary for you to simulate a full-year business cycle so that all date routines are executed (for example, *end-of-month* and *end-of-year* routines).

Clean Management. You must implement processes to help ensure that date problems are not reintroduced into a *clean* environment. This may require additional, on-going testing of systems.

You will probably want to contact your trading partners to:

- make sure the trading partners put support plans in place to match the actions you will be making;
- test your applications with the data you normally receive from trading partners;
- send new interface files to any trading partners who normally receive them;
- confirm that the new files were used in the trading partner test suites and that they worked correctly.

If you need to process archived data, you may need to test to make sure affected programs still process correctly after the transition.

Chapter 2. VM/ESA′s Year2000 Support

Note

Most of the information in this chapter was extracted from the *Year2000 Technical Reference*, the full text of which may be found on the World Wide Web at URL `http://www.vm.ibm.com/year2000/y2vmtref.html` in its latest, most up-to-date version. Examples and references have been added here for clarity and completeness. The examples provided can be found in Appendix A, “Sample REXX Routines for Year2000” on page 97.

2.1 Releases Supported

Year2000 support is provided in VM/ESA Version 2 Release 2 (V2R2), which became generally available on December 20, 1996. Year2000 support will be included in any future releases of VM/ESA.

IBM does not plan to provide Year2000 support in earlier ESA Feature releases of VM/ESA nor the VM/ESA 370 Feature (Version 1 Release 1.5).

2.2 Overview of VM/ESA′s Year2000 Support

The primary objective of VM/ESA′s Year2000 support is to ensure that the VM/ESA operating system continues to operate correctly in the year 2000 and beyond, as well as providing function to allow applications to do so.

In addition, the ability to define system-wide and user default date formats provides flexibility for installations and/or users for all VM/ESA commands which support multiple date formats.

This support includes:

- Application Programming Interfaces (APIs) support of date formats which include four-digit years.

- Commands to accept and display date formats which include four-digit years.
- Logic to cause existing dates with two-digit years to be resolved with correct four-digit year information.
- Enhancements to existing support for changing the system time of day (TOD) clock to make it easier to IPL a VM/ESA system in the year 2000 or later.

VM/ESA supports running guest systems with a different system date/time than the host VM/ESA system. This enables testing of MVS, OS/390, TPF, VM, and VSE systems with a Year2000 date without affecting the rest of your VM/ESA system.

2.3 Effects of NOT Migrating to a Year2000-Ready Release

The following are examples of how VM systems which are not Year2000-ready (releases prior to VM/ESA V2R2) will be affected by the year 2000.

2.3.1 Control Program (CP)

There is no known loss of CP function caused by not migrating to a Year2000-ready release of VM/ESA. However, application programs and other products which rely upon CP date functions might be affected if they require four-digit year output to execute correctly in the year 2000.

2.3.2 Conversational Monitor System (CMS)

Not all components of CMS will correctly interpret dates in the year 2000 and thereafter if VM/ESA is not migrated to a Year2000-ready release.

- All dates associated with files in both the CMS minidisk file system and the shared file system (SFS) are interpreted as if the year were **19xx**.
- The BEFORE and AFTER options of the FILELIST and LISTFILE commands do not work correctly for files that have a date in the year 2000 or beyond. For example, `FILELIST * * A (BEFORE 12/31/99` would fail to exclude a file whose actual date was January 1, 2000.
- FILELIST sorts by date by default and lists files from newest to oldest. Files with a year of 2000 or beyond are placed at the bottom of the list due to the sort acting on the two-digit year.
- Application programs and other products which rely upon the CMS implementation of the OS TIME macro will not be compatible with MVS output beginning in the year 2000.
- After December 31, 1999, CMS OS Simulation will treat tapes with expiration dates of 99365 and 99366 as having expired. It is a common practice to specify these as “never expire” dates.

2.3.3 Group Control System (GCS)

There will be some loss of GCS function caused by not migrating to a Year2000-ready release of VM/ESA.

- The GCS service aid command `QUERY MODDATE LAST` will not correctly provide a list of all GCS modules compiled on the most recent compilation date if the date of compilation is after 2000 and older GCS modules compiled before 2000 are also loaded.

- Application programs and other products which rely upon the GCS implementation of the OS TIME macro will not be compatible with MVS output beginning in the year 2000.

2.4 IPLing a VM/ESA System in the Year 2000

VM/ESA Version 1 Release 2.1 (V1R2.1) and later systems will IPL correctly with a year of 2000 or later. For VM/ESA V1R2.1 and V1R2.2, APAR VM57927 must be applied. Beginning with VM/ESA V2R1, this capability is included in the base VM system.

In order to change the date of your VM/ESA system, answer ′Yes′ to the ′Change TOD clock (Yes|No)′ prompt during IPL, and change the date and time to the desired date and time. To specify a year of 2000, simply enter ′00′ as the year (′01′ for 2001, and so on).

On systems prior to VM/ESA V2R2, an immediate re-IPL is necessary when the system time of day (TOD) clock is changed at IPL time unless APAR VM60324 is installed (available on VM/ESA Version 2 Release 1 and Version 1 Release 2.2).

If VM60324 is not installed, or is not available for the release of VM that you are running, then an immediate re-IPL is necessary after changing the system TOD clock. By the time the ′Change TOD clock (Yes|No)′ prompt is issued, components of CP have already developed sensitivities to the actual value of the TOD (time of day) clock. If VM60324 is not installed, these sensitivities may cause irregular scheduling and dispatching behavior if the value of the TOD clock is changed by a large amount (more than several minutes) and the system is not re-IPLed.

Figure 2 on page 16 shows a sample IPL of VM/ESA V2R2 changing the date just prior to 2000, to check the “rollover.”

```
 03:44:04 VM/ENTERPRISE SYSTEMS ARCHITECTURE  V2 R2.0  SERVICE LEVEL 9705;
 03:44:08 SYSTEM NUCLEUS CREATED ON 05/06/97 AT 16:25:50, LOADED FROM 220RES
03:44:10
03:44:13 ************************************************************
03:44:14 * LICENSED MATERIALS - PROPERTY OF IBM*                     *
03:44:18 *                                                          *
03:44:21 * 5654-030 (C) COPYRIGHT IBM CORP. 1983, 1996. ALL RIGHTS  *
03:44:25 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,    *
03:44:28 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE  *
03:44:30 * CONTRACT WITH IBM CORP.                                   *
03:44:33 *                                                          *
03:44:36 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.          *
03:44:39 ************************************************************
03:44:42
 03:44:45 HCPZCO6718I Using parm disk 1 on volume 220RES (device 65E6).
 03:44:45 HCPZCO6718I Parm disk resides on cylinders 2367 through 2396.
03:44:45 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
03:44:45       (NOAUTOlog)) or (SHUTDOWN)
03:45:03
03:45:03 NOW 03:45:03 EDT THURSDAY 06/19/97
03:45:03 Change TOD clock (Yes|No)
03:45:18 ==> YES
03:45:18 Set date MM/DD/YY
03:45:38 ==> 12/31/99
03:45:38 Set time HH:MM:SS
03:45:56 ==> 23:30:00
03:45:56 Press "TOD ENABLE SET" key at designated instant.
23:30:00 NOW 23:30:00 EST FRIDAY 12/31/99
23:30:00 Change TOD clock (Yes|No)


                                                        CP Read   VMESA220
```

Figure 2. Changing Date and Time during VM IPL

2.5 Using VM/ESA Guest Systems for Year2000 Testing

All supported ESA feature releases of VM/ESA allow you to run guest systems with a different system date/time than your host VM/ESA system. This facilitates testing of MVS, OS/390, TPF, VM, and VSE systems in the year 2000 without affecting the rest of your VM/ESA system.

Clock requests by a VM/ESA guest system are intercepted by the virtual machine in which the guest is running, and their effect is limited to that virtual machine. The real system clock is never changed by a guest system. When a SET CLOCK (SCK) instruction is issued in a virtual machine, an offset from the real system clock is computed and stored in field VMDEPOCH in the descriptor control block (VMDBK) for that virtual machine. Then, if a STORE CLOCK (STCK) instruction is issued in that virtual machine, the time that is stored is computed by adding the offset in VMDEPOCH to the real system clock value.

To change the date/time of a guest system running under VM/ESA, add an OPTION TODENABLE statement in the directory entry for the virtual machine where the guest will be running. Then simply change the system date and time for your guest system.

2.6 Checking for Year2000 Support

Programming interfaces have been provided in CP, CMS, and GCS to indicate whether Year2000 support is present. CMS and GCS indicate that Year2000 support is present only if running on a level of CP that also includes Year2000 support.

2.6.1 CP

Year2000 support is indicated by the bitmap provided by Diagnose X'00'. The bitmap is as follows:

- Version 2 Release 1 (previous)
  - X'7FF8'
- Year2000 Support
  - X'7FFC'
- Version 2 Release 2 (includes Year2000 support)
  - X'7FFE'

Sample

See Figure 34 on page 98 for a sample REXX program that uses diagnose X'00' to test for the presence of CP Year2000 support.

2.6.2 CMS

The presence of Year2000 support may be determined through the CSL routine DMSERP (Extract/Replace) or the REXX CMSFLAG function. For DMSERP, information name YEAR2000_SUPPORT has been added to the General System Set. This flag is set to '1' if Year2000 support in **both** CP and CMS is present.

Sample

See Figure 36 on page 100 for a sample REXX program that uses DMSERP to check for Year2000 support.

Flag YEAR2000 has been added to the REXX CMSFLAG function. This flag returns 1 if Year2000 support is present in **both** CP and CMS, and returns 0 if Year2000 support is absent from either CP or CMS.

Sample

See Figure 35 on page 99 for an example of safe use of CMSFLAG to detect Year2000 support.

2.7 GCS

The FLS macro can be used to determine the presence of Year2000 support. A new bit, FLS2000, has been defined in the FLSFLG byte. As with CMS, GCS will indicate that Year2000 support is present only if CP's Year2000 support is also present.

2.8 Date Formats for CP and CMS Commands

Default date formats may be set on a system-wide and user (virtual machine) basis for CP and CMS commands which provide date output.

The following are supported date formats for CP and CMS commands:

Table 1. Date Format Definitions for CP and CMS Commands

Name (Abbreviation in CAPS)	Format
SHOrtdate	*mm/dd/yy*, *mm/dd*, or *yy/mm/dd* (same as earlier releases of VM/ESA)
FULldate	*mm/dd/yyyy* or *yyyy/mm/dd*
ISOdate	*yyyy-mm-dd*
SYSdefault	Use the system-wide default setting (for setting user defaults)
VMDate	Use the user's default setting (for FILELIST and RDRLIST)

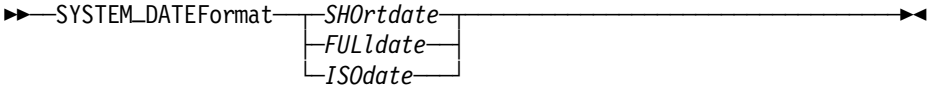
2.9 Setting Date Format Defaults

A new system configuration file statement, SYSTEM_DATEFORMAT, allows the system-wide default date format to be set. New commands SET DATEFORMAT and QUERY DATEFORMAT allow the default date formats for both the system and individual users to be set and queried. A new directory statement, DATEFORMAT, allows a user's (virtual machine's) default date format to be specified in the directory.

Sample

The ISPF sample **ISPB2042 EXEC** in Figure 45 on page 118 shows usage of the QUERY and SET DATEFORMAT commands.

2.9.1 SYSTEM_DATEFormat Statement



2.9.1.1 Purpose

Use the SYSTEM_DATEFORMAT statement to set the system-wide default date format for commands that provide multiple date formats.

2.9.1.2 How to Specify

Include as many statements as needed; they are optional. You can place SYSTEM_DATEFORMAT statements anywhere in the system configuration file. If you specify more than one SYSTEM_DATEFORMAT statement, the last statement overrides any previous specifications.

2.9.1.3 Operands

SHOrtdate

specifies that dates in command responses be displayed in *mm/dd/yy*, *mm/dd*, or *yy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yy* is the two-digit year.

FULldate

specifies that dates in command responses be displayed in *mm/dd/yyyy* or *yyyy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yyyy* is the four-digit year.

ISOdate

specifies that dates in command responses be displayed in *yyyy-mm-dd* format, where *yyyy* is the four-digit year, *mm* is the month, and *dd* is the day of the month.

2.9.1.4 Usage Notes

1. If the SYSTEM_DATEFORMAT statement is not in the system configuration file, then the system-wide default date format is SHORTDATE.
2. The choice between the possible SHORTDATE or FULLDATE formats is dependent upon the command or routine that displays or generates the date.

2.9.1.5 Examples

1. To define a default date format of *mm/dd/yy*, *mm/dd*, or *yy/mm/dd*, use the following SYSTEM_DATEFORMAT statement:

   ```
   SYSTEM_DATEFORMAT SHORTDATE
   ```

2. To define a default date format of *mm/dd/yyyy* or *yyyy/mm/dd*, use the following SYSTEM_DATEFORMAT statement:

   ```
   SYSTEM_DATEFORMAT FULLDATE
   ```

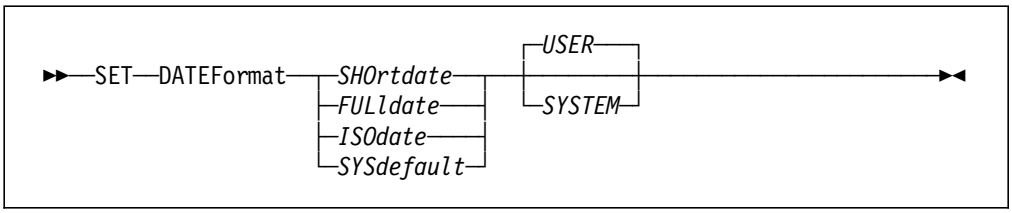
3. To define a default date format of *yyyy-mm-dd*, use the following SYSTEM_DATEFORMAT statement:

   ```
   SYSTEM_DATEFORMAT ISODATE
   ```

2.9.1.6 Migration Aids

Previous releases of VM did not let you set a default date format. Therefore, there are no macroinstructions in your HCPSYS or HCPRIO files that you need to migrate to a system configuration file.

2.9.2 SET DATEFormat Command



2.9.2.1 Authority

Privilege Class: B,G

2.9.2.2 Purpose

Use SET DATEFORMAT to set the default date format for commands which provide multiple date formats. Default date formats may be set for individual users or the entire system.

2.9.2.3 Operands

SHOrtdate

specifies that dates will be displayed in *mm/dd/yy*, *mm/dd*, or *yy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yy* is the two-digit year.

FULldate

specifies that dates will be displayed in *mm/dd/yyyy* or *yyyy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yyyy* is the four-digit year.

ISOdate

specifies that dates will be displayed in *yyyy-mm-dd* format, where *yyyy* is the four-digit year, *mm* is the month, and *dd* is the day of the month.

SYSdefault

specifies that the default date format for this user will be set to the system-wide default.

USER

specifies that the default date format is being set for the user.

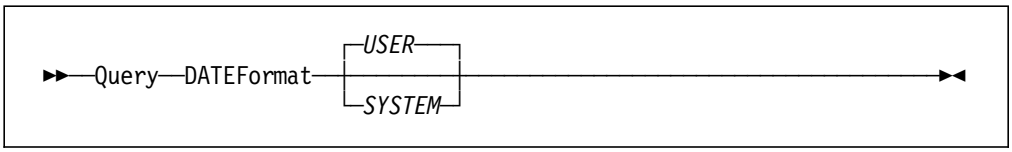
SYSTEM

specifies that the default date format is being set for the entire system.

2.9.2.4 Usage Notes

1. If SET DATEFORMAT is not specified for an individual user and there is no DATEFORMAT statement in the user's directory entry, the default date format for that user is the system-wide default.
2. SYSDEFAULT is not valid with SYSTEM.
3. SYSTEM is valid only for class B.
4. If the default date format for a user is set to (or defaults to) the system-wide default date format, and the system-wide default is changed, the user continues to see the old system-wide default as the user default until the user either logs off and logs back on or issues SET DATEFORMAT SYSDEFAULT to switch to the new system-wide default.

2.9.3 QUERY DATEFormat Command



2.9.3.1 Authority

Privilege Class: G

2.9.3.2 Purpose

Use QUERY DATEFORMAT to display the current default date format for the system or an individual user.

2.9.3.3 Operands

USER

tells CP to display the default date format for this user.

SYSTEM

tells CP to display the default date format for the system.

2.9.3.4 Usage Notes

1. If your user default date format is set to (or defaults to) the system-wide default date format, and the system-wide default is changed, the response to QUERY DATEFORMAT USER will continue to indicate the old system-wide default as your user default until you either log off and log back on or issue the SET DATEFORMAT SYSDEFAULT command to switch to the new system-wide default.

2.9.3.5 Responses

If you issue QUERY DATEFORMAT USER, and your user default date format is set to SHORTDATE, FULLDATE, or ISODATE, you will get one of the following responses:

```
User Dateformat = SHORTDATE
User Dateformat = FULLDATE
User Dateformat = ISODATE
```

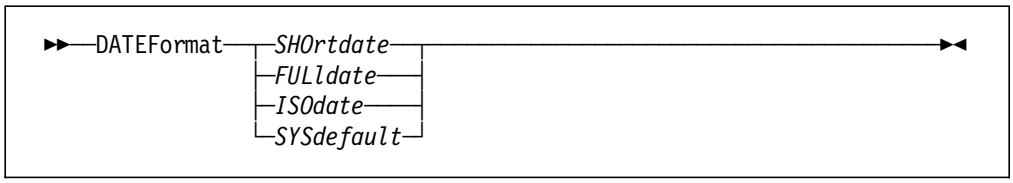
If you issue QUERY DATEFORMAT USER, and your user default date format is set to (or defaults to) the system-wide default, you will get one of the following responses:

```
User Dateformat = SHORTDATE (SYSDEFAULT)
User Dateformat = FULLDATE (SYSDEFAULT)
User Dateformat = ISODATE (SYSDEFAULT)
```

If you issue QUERY DATEFORMAT SYSTEM, you will get one of the following responses:

```
System Dateformat = SHORTDATE
System Dateformat = FULLDATE
System Dateformat = ISODATE
```

2.9.4 DATEFormat Directory Statement



2.9.4.1 Purpose

The DATEFORMAT statement specifies a user's default date format for commands that provide multiple date formats.

2.9.4.2 How to Specify

One DATEFORMAT statement is allowed in a user or profile entry. A DATEFORMAT statement in a user entry overrides a DATEFORMAT specification in a profile entry.

If you specify the DATEFORMAT statement, it must precede any device statements that you specify in a profile or user entry.

2.9.4.3 Operands

SHOrtdate

specifies that dates in command responses be displayed in *mm/dd/yy*, *mm/dd*, or *yy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yy* is the two-digit year.

FULldate

specifies that dates in command responses be displayed in *mm/dd/yyyy* or *yyyy/mm/dd* format, where *mm* is the month, *dd* is the day of the month, and *yyyy* is the four-digit year.

ISOdate

specifies that dates in command responses be displayed in *yyyy-mm-dd* format, where *yyyy* is the four-digit year, *mm* is the month, and *dd* is the day of the month.

SYSdefault

specifies that the default date format for this user be set to the system-wide default. The system-wide default date format may be set in the system configuration file with the SYSTEM_DATEFORMAT statement, or with the CP SET DATEFORMAT command.

2.9.4.4 Usage Notes

1. The choice between the possible SHORTDATE or FULLDATE formats is dependent upon the command or routine that displays or generates the date.
2. If you omit the DATEFORMAT statement when you code a user's directory entry, the default date format for that user will be the system-wide default (SYSdefault). A user may change his or her own default date format with the CP SET DATEFORMAT command.
3. If the user's default date format is set to SYSDEFAULT, the system-wide default date format that is in effect at logon time will be used until the user logs off or issues the SET DATEFORMAT command. If the system-wide default date format is changed, the user must log off and log back on or issue the SET DATEFORMAT SYSDEFAULT command to switch to the new system-wide default.

2.9.5 DEFAULTS Command

The DEFAULTS command allows the default date format to be changed for FILELIST and RDRLIST. The date format specified on the DEFAULTS SET command will override the user′s (virtual machine′s) default date format setting.

In addition to SHOrtdate, FULldate, and ISOdate, VMDate is also supported as a date format option for FILELIST and RDRLIST. When VMDate is specified as the default, the user′s (virtual machine′s) current default date setting will be used.

2.9.6 Date Formats for Individual Commands

Date formats may be specified on each CP and CMS command that provides dates as part of its output. If a date format is specified on a command, it overrides all default date format settings for that execution of the command.

2.9.7 Compatibility

To maintain compatibility with earlier releases of VM/ESA, the system-wide and user default date formats are initialized as indicated below. Also shown is the initial CMS DEFAULTS setting for the FILELIST and RDRLIST commands. These settings cause all commands to initially provide date output in the same format as earlier releases of VM.

Table 2. Initial Date Format Settings

Type	Date Format Default
System-wide	SHOrtdate
User (Virtual Machine)	SYSdefault
CMS DEFAULTS for FILELIST and RDRLIST	VMDate

2.10 Extended Commands

2.10.1 CP

The following CP commands have been changed to accept date format operands and provide dates with four-digit years. If no date format operand is specified, the user′s default date format setting is used.

- CPLISTFILE
- QUERY CPLEVEL
- QUERY IMG
- QUERY NLS
- QUERY NSS
- QUERY READER/PRINTER/PUNCH ALL
- QUERY TIME
- QUERY TRFILE
- QUERY UCR

2.11 CMS Commands

The following CMS commands have been changed to accept date format operands and/or provide dates with four-digit years. If no date format operand is specified, the user's default date format setting is used (except in the case of FILELIST and RDRLIST, where the DEFAULTS setting is used).

- FILELIST
- IDENTIFY
- LISTFILE
- NOTE
- RDRLIST

Note: Enhancements to CMS Pipelines stages are listed in section 2.12, Application Programming Interfaces (APIs) under CMS Pipelines on page 27.

2.11.1 GCS Commands

Output from the following GCS commands has been changed to provide dates with four-digit years:

- IPL Message
- QUERY MODDATE

2.12 Application Programming Interfaces (APIs)

The following VM/ESA Application Programming Interfaces have been changed or introduced to support four-digit years or to include century information:

- CP Diagnose Codes
  - Diagnose X'00'

    The bitmap now shows if Year2000 support is present.

    Sample

    See the sample **Y2000 EXEC** in Figure 34 on page 98 and the ISPF sample **ISPA2042 EXEC** in Figure 44 on page 116 for examples of using diagnose X'00'.

  - Diagnose X'0C'

    There are no changes to diagnose X'0C', but new diagnose X'270' extends diagnose X'0C' output to show dates with four-digit years.

  - Diagnoses X'14' and X'D8'

    The SFBLOK buffer returned by certain subcodes of these diagnose codes includes a one-byte century indicator at displacement X'8F'. This byte contains a hexadecimal representation of the century portion of the year (X'13' ='19', X'14' ='20' and so on).

Note

The **SPORDER EXEC** (CP Utility), for sorting reader files in chronological order, uses diagnose X′14′. Since this utility was written before the new *century* flag existed, it currently does not work for reader files created in the 21st century.

IBM VM development has already been made aware of this.

- Diagnose X′84′

  A new operation, DATEFMT, allows a user′s default date format to be replaced in that user′s CP object directory entry.

- Diagnose X′BC′

  Fields containing the spool file date in four-digit year format (*mm/dd/yyyy* and *yyyy-mm-dd*) have been added to the end of the data buffers returned by subcodes X′0000′ and X′0004′.

- Diagnose X′270′

  New diagnose code X′270′ returns the same information as diagnose X′0C′ with the addition of the date in four-digit year format.

- Callable Services Library (CSL) Routines

  The tables below list the CMS file system CSLs by category:

  - CSL routines that provide the SHORTDATE, FULLDATE and ISODATE formats of the date(s) in a buffer specified by the user

CSL routine	Description
DMSEXIST	Checks for the existence of a file, directory, or external object and retrieves information about the file, directory, or external object.
DMSGETDI	Reads directory records after a directory has been opened using Open Directory (DMSOPDIR).
DMSRDCAT	Returns catalog information.

  - CSL routines that will add new keywords to an existing keyword parameter to specify the format of the input or output date

CSL routine	Description
DMSCLBLK	Closes files that have been opened previously using Open Blocks (DMSOPBLK).
DMSCLDBK	Closes files that have been opened previously using the Open Data Block routine (DMSOPDBK)
DMSCLOSE	Closes files that have been opened previously using the Open routine (DMSOPEN).
DMSCRDIR	Creates a directory once you have been enrolled as a user in an SFS file pool.
DMSCRFIL	Creates a new empty file in an SFS directory.
DMSCROB	Creates an external object in an SFS directory.
DMSEXIDI	Checks for the existence of a directory and retrieves directory-related information.
DMSEXIFI	Checks for the existence of a file and retrieves file-related information.

CSL routine	Description
DMSOPBLK	Prepares a file for subsequent use by DMSRDBLK, DMSWRBLK, or DMSCLBLK.
DMSOPDBK	Prepares a file for data block I/O operations.
DMSOPEN	Prepares a file for subsequent reading or writing of data records.
DMSTRUNC	Deletes records from the end of an existing minidisk or SFS file.

– CSL routines that will have a new keyword parameter added to the end of the parameter list to specify the format of the input or output date

CSL routine	Description
DMSENUSR	Enrolls one or more SFS users or one-byte file system in a specified file pool.
DMSGETDA	Reads one directory record after a directory has been opened using Open Directory (DMSOPDIR) with an intent of SEARCHALL.
DMSGETDF	Reads one directory record after a directory has been opened using Open Directory (DMSOPDIR) with an intent of FILE.
DMSGETDS	Reads one directory record after a directory has been opened using Open Directory (DMSOPDIR) with an intent of SEARCHAUTH.
DMSGETDX	Reads one directory record containing extended file attributes after a directory has ben opened using Open Directory (DMSOPDIR) with an intent of FILEEXT.

Sample

See Figure 37 on page 102 for an illustration of how to call DMSEXIFI with a date format specification.

- DMSERP Changes for Year2000

  For CSL routine **DMSERP (Extract/Replace)**, information names have been added to the information sets which are related to date/time, and a new information name has been added to indicate the presence of Year2000 support in the user's virtual machine.

  Sample

  See Figure 36 on page 100 for examples of the DMSERP enhancements.

- VMTMDTS - DateTimeSubtract CSL Routine

  A new CMS multitasking function, **DateTimeSubtract**, provides a comprehensive facility for the conversion and manipulation of date/time stamps. DateTimeSubtract is located in the VMMTLIB CSL. For a complete description of the DateTimeSubtract function, see *VM/ESA CMS Application Multitasking*, SC24-5766, or issue HELP ROUTINE DATETIMS.

  Sample

  See the samples listed in A.7, "Sample CSL VMTMDTS (DateTimeSubtract)" on page 106 for examples using this new CSL routine.

- Macros

- DIRBUFF

  The record map generated by this macroinstruction has been expanded to include the date formats with four-digit years that are returned by a Get Directory request.

- EXSBUFF

  The record map generated by this macroinstruction has been expanded to include the date formats with four-digit years that are returned by an Exist request for a file or directory.

- FSSTATE

  When the specified file exists, the address of a copy of its File Status Table (FST) information is pointed to by Register 1. See File Status Table on page 28 for a description of the change that has been made to the FST to permit dates in the year 2000 and beyond to be represented.

- FSTD

  The DSECT for the FST control block returned by this macro has been updated to include the definition of the century indicator bit in the FST Flag Byte.

- TIME Macro (OS Simulation, GCS)

  An indicator for the first two digits of the year has been added to the output for the simulated MVS TIME macro.

- CMS Pipelines

  The following CMS Pipelines stage commands have been enhanced to accept new date format options SHORTDATE, FULLDATE, and ISODATE, and provide date output in the appropriate format:

  - AFTFST
  - STATE
  - STATEW
  - FMTFST

  A new, experimental Pipeline filter, DATECONVERT, has recently been made available in the CMS Pipelines Runtime Library Distribution on the World Wide Web at URL `http://pucc.princeton.edu/~pipeline/`. As the name implies, DATECONVERT provides a Pipelines interface to the DateTimeSubtract CSL routine to perform date conversion and validation, thus also permitting date arithmetic to be accomplished in a pipeline. It supports all of the date formats recognized by DateTimeSubtract and by the REXX DATE() function, plus some others. A parameterizable windowing capability controls the interpretation of dates with two-digit years.

  An overview of the capabilities and syntax of the DATECONVERT stage may be found at URL `http://pucc.princeton.edu/dateconv.html/`; complete documentation in printable form may be downloaded from the Pipelines Runtime Distribution site.

  The DATECONVERT stage is part of the additional Year2000 Readiness features described in the preview announcement of VM/ESA Version 2 Release 3 issued by IBM on October 7, 1997.

- REXX

  The following REXX functions have been updated:

– CMSFLAG()

  Flag YEAR2000 has been added to the CMSFLAG function. This flag returns 1 if Year2000 support is present in both CP and CMS, and returns 0 if Year2000 support is not present in either CP or CMS.

  Sample

  See the sample **Y2K EXEC** in Figure 35 on page 99 for a simple example of using this new flag.

– DIAG() and DIAGRC()

  The DIAG and DIAGRC functions have been enhanced to include support for diagnose X'270'.

– DATE()

  The DATE() function has been enhanced to provide date format conversion capability. All existing DATE formats are supported. The 'C' (Century) and 'J' (Julian) formats are supported as input formats for conversion, but not as conversion output formats. For compatibility purposes, existing support of DATE('C') and DATE('J') is unchanged.

  Sample

  See the sample **HOLIDAYS EXEC** in Figure 43 on page 111 for a practical example of using this new date conversion capability.

- EXEC2

  – &FULLDATE

    The &FULLDATE statement has been added. This statement returns the date in Coordinated Universal Time (UTC) in the form *yyyy/mm/dd*.

  – &ISODATE

    The &ISODATE statement has been added. This statement returns the date in Coordinated Universal Time (UTC) in the form *yyyy-mm-dd*.

2.13 Miscellaneous Changes

VM/ESA Application Programming Interfaces which have been changed:

- CMS GUI Facility

  The FILELIST and RDRLIST functions of the CMS Desktop have been enhanced to display correct four-digit year information. New date format options that have been added to VM/ESA are not supported by the CMS GUI Facility.

- File Status Table (FST)

  When you access a minidisk or SFS directory, a file directory is stored in your virtual machine. The entries in the file directory for each CMS file are called the File Status Tables (FSTs). The FSTs describe the attributes of each file. One of the attributes of a file is the date and time it was last updated. This timestamp is stored in a six byte field, in the format *yy mm dd hh mm ss*, with each byte holding two decimal digits. A century flag (FSTCNTRY, X'08') has been added to the FST flag byte (FSTFLAGS) at decimal displacement 31 in both the extended and standard forms. This flag represents the century portion of the year in which the file was last written or

updated. If the bit is off, then the year is 19xx; if it is on, then the year is 20xx. This solution supports years in the range 1900-2099.

- DMSPLU

  DMSPLU accepts dates with four-digit years in the form *mm/dd/yyyy*. DMSPLU continues to accept dates with two-digit years (*mm/dd/yy*) as well. When a two-digit year is specified, a sliding window with a range -50 to +49 from the current year is used to determine the century portion of the specified year.

  Prior to VM/ESA V2R2, DMSPLU accepted years in the range (19)80 through (19)99 only. Beginning with VM/ESA V2R2, DMSPLU accepts dates beyond 1999 (and before 1980). Although there are no restrictions on the four-digit years accepted by DMSPLU, specification of a date outside the range of years 1900-2099 yields unpredictable results since the CMS file system does not support dates outside that range.

  Sample

  See Figure 40 on page 105 for an example of DMSPLU usage.

  Note: DMSPLU remains an unsupported utility.

- SPTAPE

  When a spool file is SPTAPE DUMPed, the century portion of the year is stored in hexadecimal at displacement X′8F′ in the SFBLOK that is written to tape (the spool file date is currently stored in format *mm/dd/yy* at displacement X′40′).

  When a spool file is SPTAPE LOADed, the century portion of the year is included in the TOD clock value in the SPFBK of the spool file that is being loaded.

- DVF

  The Dump Viewing Facility has been changed to properly support dates in the year 2000 or later.

  When run on a VM/ESA CP which contains Year2000 support (V2R2), the TRACERED and VIEWSYM commands (and specifically the VDATE EXEC which is called by the VIEWSYM command) interpret two-digit years with a sliding window of -50,+49.

  If the new level of the Dump Viewing Facility (containing this support) is run on a VM/ESA CP that does not have Year2000 support (pre-V2R2), dates are interpreted as they were previously.

Chapter 3. The Case Study

3.1 Overview

The objectives of the case study were to record real life experiences about the preparation of VM systems for the year 2000. The scope of the study included both operating system migration and the review and preparation of program products and applications. Two “real world” customer systems were made available for the study.

3.2 Case Study Systems - Description

The two systems selected were chosen to best meet the requirements of the study; one for modelling the migration to VM/ESA Version 2 Release 2, the other to provide a representative set of program products, third party products and locally developed applications.

3.2.1 Operating System Migration

This system was made available by Telecom New Zealand Ltd. The existing operating system is VM/ESA Version 1 Release 2.2, Service Level 9510, and has not been modified in any way.

This platform is used primarily for intensive batch processing and has a small interactive user population. The principal application is written mainly in REXX and PL/I, with a small number of Assembler modules. Dialog management is provided by ISPF and CMS Fullscreen. DFSORT is used extensively for the (re-)organization of data.

Status files are maintained throughout the application for the batch “units of work.” The status file entries include date and time stamps and are often used in calculation and comparison operations, for example to calculate the age of a unit of work, or to determine if one unit of work is older than another.

The WAKEUP module is intrinsic to the application set for interrupt and timed event handling.

3.2.1.1 Objectives of the Study

For this system, the case study objectives were:

- to perform a complete and successful migration of the base operating system and user environment to VM/ESA Version 2 Release 2;
- to assess the “difficulty” of the migration exercise;
- to record and resolve any issues with system administration, operation or reliability that resulted from the conversion;
- to migrate the user community to the position where the features of VM/ESA Version 2 Release 2 could be exploited, in particular running CMS 13 in XC mode;
- to review the ESAMIGR tool;

- to perform a high level assessment of the impact on the application resulting from the change of operating system.

3.2.2 Application Study

The second case study was provided by the Government of the Province of British Columbia, Canada. This system was already running VM/ESA 2.2.0 at Service Level 9705. Both CP and CMS have some modifications but, for the most part, those modifications are independent of Year2000 considerations and will not be mentioned further.

At the Government of the Province of British Columbia, VM supports a broad range of interactive and batch workloads. Interactive workload includes OfficeVision/VM and various dialog management interfaces to data stored in DB2 for VM (formerly named SQL/DS) databases. Typical batch processing includes report generation (again based on data stored in DB2) and AFP printing; to some extent, existing Web serving can also be classified as batch work. Applications have been developed in REXX, PL/I, CSP, C, COBOL and Assembler.

In additional to the above suite of IBM products, products from additional vendors of VM solutions were installed. Products from MiraSoft, Sterling Software and Syncsort are key to the proper functioning of many applications. Several members of the VM:Manager group of products from Sterling Software are heavily used by applications, including VM:Secure, VM:Batch, VM:Schedule and VM:Tape. dVMCF provided by MiraSoft is vital to the functioning of the current system complex. The Syncsort sorting package is also widely used.

Several applications that contained date sensitivities and were felt to be representative of applications that might be found on “average” VM systems were selected for study and migration to a Year2000-ready state. The intent is to eventually run this system with the default system date format set to ISODATE, and thus the changes required to support alternate date formats were also considered.

Chapter 4. Migrating from VM/ESA Version 1 Release 2.2 to VM/ESA Version 2 Release 2

4.1 Summary

This migration is an exercise you must undertake to attain Year2000 compliance at the operating system level.

However, the news is good. Our experience was that the migration is relatively quick and simple to perform and, provided that the necessary planning is performed beforehand, can be achieved with minimal disruption to applications and users.

In addition, there are several features in VM/ESA Version 2 Release 2 that can further simplify operating system customization and administration tasks.

Further information about Year2000 issues, VM/ESA Version 2 Release 2 support for Year2000, and program product readiness status is available from one or more of the following sources:

- The VM Home Page at URL `http://www.vm.ibm.com/`
- The VM and Year2000 Page at URL `http://www.vm.ibm.com/year2000/`
- YR2000VM PSP bucket
- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, also available at URL `http://www.software.ibm.com/year2000/resource.html`
- this book !

4.2 Compatibility Considerations

There are two topics that deserve separate attention when planning your migration:

- CMS 13
- user and system DATEFORMAT

4.2.1 CMS 13

CMS 13 is the intended VM/ESA Version 2 Release 2 CMS, and is the only version that has all the features for Year2000 readiness. We recommend that you plan to migrate your user community to CMS 13.

Note

CMS 13 will run in XA or XC mode virtual machines only; you cannot run CMS 13 in a 370 mode virtual machine.

Ideally, applications with 370 dependencies will be converted or upgraded to run in XA or XC virtual machines without the need for 370 support. However,

VM/ESA Version 2 Release 2 does have facilities for accommodating virtual machines or applications with 370 mode dependencies:

CP 370ACCOM

Use the CP command SET 370ACCOM ON | OFF to control CP support for virtual machines using applications that do not execute correctly in XA or XC mode virtual machines. More information about this command can be found in the *VM/ESA CP Command and Utility Reference*, SC24-5773.

CMS CMS370AC

The CMS command SET CMS370AC ON | OFF (see the *VM/ESA CMS Command Reference*, SC24-5776) controls additional CMS support for 370 applications that will not execute properly in XA or XC mode virtual machines. Note that CMS370AC support is a superset of the basic CP 370ACCOM support; enabling CMS370AC automatically enables CP 370ACCOM if it is not already active.

These options are presented in order of preference. CMS370AC provides more extensive support for 370 mode than CP 370ACCOM, and thus better toleration and support of 370 applications, but at the cost of introducing more processing overhead. You should enable CMS370AC support only for 370 programs which do not execute successfully with just the CP 370ACCOM support active.

Also included as part of the system product is the *CMS Migration Utility Feature*. This is a complete CMS 11 which is intended to be used only while modifying applications or products during migration to ESA capability.

Information about migrating applications and virtual machines to run in XA or XC mode, and the *CMS Migration Utility Feature* may be found in the *VM/ESA Conversion Guide and Notebook*, SC24-5831.

4.2.2 DATEFORMAT

In the absence of any other specification, the default SYSTEM DATEFORMAT is SHORTDATE (that is, *mm/dd/yy*). In other words, this is the same as the date information available prior to the introduction of Year2000 support. The system default dateformat may be controlled by an entry in the system configuration file, or by privileged command.

This format will provide the greatest level of compatibility as you migrate your users and applications to VM/ESA Version 2 Release 2. This date format does not protect you from the well documented effects of dates in the year 2000 and beyond; and you, your applications, and your users should plan to adopt a date format that does provide the necessary protection. Windowing techniques are an alternate approach.

Recommendation

We recommend that one of the goals of your Year2000 migration be the ultimate adoption of ISODATE (that is, *yyyy-mm-dd*) as the default SYSTEM DATEFORMAT since this date format:

- is Year2000 enabled

  and

- is consistent across all commands and responses.

The change in format from SHORTDATE to ISODATE is significant, and will probably not be transparent to applications and users.

Using standard CP facilities (see 2.9, “Setting Date Format Defaults” on page 18), the DATEFORMAT can be set for individual virtual machines either by directory entry or command.

Refer to *VM/ESA Planning and Administration*, SC24-5750, and to *VM/ESA CP Command and Utility Reference*, SC24-5773, for further information about system configuration file entries and the CP commands for controlling date formats at system and virtual machine level.

4.3 CP

The case study system does not run guest operating systems and has no modifications to CP. It runs from a CPLOAD MODULE built using standard facilities.

If you are migrating from VM/ESA Version 1 Release 2.2 or an earlier release of VM, you should consider the following questions:

- Do you have CP modifications?
- Have you re-structured your privilege classes?

If you answered yes to either of these questions, you should know that VM/ESA Version 2 Release 2 includes many features to support the dynamic implementation of CP customization or extensions wherever possible. Exploiting these facilities can greatly simplify the administration and implementation of CP modifications.

- Do you have guest operating systems that take advantage of V=R Recovery Support?

If you do, you should know that preferred guest support now includes *dynamic V=R recovery*; if **NO** devices are defined in HCPRIO ASSEMBLE, then all dynamically defined devices dedicated to a V=R guest will survive over software IPLs. Note that if any devices for a preferred guest are defined in the HCPRIO ASSEMBLE file, then **all** dedicated devices must be included there.

For more information about new features in VM/ESA Version 2 Release 2 and detailed information on preparing for migration tasks, see:

- *VM/ESA Introduction and Features Summary*, SC24-5746
- *VM/ESA Planning and Administration*, SC24-5750
- *VM/ESA Conversion Guide and Notebook*, SC24-5831

Precautions

The case study was performed in a laboratory environment on a second level guest system. If you are working on a first level system, we recommend that you take a complete system backup before starting your system migration work.

It is possible to destroy spool files on your system during the move to VM/ESA Version 2 Release 2 or when backing out from VM/ESA Version 2 Release 2 to an earlier release of VM, particularly when migrating from pre-ESA releases.

To avoid problems in this area, you should read the sections of the *VM/ESA Conversion Guide and Notebook*, SC24-5831, that discuss how to manage spool files between releases of VM, and identify any APARs required for or during system migration.

The VM/ESA Version 2 Release 2 software was loaded on a separate, dedicated user ID. The same user ID was used for the build and generation of the new CPLOAD MODULE.

See the *VM/ESA Service Guide*, SC24-5749, for more information and guidance on (re-)generating CP for your system.

Prior to starting the build of the new CP, we made sure that we had a secondary parm disk area available, and that it held files of the same level as those currently in production. This disk would have been used for backing out if we had needed to return to the earlier version of VM/ESA.

The new CPLOAD module was generated without modifications, using the standard facilities. Generation was straightforward and without error. We copied the new module onto the production parm disk and issued a SHUTDOWN command to terminate the system.

The system residence pack was IPLed to invoke the SAPL (stand alone program loader). The PARM disk was listed, and the new CPLOAD MODULE selected for loading. During IPL, we wanted to control the intialization of server virtual machines, and selected options **WARM** and **NOAUTO** (as shown in Figure 3 on page 37) to allow us to do this.

```
20:36:10 VM/ENTERPRISE SYSTEMS ARCHITECTURE  V2 R2.0  SERVICE LEVEL 9705;
20:36:12 SYSTEM NUCLEUS CREATED ON 07/02/97 AT 03:02:44, LOADED FROM SYSRES
20:36:12
20:36:12 ************************************************************
20:36:12 * LICENSED MATERIALS - PROPERTY OF IBM*                     *
20:36:12 *                                                          *
20:36:12 * 5654-030 (C) COPYRIGHT IBM CORP. 1983, 1996. ALL RIGHTS  *
20:36:12 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,    *
20:36:12 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE  *
20:36:12 * CONTRACT WITH IBM CORP.                                   *
20:36:12 *                                                          *
20:36:12 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.          *
20:36:12 ************************************************************
20:36:12
20:36:12 HCPZCO6718I Using parm disk 1 on volume SYSRES (device 0710).
20:36:12 HCPZCO6718I Parm disk resides on cylinders 30 through 49.
20:36:12 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
20:36:12       (NOAUTOlog)) or (SHUTDOWN)
20:36:24 WARM NOAUTO
20:36:24 NOW 20:36:24 NZS THURSDAY 07/03/97
20:36:24 Change TOD clock (Yes|No)
20:36:26 NO
20:36:26 The directory on volume SYSRES at address 0710 has been brought online.
20:36:27 HCPWRS2513I
20:36:27 HCPWRS2513I Spool files available      320
20:36:29 HCPWRS2512I Spooling initialization is complete.
20:36:33 DASD 0710 dump unit CP IPL pages 2103
20:36:33 There is no logmsg data
20:36:33 FILES: 0016 RDR, 0001 PRT,   NO PUN
20:36:33 LOGON AT 20:36:33 NZS THURSDAY 07/03/97
```

Figure 3. VM Migration Case Study - IPL Options

No errors were encountered during the IPL.

Notes

IPL processing performs any spool file conversion that is required; however, you may want to take additional precautions for any special or important spool files you may have (see the *VM/ESA Conversion Guide and Notebook*, SC24-5831, for assistance).

At this stage, the system had no new CMS, we were running CMS 11 on CP V2R2 as shown in Figure 4 on page 38.

To avoid the possible introduction of additional issues, the system dateformat was left to the default, that is SHORTDATE, during the initial series of tests. For the same reason, the tests were not carried out with a system date in the year 2000 or beyond.

```
q cmslevel
CMS Level 11, Service Level 510
Ready; T=0.01/0.01 20:55:15

q cplevel
VM/ESA Version 2 Release 2.0, service level 9705
Generated at 07/02/97 03:02:44 NZS
IPL at 07/03/97 20:36:10 NZS
Ready; T=0.01/0.01 20:55:19
```

Figure 4. VM Migration Case Study - New CP, Old CMS

After the IPL had completed, we logged onto a representative set of interactive and server user IDs to observe CMS IPL and operation and to test application performance and stability.

The selected user IDs run in 370, ESA or XC mode. The scope of testing included the use of standard system facilities (FILELIST, SENDFILE, RECEIVE, XEDIT, COPYFILE), components of CUF (the CMS Utilities Feature - WAKEUP, BROWSE, FLIST, SFPURGER, AUDITOR), and application development tools (PL/I compiler, ISPF).

No errors or discrepancies were encountered.

4.3.1 Compatibility With Earlier Releases of CMS

The case study system has CMS 6 available for application compatibility and migration purposes. To see if it could be done, we IPLed CMS level 6 on CP V2R2. Our ability to test this combination was limited since without service (which we did not have) CMS level 6 does not support minidisks on 3390 devices.

Of the functions tested, most were without error; however, some inconsistencies can be introduced if CP facilities are invoked that do not have matching support from the earlier CMS level. An example of this is the CMS IDENTIFY command as seen in Figure 5 on page 39.

```
q cplevel
VM/ESA Version 2 Release 2.0, service level 9705
Generated at 07/02/97 03:02:44 NZS
IPL at 07/03/97 20:36:10 NZS
Ready; T=0.01/0.01 03:07:23

q cmslevel
VM/SP Release 6, Service Level 604
Ready; T=0.01/0.01 00:06:04

q v 191
DASD 0191 3390 ICMSD9 R/W          5 CYL ON DASD  0711
Ready; T=0.01/0.01 00:06:11

format 191 a
DMSFOR114S Device 191 is an unsupported device type or requested BLKSIZE
is not supported for the device
Ready(00088); T=0.01/0.01 00:06:20

q time
TIME IS 00:06:40 NZS WEDNESDAY 01/05/00
CONNECT= 00:00:48 VIRTCPU= 000:00.27 TOTCPU= 000:00.39
Ready; T=0.01/0.01 00:06:40

id
NMC1     AT IE157    VIA RSCS     01/05/00 00:06:44 NZS      WEDNESDAY
Ready; T=0.01/0.01 00:06:44

set dateformat iso
Ready; T=0.01/0.01 00:06:50

q time
TIME IS 00:06:53 NZS WEDNESDAY 2000-01-05
CONNECT= 00:01:00 VIRTCPU= 000:00.28 TOTCPU= 000:00.40
Ready; T=0.01/0.01 00:06:53

id
NMC1     AT IE157    VIA RSCS     2000-01- 00:06:56 NZS      WEDNESDAY
Ready; T=0.01/0.01 00:06:56
```

Figure 5. VM Migration Case Study - CP and Backlevel CMS

4.4 CMS

The migration path for this study is from CMS 11 to CMS 13. As for CP, the generation of CMS was performed on the user ID dedicated for VM/ESA Version 2 Release 2 software.

The new CMS nucleus was generated without modifications, using the standard facilities. Generation was straightforward and without error.

See the *VM/ESA Service Guide*, SC24-5749, for more information and guidance on (re-)generating CMS for your system.

To simplify testing and recovery procedures, CMS 13 was installed on a separate minidisk (MAINT D190, not the default MAINT 190), and as a separate NSS (Named Saved System). As the majority of user IDs on the case study system use directory profiles for common entries, this allows simple and rapid global changes to the default CMS environment of the user community. Examples are

provided to illustrate this: Figure 6 on page 40, Figure 7 on page 41, and Figure 8 on page 41.

Once the test environment(s) were available, the system was brought to its "normal" state.

The user community was switched from the different CMS levels, and between different "MACHINE" settings during the test cycle in an attempt to identify 370 dependencies that might exist.

Hint

If you want to create a CMS nucleus as a disk file (for example you want to keep it on non-volatile storage, or transfer it to another system), you can do so as follows:

- follow the CMS generation procedures to the point where the IPLable card deck is in your virtual reader
- **CP SPOOL RDR HOLD** - just in case
- **CP ORDER RDR <nucspid>** - put it first in your reader queue
- **READCARD fname ftype fmode** - and read it to disk

To create an IPLable card deck from the disk file, for generating a nucleus for example):

- **CP SPOOL PUNCH *** - spool your virtual punch to yourself
- **PUNCH fname ftype fmode (NOHEADER** - and punch the file

```
q cmslevel
CMS Level 13, Service Level 705
Ready; T=0.01/0.01 03:59:16

det 190
DASD 0190 DETACHED

Ready; T=0.01/0.01 03:59:20

link maint 190 190 rr
Ready; T=0.01/0.01 03:59:28

i cms

VM/ESA REL. 2.2 06/26/97 13:50

                    Welcome to the Development VM Service

Ready; T=0.20/0.23 03:59:33

q cmslevel
CMS Level 11, Service Level 510
Ready; T=0.01/0.01 03:59:38
```

Figure 6. CMS - Swapping CMS Levels Interactively

```
PROFILE CMSOLD
IPL CMS PARM AUTOCR
MACH XC
CONSOLE 0009 3215
SPOOL 000C 2540 READER A
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019E 019E RR
LINK MAINT 019D 019D RR

PROFILE CMSNEW
IPL CMS13 PARM AUTOCR
MACH XC
CONSOLE 0009 3215
SPOOL 000C 2540 READER A
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT D190 0190 RR
LINK MAINT 019E 019E RR
LINK MAINT 019D 019D RR
```

Figure 7. CMS - Sample Directory Entries, Profiles

```
USER TESTNEW XXXXXXXX 32M 128M BCDEFG
   INCLUDE CMSNEW
   ACCOUNT SYSPROG SYSPROG
   OPTION LNKNOPAS
   MDISK 0191 3390 2981 10 VOLID MR XXXXXXXX XXXXXXXX XXXXXXXX

USER TESTOLD XXXXXXXX 32M 128M BCDEFG
   INCLUDE CMSOLD
   ACCOUNT SYSPROG SYSPROG
   OPTION LNKNOPAS
   MDISK 0191 3390 3081 10 VOLID MR XXXXXXXX XXXXXXXX XXXXXXXX
```

Figure 8. CMS - Sample Directory Entries, User IDs

4.5 DirMaint

IBM Directory Maintenance VM/ESA Version 1 Release 5.0 is the version of DirMaint in use on this system; earlier versions of DirMaint do not support VM/ESA Version 2 Release 2 or Year2000.

Prior to VM/ESA Version 2 Release 2 the CMS Pipelines CRC stage did not use the documented algorithm, and returned repeatable but incorrect values. Programs that rely on incorrect values generated by the old CRC stage have compatibility problems with CRC values generated by the current, correct stage. This problem affects DirMaint.

The CRC compatibility problem may be resolved by applying service or by rebuilding the user directory with the correct CRCs.

Figure 9 on page 42 shows console messages from a DirMaint machine that has been IPLed with CMS 13 without applying service or rebuilding the directory.

Note that when a disconnected DirMaint virtual machine detects these errors, it will enter a shutdown/reIPL cycle until it exceeds the “threshold” limit defined in CONFIG DATADVH and logs itself off (see *IBM Directory Maintenance VM/ESA: Tailoring and Administration Guide*, SC23-0533, for more information about configuring DirMaint).

Refer to APARs VM61032, VM58560 and VM59897 for more information about the service options.

```
DVHILZ3510I DVHINITL Parms: BLDCLUSTER BLDLINK
DVHBCK3873I Disk backup processing completed.
DVHREQ2289I Your BACKUP request for DIRMAINT at * has completed; with RC
DVHREQ2289I = 0.
DVHWAI2146I Wakeup caused by console attention on 97/07/05 at 00:25:30.
DVHREQ2290I Request is: SHUTDOWN
DVHREQ2288I Your SHUTDOWN request for DIRMAINT at * has been accepted.
DVHSHU2193I A shutdown command has been issued by
DVHSHU2193I DIRMAINT from IE157.
DVHSHU2197I The DIRMAINT machine is attempting to
DVHSHU2197I re-IPL and restart.
DVHBIN6325E While attempting to initialize
DVHBIN6325E directory entry ESAOP DIRMAINT
DVHBIN6325E has determined that the calculated
DVHBIN6325E CRC does not match the
DVHBIN6325E existing CRC.  The request is rejected.
DVHBIA3206E Source update for 6325 has failed, RC=

DVHBIN3209E Unexpected RC= 3206, from Pipe:
DVHBIN3209E ReadClust
DVHITI3201E The attempt to initialize the internal format directory
DVHITI3201E for DATAMOV1 failed with return code 9003209 from DVHBBINI.
DVHBEG2119T Error in CMS command; RC= 3201
DVHBEG2119T from: EXEC DVHRLDD
DVHBEG2119T at line 40.
DVHSHU2194T Automatic shutdown/restart
DVHSHU2194T initiated.  Machine= DIRMAINT,
DVHSHU2194T caller= DVHBEGIN, reason= 2119
DVHSHU2196I The failing command will be retried.
DVHSHU2197I The DIRMAINT machine is attempting to
DVHSHU2197I re-IPL and restart.
```

Figure 9. DirMaint - CRC Compatibility Issues

4.5.1 Rebuilding the User Directory

Assumptions

The example below assumes that the DirMaint service machine has been installed and configured following the standard installation instructions.

The procedure is:

- log on to the DirMaint service machine
- ensure that the DirMaint service machine is running on the **“old”** level of CMS; if necessary reIPL on that level
- run a **BACKUP** to build the USER BACKUP file (that is, a refreshed monolithic directory source file)
- **SHUTDOWN** the DirMaint service machine

- erase or rename the USER DIRECT E file:

  ERASE USER DIRECT E

  or

  RENAME USER DIRECT E = ODIRECT =

- copy the new backup file as USER INPUT E:

  COPY USER BACKUP G = INPUT E (OLDDATE

- IPL the DirMaint service machine on the **"new"** CMS level
- at the prompt, enter **DVHBEGIN** to start DirMaint. The initialization processing will detect that no USER DIRECT file exists, and will build a new source directory from the new USER INPUT file.

Sample console output following this procedure may be seen in Figure 10 on page 44.

```
DVHWAI2140I Waiting for work on 00/01/04 at 23:34:17.

backup

DVHWAI2146I Wakeup caused by console attention on 00/01/04 at 23:34:27.
DVHREQ2290I Request is: BACKUP
DVHREQ2288I Your BACKUP request for DIRMAINT at * has been accepted.
DVHBCK3871I Disk backup processing started.
           :
           :
DVHBCK3873I Disk backup processing completed.
DVHREQ2289I Your BACKUP request for DIRMAINT at * has completed; with RC
DVHREQ2289I = 0.
DIRMAINT IE157... - 2000/01/04; T=4.42/4.68 23:34:35
DVHWAI2140I Waiting for work on 00/01/04 at 23:34:35.

shutdown

* After DirMaint shutdown has completed ...

li user * * (date

FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
USER     DIRECT   E1 F         80        183          4  1/04/00 23:34:35
USER     BACKOLD  G1 F         80       2493         49  1/04/00 23:19:43
USER     BACKUP   G1 F         80       2493         49  1/04/00 23:34:31
DIRMAINT IE157... - 2000/01/04; T=0.01/0.01 23:34:58

erase user direct e

DIRMAINT IE157... - 2000/01/04; T=0.01/0.01 23:35:17

copy user backup g = input e (oldd repl

DIRMAINT IE157... - 2000/01/04; T=0.01/0.02 23:35:43

* At this point, switch CMS levels and IPL the new CMS.

DVHPRO2002A Manual start is required for DIRMAINT.  Enter "DVHBEGIN"
DVHPRO2002A when ready to start.
DIRMAINT IE157... - 2000/01/04; T=0.52/0.62 23:36:02

dvhbegin
 DVHILZ3510I Starting DVHINITL with directory: USER INPUT E
 DVHILZ3510I DVHINITL Parms: BLDMONO NOCRCWARN
 DVHILZ3510I Starting DVHINITL with directory: USER TEMPDIR G
 DVHILZ3510I DVHINITL Parms: BLDCLUSTER BLDLINK BLDDASD
DIRMAINT IE157... - 2000/01/04; T=6.88/7.21 23:36:29
DVHWAI2140I Waiting for work on 00/01/04 at 23:36:29.

* DirMaint recognises that a new USER DIRECT file is required,
* and builds one, with new CRCs, from the USER INPUT file.
```

Figure 10. DirMaint - Rebuilding the USER DIRECT File

4.6 IOCP

VM/ESA Version 2 has many enhancements for I/O configuration support, particularly for the support of dynamic I/O configuration changes. For more information on this topic see:

- *VM/ESA Planning and Administration*, SC24-5750
- *VM/ESA Planning Dynamic I/O Configuration*, GC24-5695
- *VM/ESA CP Command and Utility Reference*, SC24-5773
- *Input/Output Configuration Users Guide and ESCON Channel to Channel Reference*, GC38-0401

As shown in Figure 11, you can see that the IOCP utility does not use four-digit years when creating output listings. This particular sample was produced with a system date in the year 2000.

```
TIME 20.46 DATE 00.004     PAGE  1

STATEMENTS PROCESSED BY IZP IOCP VERSION 1 RELEASE 5.0, IGNORE=NO, LPAR=YES
             ID      MSG1='Production LPAR IOCP June 1996',              C
                     SYSTEM=(9672,2)

IZP157I SYSTEM=(9672,4) USED BY IZP IOCP

             RESOURCE PART=((IE152,1),(IE157,2))
     *--------------------------------------------------------------------*
     * IE152 = Production partition, IE157 = Dev
     * GenOpts : IOCP LPAR (IOCP <TERM | NOTERM> <WRTAx>
     *--------------------------------------------------------------------*
     * CHANGES : extend RAMAC DASD addresses to full 48 devices SPB 0396
     *           add SS7   SNA 3174s and downstreams            SPB 0496
     *           add CISCO 7000 CIP                             SPB 0596
     *           add IE157 SNA 3174 and downstreams             SPB 0696
     *           add 570   3490 units                           SPB 0896
     *           add new   ESCON CHPs (35-37), move CISCO there SPB 1196
     *           make CISCO CHPs non-shared                     SPB 1196
     *           make IE157 primary for chp 2E (cart units)     SPB 1196
     *
     *--------------------------------------------------------------------*
     *                       CHPID DEFINITIONS
     *--------------------------------------------------------------------*
             CHPID   PATH=(20),TYPE=CNC,PART=(IE152,IE157),SHARED
             CHPID   PATH=(21),TYPE=CNC,PART=(IE152,IE157),SHARED
             CHPID   PATH=(22),TYPE=CTC,PART=(IE152)
             CHPID   PATH=(23),TYPE=CNC,PART=(IE152,IE157),SHARED
             CHPID   PATH=(24),TYPE=CNC,PART=(IE152,IE157),SHARED
             CHPID   PATH=(25),TYPE=CNC,PART=(IE157)
             CHPID   PATH=(26),TYPE=BL,PART=(IE152)
             CHPID   PATH=(27),TYPE=BL,PART=(IE152)
```

Figure 11. IOCP Compile Listing - Date Format

Chapter 5. Application Enabling Program Products

This chapter discusses the date support provided by a selection of application enabling products.

Note: The definitive guide to program product readiness for Year2000 is *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, the most current version of which may be found on the World Wide Web at URL `http://www.software.ibm.com/year2000/resource.html`.

You may also want to refer to Chapter 8, “Language Environment and PL/I” on page 73 as a related topic.

5.1 ISPF

At the time of publication, the current product level is *ISPF/PDF Version 3 Release 2 for VM*.

ISPF has internal variables for storing date and time information, for example ZDATE, ZTIME. This version of the product stores two-digit year information only. The format used for presentation of the ISPF date is dependent on the language setting of the virtual machine displaying it.

See the *ISPF Dialog Management Guide and Reference*, SC34-4273, for more information about the ISPF date and time variables and the ISPF default format for date presentation.

A sample set is provided to:

- demonstrate how to obtain date (and time) information from VM/ESA;
- show how the information may be made available for use in ISPF panels;
- illustrate the difference between date and time information and presentation format as provided by ISPF, and the date and time information and formats available from VM/ESA Version 2 Release 2.

You will find the sample set in Appendix B, “Sample ISPF Routines for Year2000” on page 115.

ISPF is tolerant of dates from the year 2000 onwards; however, use of the ISPF date variables from year 2000 onwards may give rise to unwanted results for comparison operations, and the presentation of ambiguous dates to users.

5.2 SQL/DS and DB2 for VM

At the time of publication, the current product level is *DB2 for VM Version 5 Release 1*.

5.2.1 DB2 for VM Application Programs

DB2 for VM has supported four-digit dates since 1988. DB2 for VM users that have SQL tables defined using the DATE and TIMESTAMP data types will not have any problems when the year 2000 arrives. Other storage options for date values such as character data types or numeric data types may encounter problems. Using a two-digit year will affect the collating sequence of your data. If you have a range of years 1996-2001 the collating sequence with a two-digit year might appear so (00, 01, 96, 97, 98, 99) compared to a four-digit year sequence of (1996, 1997, 1998, 1999, 2000, 2001). The solution to this problem is to migrate the columns and the applications that use them to a data type which supports the Year2000.

Example

```
 EMPLOYEE TABLE

EMPNO     CHAR(6)
FIRSTNME  VARCHAR(12)
LASTNAME  VARCHAR(15)
WORKDEPT  CHAR(3)
HIREDATE  DATE
SALARY    DECIMAL(9,2)
BIRTHDATE DATE
```

When dates are defined as shown in the table example above, they are stored as a string of four bytes. Each byte is two packed decimal digits. The first two bytes are the year, the next byte is the month, and the last byte is the day. The year is already a four-digit value. There will be no problems with tables or applications using this data type.

Let us look at the case when the same table is defined as shown below:

```
 EMPLOYEE TABLE

EMPNO     CHAR(6)
FIRSTNME  VARCHAR(12)
LASTNAME  VARCHAR(15)
WORKDEPT  CHAR(3)
HIREDATE  CHAR(8)
SALARY    DECIMAL(9,2)
BIRTHDATE CHAR(8)
```

If HIREDATE and BIRTHDATE are stored in the format *mm/dd/yy*, there is a problem. The columns HIREDATE and BIRTHDATE should be changed to allow for a four-digit year, and the application program using this table should be adapted to cope with these changes. The best solution in this case, however, is to change the columns HIREDATE and BIRTHDATE to a data type of DATE. In this case the applications would also need to be changed to handle the new data type.

5.2.2 General Use of Dates in DB2 for VM

DB2 for VM uses a fixed window with a size of 42. The year is assumed to be in 20xx if it is less than 43; otherwise, it is assumed to be 19xx. This means that dates before 1943 cannot be given to filtered log recovery or to the Trace Formatter since these dates will be assumed to be in the 21st century. This causes no difficulty because there are no traces or logs containing these years, and they will never be produced.

In DB2 for VM the window rule applies to dates displayed by:

- The filtered log recovery process
- The Trace Formatter
- DBSU output messages
- ISQL printed output
- SQL EXEC terminal output
- DB2 for VM messages
- Operator command responses, including those returned to an ISQL session or to a program via the RDIIN interface, such as RXSQL.

Filtered log recovery

Filtered log recovery accepts control cards specifying dates and times to delimit the range of its actions. These dates and times are converted to TOD values for comparison to the TOD values in the log. Filtered log recovery parses the input date and time control records, but does not syntax check them. This is done by another module that accepts two-digit years and uses the fixed-42 window to convert them to four-digit years, and thus no changes are required.

Trace Formatter

The Trace Formatter accepts control cards specifying dates and times to determine the trace records that are selected for formatting and printing. These date values are compared to the TOD values from the trace records after both are converted to a format of *yymmdd*. The Trace Formatter also uses the fixed window of 42 to do the conversion.

Due to the way the Trace Formatter compares dates with the trace records, it is not possible to select a date range that crosses the century boundary. Trace files that span the century boundary must be formatted with two executions of the Trace Formatter. The same restriction exists for time ranges spanning midnight.

Accounting

DB2 for VM now places century information in the accounting records it generates. This recent change is transparent to existing user-written applications because the century information is stored in a previously reserved field in the accounting record.

Note: The default date format used by SQL/DS and DB2 for VM is an installation tailoring parameter. At installation the default date format for SQL/DS and DB2 for VM databases is ISO format (that is, *yyyy-mm-dd*). This may be changed by tailoring the SQLOPTION.DATE value in the SYSTEM.SYSOPTIONS table. Refer to *DB2 Server for VM System Administration*, GC09-2405, for more information on this topic.

5.3 Query Management Facility - QMF

At the time of publication, the current product level is *Query Management Facility Version 3 Release 3*.

If using earlier versions of the product, you need to be aware that there are some Year2000-related APARs for QMF: PN78157, PN90962 and PN90673.

QMF is used to manage the extraction and manipulation of data from SQL/DS or DB2 for VM databases. It has facilities for sorting and tailoring the format of the extracted data, including *functions* and *edit codes* specifically for operations with the date and time datatypes supported by SQL/DS and DB2 for VM. However, QMF has no intrinsic “date” processing of its own. See:

- *QMF Reference*, SC26-4716, for more information about QMF functions and edit codes for date and time data;
- *DB2 Server for VSE & VM SQL Reference*, SC09-2404, for a description of the date and time datatypes incorporated in SQL/DS and DB2 for VM;
- 5.2, “SQL/DS and DB2 for VM” on page 47 for information about the SQL/DS and DB2 for VM program products.

Data Considerations

An application can store and use data in an SQL/DS or DB2 for VM database in any way. For example, an application may store date or time information as character or numeric data. If these fields are used for queries or date/time calculations, it is an application responsibility to ensure accuracy of the results. An example of this scenario, and how it can result in the incorrect presentation of data, can be see in Figure 12 on page 51. In this example the application stores Year2000-compliant dates for each record and also builds a character sort field consisting of a two alpha string, a two-digit year and five-digit serial number. This field is used to sort the data for presentation to the user. Entries created with a year 2000 date will be sorted to the bottom of the list, which is probably not where they are wanted.

```
F. & C. R.          CORRESPONDENCE LOGGING SYSTEM - LIST LETTERS      MCLS21

 PAGE    1 OF    1     (      5 LETTERS RETRIEVED )

Ref. num   Corresp./Organization  Subject                              Date
---------- --------------------- ------------------------------------ --------
MO9900001  BYRNE                 SG24-2042                            19991212
MO9700002  ROURKE                TEST                                 19970306
MO9700001  ROURKE                TEST BATCH                           19970101
MO9600029  ROURKE                TEST                                 19960101
MO9600024  ROURKE                TEST                                 19960101
MO0000001  BYRNE                 SG24-2042                            20000101









PF1 LETTER      PF2 COMMENTS                      PF5  CLOSING REMINDERS
PF7 PRINT LIST  PF8 DEFINE   PF9 HELP PF10 FORWARD PF11 BACKWARD  PF12 PREVIOUS
```

Figure 12. QMF - Using Application Date Data for Sorting

5.4 DFSORT

At the time of publication, the current product level is *DFSORT/CMS Version 2 Release 1*.

Unlike the versions offered on MVS and VSE, no specific Year2000 support has been added to DFSORT/CMS. As a result, applications that produce data for sorting must ensure that the data is of a format that will result in the correct sort order. A typical example of the kind of problem that may be encountered is in Figure 13 on page 52.

```
type 199705 cal002

20970531WN3IA           0  1
21970527NS1RV     Ő   Ő   ½
24970228NS2RV     Ő   Ő   ½
60960527WN4RV     Ő   Ő   ½
80970629NS1RV     Ő   Ő   ½
09990527CH5RV     Ő   Ő   ½
21970527HN1RV     Ő   Ő   ½
21000101HN1RV     Ő   Ő   ½  2
22970601NS1RV     Ő   Ő   ½

Ready; T=0.01/0.01 05:52:36

dfsort 199705 cal002 a sorted file a cntl(sort fields=(01,13,CH,A))
         SORT FIELDS=(01,13,CH,A)  3

Ready; T=0.02/0.04 05:52:41

type sorted file

09990527CH5RV     Ő   Ő   ½
20970531WN3IA           0
21000101HN1RV     Ő   Ő   ½  4
21970527HN1RV     Ő   Ő   ½
21970527NS1RV     Ő   Ő   ½
22970601NS1RV     Ő   Ő   ½
24970228NS2RV     Ő   Ő   ½
60960527WN4RV     Ő   Ő   ½
80970629NS1RV     Ő   Ő   ½

Ready; T=0.01/0.01 05:52:49


                                                          RUNNING   IE157
```

1 Here we see the raw, unsorted data. The first 13 bytes of each record form a sort key. The key is composed of two bytes for record type, six bytes for date in *yymmdd* format, and five bytes for sub-classes within record type. The remainder of each record contains record counts in packed decimal format.
2 Here we see that a record from 2000-01-01 has been included in the file.
3 Next we issue the DFSORT command to sort the records in ascending order, using the first 13 characters as the sort field.
4 Now we have a problem; the Year2000 record should have been in position five, but instead is in position three.

Figure 13. DFSORT - Inconsistent Sorts with Two-digit Year Data

5.5 OfficeVision/VM

At the time of publication, the current product level is *OfficeVision/VM Release 3*. This release does not provide Year2000 support. There is a PTF available for OfficeVision/VM Release 3 for APAR PN89361 which addresses certain Year2000 issues, and some people may have found the description of PN89361′s support confusing. This PTF simply adds support for VM/ESA V2R2′s FULLDATE and ISODATE date formats to the base OV product. It does **not** bring full Year2000 support to the OV product. It also allows adding entries to the calendar for the year 2000 and beyond.

We did not experiment with release 3 of OfficeVision/VM in the year 2000 and beyond because a pre-beta copy of release 4 was made available for testing during our residency. In the testing we performed, release 4 functioned properly when the system date was beyond 1999. Both release 4, and release 3 with the PTF for APAR PN89361 applied, functioned properly with the user's CP DATEFORMAT set to ISODATE or FULLDATE.

5.5.1 OfficeVision/VM Local Enhancements

Most OfficeVision/VM installations have developed local enhancements to extend its capabilities. The OfficeVision/VM installation studied as part of this residency has many such local enhancements. Three of these enhancements were considered for study and preparation for Year2000 readiness:

- a replacement reminder facility;
- a note bring forward service;
- a public bulletin board service.

These three applications and their required Year2000 updates are treated in detail in Chapter 6, "REXX Based Applications" on page 55.

Chapter 6. REXX Based Applications

REXX is a key component on all VM systems. The uses it is put to vary tremendously, from small execs only a few lines in size used to make tedious command sequences easier, to large programs of thousands of lines performing critical business applications, to a macro language used to extend XEDIT, CMS Pipelines, the Dump Viewing Facility (DVF), et al. COBOL gets all the Year2000 press; but for many VM systems, REXX may be where the Year2000 challenges reside.

In a February, 1996 append on VMSHARE, Alan Ackerman, a member of the SHARE VM Technical Steering Committee, put it this way: “I hate it when people dismiss REXX. I submit to you that REXX is the glue that holds most VM shops together, and that the Year2000 conversion will succeed or fail based on how well we convert our REXX routines.” Nicely said, Alan.

6.1 REXX Date Considerations

Due to the importance of REXX to many installations, it is worthwhile reviewing REXX′s various date formats. Through the date() function REXX has always made available the current date in a varied number of formats. Many of these formats do not include century information and hence may pose problems.

Note: For a complete description of the capabilities of the REXX date() function, consult *VM/ESA REXX/VM Reference*, SC24-5770.

REXX Date formats that do not include century information are:

- ′C′ (Century) - days thus far this century
- ′D′ (Days) - days thus far this year
- ′E′ (European) - current date in *dd/mm/yy* format
- ′J′ (Julian) - year and days thus far this year in *yyddd* format
- ′O′ (Ordered) - current date in *yy/mm/dd* format
- ′U′ (USA) - current date in *mm/dd/yy* format

Date formats that do include century information are:

- ′B′ (Base) - Gregorian rendering of days since 1 Jan 0001
- ′N′ (Normal) - current date in *dd mon yyyy* format
- ′S′ (Standard) - current date in *yyyymmdd* format

There are also date formats that provide a textual representation of the current date:

- ′M′ (Month) - month name of current date
- ′W′ (Weekday) - day name of current date

Additionally, a REXX program could obtain the current date from various external sources; for example:

- Diagnose X′0C′ or X′270′
- CP QUERY TIME
- IDENTIFY

Of these various formats, one format in particular, 'C' (Century) type dates, was often used programmatically when date information or date arithmetic was needed. Such dates were frequently stored in GLOBALVs and the like and then checked against the current date ('C') value to determine if an event was on a new day. The popularity of 'C' (Century) dates over 'B' (Base) dates for this work stems from the fact that 'B' (Base) dates were not available in the first few releases of CMS that included REXX. Even though 'B' (Base) dates have been available for many years now, much code was already written using 'C' (Century) dates; and even new code developed after the availability of 'B' (Base) dates continued to use 'C' (Century) dates out of habit and familiarity.

This is a problem since 'C' (Century) dates will reset and begin counting again from one in the year 2000. 'B' (Base) dates, however, will continue incrementing. Put another way, 1 Jan 2000 is greater than 31 Dec 1999 with 'B' (Base) dates, but it is less with 'C' (Century) dates.

Thus, 'C' (Century) dates are a significant problem area. Another problem area is programs that determine the local date by some means other than the REXX date() function. These programs are susceptible to failure if the user changes the CP DATEFORMAT setting to something other than SHOrtdate.

We'll see below an example of a program that was determining the current date by using IDENTIFY. This program failed when the user's CP DATEFORMAT was changed. An interesting variant of this problem is programs that manipulate timestamps which are returned as part of a command's output, since they too may fail if the CP DATEFORMAT is changed. We'll also review an example of one such program below.

During the residency we ran both case study systems with the system default dateformat changed to ISO. Very little failure was experienced in either environment.

Although REXX does not provide native support in the date() function for the two new CP date formats, ISOdate and FULldate, it is an easy matter to translate from one supported REXX 'S' (Standard) date format, to or from ISOdate and FULldate. The following code illustrates how to convert 'S' (Standard) dates into the two new CP date formats:

```
/**/
  say translate('0123845867',date('S')'-','012345678') /* 'S'=>ISO  */
  say translate('4586780123',date('S')'/','012345678') /* 'S'=>Full */
exit
```

The inverse translation of an ISOdate or FULldate into a REXX 'S' (Standard) date would be done as follows:

```
/**/
  say translate('01235689','1990-10-03','0123456789')  /* ISO =>'S' */
  say translate('67890134','10/03/1990','0123456789')  /* Full=>'S' */
exit
```

Part of the motivation for translating a given date into one of REXX's natively supported date formats is the ability to then use the date() function for date conversion. The date() function has been significantly enhanced in CMS 13 with date conversion. This capability has long been wanted, and it is a welcome update to REXX. Date conversion allows the REXX programmer to convert between the various supported REXX date formats, perhaps converting a date to

'B' (Base) format, doing arithmetic on it, and then converting the date back to its original format. Here is a simple example:

```
/**/
  today = date('B')
  next_week = today + 7
  say "Next week is" date('W',next_week,'B') date('N',next_week,'B')
```

For a more complete, practical example of using this powerful new capability, see the **HOLIDAYS EXEC** in Figure 43 on page 111.

One useful capability of the date() function's conversion ability is to convert all those date formats that do not provide century information to date formats that do. The REXX date() function will use a sliding (-50,+49) window for all dates that do not contain century information except 'C' (Century) dates. 'E' (European), 'J' (Julian), 'O' (Ordered), and 'U' (USA) dates will all have the sliding window applied, whereas 'C' (Century) dates will always take the current century (at the time of conversion) as the base. Should one have 'C' (Century) dates based on 1900 to convert, when the base is 2000, the following REXX snippet may be useful:

```
  new_b_date = date('B','19000101','S') + old_c_date - 1
```

Although it is not entirely straightforward, it is also possible to use the new date() function to determine if a given date is valid. This is useful since it allows one to quickly, easily and correctly validate, for example, a date provided by a user. We provide here a REXX function that can be included in your execs and called for date validation. Again, for an example of actual use please refer to the **HOLIDAYS EXEC** in Figure 43 on page 111.

```
/*
 *  return true if the given date is valid for the given date format
 *  return false if it is not
 */
ValiDate:
  signal on syntax name invalid_date
  call date arg(2),arg(1),arg(2)
  return (1==1)
invalid_date:
  return (0==1)
```

One final point of discussion regarding the REXX date function: When a date() function or time() function is invoked in a REXX clause, it causes a timestamp to be taken that is then used by all time() or date() function calls in that single clause. This is important to remember, especially if the program being developed might be run near midnight. For example:

```
 ...
  today = date()
/* what if midnight occurs here? */
  right_now = time()
 ...
```

It is much better to code this as follows:

```
 ...
  parse value time() date() with right_now today
 ...
```

The latter version guarantees that a common time stamp is used by both the date() and time() function calls. As well, one often sees the date() function called

throughout a program as the program needs the date in one of various formats. For example:

```
...
 today = date('U')                        /* Get USA format date   */
...
 this_month = word(date(),2)              /* Get month name (mmm)  */
...
```

Again the day can change between calls and the exec can have inconsistent results depending on the time of day when it is run. With the new date conversion capability one can take a date stamp at the start of the program and use it as the basis for all the needed date formats throughout, thus assuring consistent results. The previous fragment would thus become:

```
 basedate = date('B')                     /* Save the current date */
...
 today = date('U',basedate,'B')           /* Convert to USA format */
...
 this_month = word(date(,basedate,'B'),2) /* Get month name (mmm)  */
...
```

6.2 REXX OfficeVision/VM Enhancements

Three OfficeVision/VM enhancements were considered for study and preparation for Year2000 readiness.

The first is a replacement reminder facility. The primary additional capability offered by this enhancement is the sending of a note if the reminder recipient is not logged on at the time of the reminder.

The second is a note bring forward facility. Notes may be filed in specially named notelogs and are returned after the requested period of time has elapsed. The notelog names are encoded using the following scheme: BFnnnD to be returned in the requested number of days, BFnnnW to be returned in the requested number of weeks, and BFnnnM to be returned in the requested number of months.

The last application was a public bulletin board service. A designated individual is a bulletin board owner and may post bulletins to the board. Each bulletin has an expiry date, after which it is deleted by the bulletin board service. Each bulletin board has a date of last change to permit the display interface to determine if a new bulletin has been posted since the previous check and to sort the list of all bulletin boards in order of most recent activity.

As you can see, each of these applications has date sensitivities, and in fact some were already causing problems by not supporting dates beyond 1999. In addition, one did not function correctly with CP DATEFORMAT set to anything other than SHORTDATE.

Each of these applications was comprised of several REXX execs, the largest consisting of approximately 15 pieces. Since none of the applications were large, specialized tools were not required to locate date sensitivities. The execs were visually inspected, updated as needed, and then tested. Each of the applications had also made some use of a locally developed REXX date conversion routine, and the characteristic call to that routine made it somewhat easier to locate areas of date manipulations. Where possible the local routine was replaced by

the enhanced REXX date function available in CMS 13. The changes required by these applications are representative of the changes required by most REXX applications that involve dates.

Caution

When updating older code it is natural to notice, and tempting to want to correct, other deficiencies not related to the year 2000 (for example, replacing DESBUF with the usually more appropriate DROPBUF 0; ensuring that all character constants, in particular host command strings, are enclosed in quotation marks, and so on). Such corrections can add greatly to the time spent both in making the modifications and in testing the results.

6.2.1 Reminders

The replacement reminder application provides capabilities similar to those of the reminder function shipped with base OfficeVision/VM, with the additional function of sending a note if the user was not logged on when the reminder message was sent. A sample screen image of the application is not included because the panel presented is identical to the one presented by base OV.

The application works by accepting a future date and time for a reminder and registering that event with a central server (the server used by this application is VM:Schedule from Sterling Software) that then takes appropriate action, sending either a message or a note, when the requested date and time occur. There were several problems with the approaches used by the application.

The biggest surprise was the use of the CMS IDENTIFY command to determine the current date and time, rather than built-in REXX functions. This caused the application to become sensitive to the virtual machine's DATEFORMAT setting. Here's an example of that code:

```
makebuf
'IDENTIFY (STACK'
parse pull USERID . . . . SCHED_DATE . ':' SCHED_MINS ':' ,
           SCHED_SECS .
```

and the same code fragment after revision:

```
userid = userid()
parse value date('S') time() with SCHED_DATE ,
            . ':' SCHED_MINS ':' SCHED_SECS .
```

The second problem was the use of REXX date formats that did not contain century information (in this application these were 'J' (Julian), 'O' (Ordered), and 'U' (USA)). These needed to be changed to other date formats that did contain century information, and the application logic needed minor updates to accommodate this additional information. For all cases the REXX 'S' (Standard) date format was chosen. You can see one illustration of this in the previous example; another is:

```
/*
    Ensure reminder is scheduled for future
*/
        if ((PANEL_REM_YEAR  || '/' ||,
             PANEL_REM_MONTH || '/' ||,
             PANEL_REM_DAY)           <= date(0)) &,
```

which after revisions becomes:

```
/*
    Ensure reminder is scheduled for future
*/
        if ((PANEL_REM_YEAR  || ,
             PANEL_REM_MONTH || ,
             PANEL_REM_DAY)           <= date('S')) &,
```

Another problem with this application was incorrect determination of leap years. It is now well known that the year 2000 is a leap year, and it is also well known that many programs do not properly handle 2000 as a leap year. This was one such program. Once the problem has been identified the fix is easy enough: make an additional check to see if the century is divisible by 400. Here's the incorrect code:

```
if (PANEL_REM_YEAR // 4 = 0) & (PANEL_REM_YEAR // 100 ¬= 0)
     then RETURN 29
     else RETURN 28
```

and here's the corrected leap year determination:

```
if (PANEL_REM_YEAR // 4 = 0) & ,
  ((PANEL_REM_YEAR // 100 ¬= 0) | (PANEL_REM_YEAR // 400 = 0))
     then RETURN 29
     else RETURN 28
```

With these changes the application was ready to schedule reminders in either century or to cross the century boundary.

6.2.2 Bring Forward

The bring forward facility allows a user to file a note from the inbasket into one of several specially named notelogs (see Figure 14 on page 61 for an example). At the requested date in the future (encoded in the notelog name), the note is returned to the inbasket. The note may be brought forward a specified number of days, weeks or months.

```
 VO1                                    Note
Fr  .-------------------------------------------------------------------. :36
To  | MLOOF                       File the Item                         |
cc  |                                                                   |
    | Type the name of the note log to which you want to file the note  |
Fr  | and press ENTER.                                                  |
Su  |                                                                   |
    | Log Name: bf3d____ Note Log                                       |
Ro  |                                                                   |
de  | Denis Meier        travel for the VM staff       Note    05/23/97 |
Mo  |                                                                   |
re  | F1=Help  F4=Prompt  F12=Cancel                                    | p
in  '-------------------------------------------------------------------'

Since I won't be back to the office until next Friday.....I'm sorry to say
you will have to deal with this item.

Denis G. Meier - Mngr. VM Support
DPS, ITSD - Province of B.C.      (voice(250)952-6181
4000 Seymour Place                (fax(250)952-6115
Victoria, B. C. Canada V8X 4S8    (DMEIER@BCSC02.gov.bc.ca
                           E N D  O F  N O T E

Command ===>
F1=Help  F2=File  F3=Exit  F4=Discard  F5=Route  F6=Reply  F7=Backward
F8=Forward  F9=Keep  F10=Resend  F11=Print  F12=Cancel
```

Figure 14. Bring Forward - Filing a Note to be Returned Later

The application works in the following manner. When a user exits from an inbasket, an exit exec is driven which looks for notelogs whose names begin with BF, end with either D, W or M, and have only numeric characters in between (for example, BF16D to return the note in 16 days, or BF1W to return the note in one week). If such a notelog is found, the requested date of return is calculated and the note(s) moved to a notelog whose name is the date of return. An exec is invoked at inbasket open time to look for a bring forward notelog with today's date; if one is found, then the notes in it are moved to the inbasket. As shown in Figure 15 on page 62, the subject and text are altered to indicate that the note originated from the bring forward system rather than the original sender. There is also a utility that allows people to manage notes in the bring forward system.

Again, this application had several problems that required revision in order to make it Year2000 ready. The primary deficiency was that the date calculations were done using REXX 'C' (Century) type dates. These needed to be changed to use REXX 'B' (Base) dates to ensure a continuity in date numbers. Here's an example of the original code:

```
current_century_date = date('C')
select
  when(test_char = 'D') then
       new_date = current_century_date + middle
  when(test_char = 'W') then
       new_date = current_century_date + (middle * 7)
  when(test_char = 'M') then
       new_date = current_century_date + (middle * 30)
  otherwise nop
end

parse_date = caldat('C','U',new_date)  /* local date conversion routine */
```

```
 V01                                    Note
From: DMEIER  --BCSC02                      Date and time    05/23/97 15:31:36

      ***** This note was returned to you by Bring Forward. *****

To: RJEWULA --VAXBCSC  Ron Jewula at VAXB
cc: JADAMS  --BCSC02   Jim Adams          PRUITER --BCSC02   Perry Ruiter

From:  Denis Meier
Subject: *BF* travel for the VM staff

Ron, assuming that Perry's travel to Germany is approved....we will have to
deal with substituting Jim Adams for Perry Ruiter for the VM Workshop in
Montreal.  As you will remember, Perry's name was submitted on this travel
request and in going to Germany Perry will not be able to go to the Workshop
in Montreal.

Since I won't be back to the office until next Friday.....I'm sorry to say
you will have to deal with this item.

Denis G. Meier - Mngr. VM Support
DPS, ITSD - Province of B.C.      (voice(250)952-6181
4000 Seymour Place                (fax(250)952-6115
Victoria, B. C. Canada V8X 4S8    (DMEIER@BCSC02.gov.bc.ca
                           E N D  O F  N O T E

Command ===>
F1=Help  F2=File  F3=Exit  F4=Discard  F5=Route  F6=Reply  F7=Backward
F8=Forward  F9=Keep  F10=Resend  F11=Print  F12=Cancel
```

Figure 15. Bring Forward - The Appearance of the Returned Note

```
mm = substr(parse_date,1,2)
dd = substr(parse_date,4,2)
yy = substr(parse_date,7,2)
new_log_date = 'BF' || mm || dd || yy
```

The revised code looks as follows:

```
current_century_date = date('B')
select
  when (test_char = 'D') then
       new_date = current_century_date + middle
  when (test_char = 'W') then
       new_date = current_century_date + (middle * 7)
  when (test_char = 'M') then
       new_date = current_century_date + (middle * 30)
  otherwise nop
end

parse value date('U',new_date,'B') with mm '/' dd '/' yy .
new_log_date = 'BF' || mm || dd || yy
```

In addition, there were places where REXX 'J' (Julian) dates were being used, primarily to obtain a year for calculations. These were replaced with REXX 'S' (Standard) dates to obtain a four-digit year. For example:

```
c_month = (substr(date('U'),1,2))
c_year = (substr(date('J'),1,2))

if t_month = '12'  & c_month = '01' then do
  c_year = c_year - 1
 end
else do
  if t_month < c_month - 1 then
    c_year = c_year + 1
 end

new_log_date = 'BF' || t_month || test_char_day || c_year
```

was changed to:

```
parse value date('S') with c_year 5 c_month 7 .

if t_month = '12'  & c_month = '01' then do
  c_year = c_year - 1
 end
else do
  if t_month < c_month - 1 then
    c_year = c_year + 1
 end

new_log_date = 'BF' || t_month || test_char_day || right(c_year,2)
```

Another area of review was the naming of the notelog files. The above examples demonstrate that the calculated date was converted to an *mmddyy* value which was used for the notelog name. We decided to leave the format of these names unchanged. There were a number of reasons for this decision:

- The data is relatively short lived, and even with only a two-digit year there is no confusion over to which century the data applies. When century information is required, a windowing technique is sufficient to determine the century.
- Some users understand the format of the logfile names and file their notes with those names to have a note returned on a specific date.

Additionally there is a utility interface that allows people to view or delete notes and to revise the bring forward date of notes in the bring forward system. One available option was to sort by date, and clearly century information is required to properly order items. Although the presentation was left unchanged, code was revised to make use of a date format that contained century information (either 'B' (Base) dates or 'S' (Standard) dates). The presentation had always been a three character month name followed by a two digit day, a slash and a two-digit year (for example, Jul 12/98). This is fortunate and was the reason that the presentation was left unchanged. Starting in January 2001, we will have totally ambiguous abbreviated dates (for example, 02/01/01 could be interpreted as 2002-01-01, 2 Jan 2001, or February 1st 2001); had such a presentation format been in use, something would have to have been done.

The biggest lesson learned from the conversion of this application is that windowing is a valuable technique for a large class of problems. Not every application needs four-digit years, either internally or externally, to be correct and usable.

6.2.3 Bulletin Boards

The final application reviewed was a public bulletin board system. There can be bulletin boards on any topic and their contents are available to all users on the system. An exit at openmail time determines whether new bulletins have been posted since the last time the user checked. If there are revised bulletin boards the user is presented with a list to review. As well, at any time, a user may review all bulletin boards (see Figure 16) by entering the BB command.

```
                    LIST OF AVAILABLE BULLETIN BOARDS                BB10

    BB Name        BB Description        Last Update  Status     BB Manager
--|---------|-------------------------|------------|-------|-----------------
   BBIVORY   IBM Ivory Letters          May 30/97              OVHELP
   BBSRVBUR  RHB-CHSS Service Bureau    May 30/97              Greg Abbott
   BBROADRS  BC HWY Weight Restrictions May 30/97              Ken Harrison
   BBOV      OfficeVision Information   May 30/97              OVHELP
   BB4CAST   Weather Forecast           May 30/97              OVHELP
   BBFIT     Fitness Centre Info        May 29/97              Darlene Therrien
   BBFIRES   Fire Situation Report      May 29/97              OVHELP
   BBTIB     Tech Info Bulletins        May 29/97              OVHELP
   BBPRESS   Press Releases             May 27/97              Paula McGuire
   BBCARE    Cross-Team Ed Working Grp  May 23/97              Sally Robertson
   BBAWARDS  Suggestion Awards          May 23/97              SUGGEST
   BBDB2     DB2 Product Info           May 22/97              DB2HELP
   BBFCNEWS  BC Ferries News Releases   May 22/97              BCFADMIN
   BBIMS     IMS Product Info           May 22/97              John Arduini
   BBLIBRY   BC Gov't IT Library Info   May 20/97              Heather Marsman
   BBCICS    CICS Product Information   May 17/97              John Arduini
   BBINDCON  Industry Conferences       May 17/97              Jim Lackey
   BBMVS     General MVS Related Info   May 17/97              Jim Lackey
   BBSHOWS   Presentations              EMPTY BB               Shirley Nelson
--|---------|-------------------------|------------|-------|-----------------
Position cursor to select Bulletin Board to view, then press ENTER.
>
 PF1  Help          PF3  Quit          PF5  Sort Name     PF6  Cursor Home
 PF7  Backward      PF8  Forward       PF9  Sort Descr'n. PF10 Sort Update
 PF11 Sort Status   PF12 Return
```

Figure 16. Bulletin Board - The Sorted List

In making this application Year2000 ready, no surprises were encountered. Since the changes required by this application mirror the changes required by the previous two, we won't go into great detail on them here. Again REXX date types had to be changed and some code revised to support the new internal date format. Again external presentation of dates was left unchanged since, due to the transient nature of the data, a two-digit year is not ambiguous. The only external change from an end user's perspective is that the date of last access (stored in a GLOBALV variable) was changed from a REXX 'C' (Century) type date to a REXX 'B' (Base) type date.

Work was already underway to provide a Web interface to this application, and many of the Year2000 changes were made in conjunction with the changes needed to more easily present the data to web connected users (for example, the bulletin board data was moved from minidisk to SFS).

6.3 Effects of Alternate Date Formats on REXX Programs

We have already seen how one application was affected by changing the system default date format. A second application was found that did not function properly with the default date format changed to ISODATE. The code in question was using the timestamp provided by the QUERY CPLEVEL command to determine the amount of time that had elapsed since the last IPL of the system. Here is the program before change:

```
/* is it more than 3 hours after IPL? */
too_long_after_ipl:

'PIPE CP Q CPLEVEL',
   '| TAKE LAST 1',
   '| SPEC WORDS 3.2 1',
   '| VAR IPLTIME'
parse var ipltime ipldate iplhh ':' iplmm ':' iplss .
days_after_ipl = date('B') - caldat('U','B',ipldate)
if days_after_ipl > 10000 then      /* cross a century boundary? */
  days_after_ipl = date('B') - caldat('S','B',,
   left(left(date('S'),2)-1||translate('78124536',ipldate,'12345678'),8))
iplsecs = iplss + (60 * iplmm) + (3600 * iplhh)
parse value time() with curhh ':' curmm ':' curss .
cursecs = curss + (60 * curmm) + (3600 * curhh)
select
   when days_after_ipl > 1 then
     return 1
   when days_after_ipl = 1 & cursecs + (86400 - iplsecs) > 10800 then
     return 1
   when days_after_ipl = 0 & cursecs - iplsecs > 10800 then
     return 1
   otherwise
     return 0
end

return
```

There are several interesting things to observe in this subroutine.

- Again we can see the local date conversion routine being used.
- The individual who coded this had the foresight to consider the century change over.
- However, because the expected output from QUERY CPLEVEL includes only a two-digit year, the code is forced to “guess” if a century boundary has occurred. It does this by implementing a fixed 10,000-day window.

The code could have quickly been corrected by adding the SHORTDATE option to the QUERY CPLEVEL command. However, that would still have left us using a window to determine if a century boundary had been crossed. Since CP will now provide a four-digit year on the output of QUERY CPLEVEL, we decided to exploit that by explicitly requesting the ISODATE format. As we've seen above, it's a simple matter to turn an ISO date into a REXX 'S' (Standard) date. At the same time the local REXX date conversion utility was replaced with the new REXX date function. The revised code is as follows:

```
/* is it more than 3 hours after IPL? */
too_long_after_ipl:

parse value date('B') time() with current_date current_time
'PIPE CP Q CPLEVEL ISO',
   '| TAKE LAST 1',
   '| SPEC WORDS 3.2 1',
   '| VAR IPLTIME'
parse var ipltime ipldate iplhh ':' iplmm ':' iplss .
ipldate = translate('01235689',ipldate,'0123456789')
days_after_ipl = current_date - date('B',ipldate,'S')
iplsecs = iplss + (60 * iplmm) + (3600 * iplhh)
parse value current_time with curhh ':' currmm ':' curss .
cursecs = curss + (60 * currmm) + (3600 * curhh)
select
   when days_after_ipl > 1 then
     return 1
   when days_after_ipl = 1 & cursecs + (86400 - iplsecs) > 10800 then
     return 1
   when days_after_ipl = 0 & cursecs - iplsecs > 10800 then
     return 1
   otherwise
     return 0
end

return
```

Chapter 7. Locating Date Usage

In addition to the standalone REXX applications discussed, we also examined a number of DB2/VM (formerly named SQL/DS) based applications written using both PL/I and REXX (using Sterling Software′s DB/REXX). Since we have already, in other chapters, discussed some of the problems we encountered with them, it is more profitable to review some of the techniques used to locate date sensitivities in these applications.

With the REXX OV enhancements, it was mentioned that the applications were small enough that specialized tools were not required. The program segments requiring updates were located through visual inspection. Such was not the case with these larger applications. They often consisted of several dozen pieces, and in a few cases the final executable module contained more than a hundred text decks.

We approached the problem of locating the source of date API usage with the following techniques:

- We made use of an execution time monitoring tool.
- We programmatically scanned the application source code.

7.1 Execution Time Monitoring

Although it would be possible to make use of the CP TRACE command to monitor execution of instructions that return date information (for example, SVC 11 or DIAG X′0C′), such an effort would be tedious, overwhelming and error prone. To simplify execution time monitoring we made use of **VM Timing Facility Monitors** from MiraSoft, Inc. Although this collection of programs offers a number of capabilities, our discussion will focus on monitoring usage of date APIs by programs executing in the CMS environment.

Two complimentary views of the application being monitored are available. First, monitor information can be collected for several users by a central server. As applications are run, use of timing APIs is noted, along with the name and type of the program currently executing. This information is relayed to a central server where the information is collected for all users who have selected such monitoring. Reporting on the collected data provides a succinct overview of which programs are making use of date APIs. Here is an example of such a report after running a DB/REXX application:

```
timesumm programs 19970713
 All intercepts by all programs in file "19970713 TIMESAVE"

 <----Program---->
 Name     Type      Intercepts  User
 -------- --------  ----------  --------
 CMS      Nucleus           27  PRUITER
 NAMEFIND NucExt             1  PRUITER
 PRODCHAN EXEC               4  PRUITER
 PRODMAP  XEDIT            340  PRUITER
 SQLEXEC  NucExt           409  PRUITER
 XEDIT    SubCom             2  PRUITER
```

Second, in the user's machine more detailed reporting is possible on the collected data because additional information is retained that is not forwarded to the central server. For example, here is local reporting on the same data as summarized above:

```
timetrap display
TTRTRA024I 00F190D4: Nucleus  CMS     +0190D4    26 Diagnoses
TTRTRA024I 016F1518: NucExt   SQLEXEC +27B438     1 Diagnose
TTRTRA024I 011D40EA: NucExt   NAMEFIND+0012FA     1 Diagnose
TTRTRA022I In PRODMAP  XEDIT    at line   768   170 times
TTRTRA022I In PRODCHAN EXEC     at line   888     1 time
TTRTRA022I In PRODCHAN EXEC     at line   902     1 time
TTRTRA024I 00F19252: Nucleus  CMS     +019252     1 STCK
TTRTRA024I 010A5DFE: SubCom   XEDIT   +0A5DFE     2 STCKs
TTRTRA024I 01601684: NucExt   SQLEXEC +18B5A4   204 STCKs
TTRTRA024I 016018A0: NucExt   SQLEXEC +18B7C0   203 STCKs
TTRTRA024I 01605EAA: NucExt   SQLEXEC +18FDCA     1 STCK
TTRTRA022I In PRODMAP  XEDIT    at line   768   170 times
TTRTRA022I In PRODCHAN EXEC     at line   888     1 time
TTRTRA022I In PRODCHAN EXEC     at line   902     1 time
```

As you can see, additional useful information is displayed. Notice that if the API is being used by a REXX exec, the line number within the exec is determined. For example, here is line 768 of PRODMAP XEDIT:

```
today = date('o')
```

Line numbers are determined even if the exec in question is compiled.

For MODULEs the module name and an offset within the module are provided. This means that it is especially important that current LOADMAPs be available for the MODULEs being monitored. Here's a small example of running a PL/I based application:

```
timetrap display
  ...
TTRTRA024I 00022A52: Module   LASM01  +0004E2     3 SVCs
TTRTRA024I 0002087E: Module   LASA56  +00087E     3 Diagnoses
TTRTRA024I 0002089A: Module   LASA13  +00C89A     3 Diagnoses
  ...
```

The important thing to remember with execution time monitoring is that the instruction has to be executed to be picked up. This means that to rely solely on execution time monitoring to locate all date API usage in an application, every possible code path would have to be executed. This is a rather daunting undertaking, which is why we recommend using code scanning in conjunction with execution time monitoring.

7.2 ESAMIGR - The Migration Aid

Don't forget that ESAMIGR (*REXX/EXEC Migration Tool for VM/ESA*) is supplied to assist you in identifying potential migration issues between releases of VM.

With suitable additional entries in the ESAMIGR SAMPLIST file, this utility can easily be customized and enhanced to help you in identifying potential Year2000 issues too. See *REXX/EXEC Migration Tool for VM/ESA*, GC24-5752, for full details on how to do this.

Note

ESAMIGR is documented as a tool for helping you migrate REXX, EXEC 2, and component CP ASSEMBLE files (for example HCPRIO) between releases of VM. The utility is not restricted to just these types of files, however; during our study, we successfully scanned REXX, EXEC 2, Assembler, and PL/I source files. All that is necessary to expand the scope of action of ESAMIGR is to include the appropriate entries in the ESAMIGR SAMPLIST file.

As an example, we added the section shown in Figure 17 to the ESAMIGR SAMPLIST file. Since we were migrating from VM/ESA Version 1 Release 2.2 to VM/ESA Version 2 Release 2 we added these entries between the :MIGF. and :EMIGF. tags. (Review the entire ESAMIGR SAMPLIST file for additional information.) A copy of these additions is also provided with the sample programs.

```
-------------------------------------------------------
*****     Local Mods - For Potential Date Issues
-------------------------------------------------------
K 21     YY                                              F   G
K 21     YYYY                                            F   G
K 21     YEAR                                            F   G
K 21     MM                                              F   G
K 21     MTH                                             F   G
K 21     MNTH                                            F   G
K 21     MONTH                                           F   G
K 21     DD                                              F   G
K 21     DAY                                             F   G
K 21     TIME                                            F   G
K 21     DATE                                            F   G
K 21     DATETIME                                        F   G
K 21     STCK                                            F   G
K 21     SCK                                             F   G
-------------------------------------------------------
*****     End of Local Mods
-------------------------------------------------------
```

Figure 17. ESAMIGR - SAMPLIST Entries

Next, we invoked ESAMIGR to produce a report for the REXX exec DSKDTE EXEC. This exec was selected for scanning since it had a known date exposure. As we were looking at the migration from VM/ESA Version 1 Release 2.2 to VM/ESA Version 2 Release 2, the command format was ESAMIGR DSKDTE * (E22 SCANONLY. In Figure 18 on page 70 you can see the summary section of the report.

```
DSKDTE EXEC A
 REC#   INCOMPATIBLE WORDS S TYPE COL   TEXT OF RECORD
 ------ ------------------ - ---- ----- ---------------
  00011 TRACE              2 R    1     Trace 'R'
  00027 YY                 2 F    1     YY = SUBSTR(DATE('E'),1,2)
  00027 DATE               2 F    13    YY = SUBSTR(DATE('E'),1,2)
Scan completed for file DSKDTE EXEC A.

   Summary of hits found during this run.
  Command          Number
  -------          ------
TRACE                  1
YY                     1
SUBSTR                 1
DATE                   1

     Total hits:       4
```

Figure 18. ESAMIGR - Reporting of Date Issues

If you are going to use ESAMIGR in edit and update mode, you should consider constructing the help files to match your extensions to the ESAMIGR SAMPLIST file. A sample is illustrated in Figure 19. Note that the filetype of our help file is "HELPMIGF" since we are migrating from VM/ESA Version 1 Release 2.2 to VM/ESA Version 2 Release 2. The *VM/ESA CMS Users Guide*, SC24-5775, provides details on creating help files.

```
YY

Explanation

 o This string has been highlighted as it is commonly used in programs
   to represent a year, often in two digit format.

User Response

 o If you are not scanning source code for Year2000 exposures, you
   may ignore this hit.

 o If you are scanning for potential date exposures, review the code and

   - attempt to determine if the highlighted item is a date variable;

   - if the item is a date or part of a date, attempt to determine the
     context of use, for example - is this a year ? If it is, does it
     matter ? Will it cause incorrect calculations, or provide confusing
     output for users ?
```

Figure 19. ESAMIGR Sample Help File - YY HELPMIGF

7.3 Code Scanning

CMS file scanning utilities are ubiquitous so we present only a very simple one, SEEK EXEC, in Figure 50 on page 146. We have found that an additional file scanning utility nicely complements the capabilities provided by ESAMIGR described above in 7.2, “ESAMIGR - The Migration Aid” on page 68. Although we concentrated our scanning efforts on PL/I and REXX source code, file scanning can just as well be applied to MODULE and TEXT files. For example, we scanned MODULEs looking for the string IBMBJDT to determine which modules would need to be relinked after APAR PN50632 was applied.

We did not keep statistics, but broadly speaking code scanning resulted in a large number of “false” hits. This most often happened because program comments contained the string being searched for, or the string was not specific enough. Unfortunately each hit must be manually reviewed. There′s no glamor here - just lots of work.

Chapter 8. Language Environment and PL/I

Read Me

Review the “PSP Installation Bucket” for this product and apply anything that you think remotely applicable.

Please bear in mind that these notes result from our early experiences with this product. Many of the issues we discuss may be resolved when more information comes to hand.

This chapter provides information about our experiences with installing *IBM Language Environment for MVS and VM* into a VM environment, as eventual replacement for *Language Environment/370 Common Runtime Libraries*. It is divided into several topics:

- the installation of *IBM Language Environment for MVS and VM* into the VM environment;
- the configuration necessary for the compilation of PL/I source, and the generation and execution of modules;
- the migration of existing PL/I modules to LE;
- some PL/I programming considerations for Year2000.

Note

We were using the OS PL/I Version 2 Release 3 compiler for this exercise, and the following restrictions apply:

- You can compile PL/I programs using PL/I V2R3 and run with LE without restriction.
- PL/I V2R3 can run with LE in compatibility mode only but cannot be used to exploit new macros or source code provided by LE.

8.1 Installing IBM Language Environment for MVS and VM in the VM Environment

Installation was performed using standard VM/SES facilities. You will need to review the program directory and planning information very carefully, as it is quite likely you will need to make one or more key system minidisks larger (for example MAINT 19E).

We made no changes to any of the default settings for LE, including the run time options.

8.2 Generating PL/I Modules with LE

Due to the restrictions for using OS PL/I Version 2 Release 3 with LE, this process was not as painless as we might have expected. We′ll discuss these topics:

- compilation of PL/I source using the LE libraries

- LOAD and GENMOD of modules from TEXT decks produced from the compiles
- execution of the modules

8.2.1 Compiling PL/I Source Using the LE Libraries

Using the PL/I samples provided with LE, compiles produced literally hundreds of error messages and associated large listings. Some of these messages are **I**nformational messages, and result from the naming conventions used for some sub-routines in the LE included libraries. These messages can be controlled by specifying FLAG(W) at compile time. However, some of these messages result from forward incompatibilities between OS PL/I Version 2 Release 3 and the version of the compiler for LE. There are facilities and routines available with LE that are only supported in the IBM PL/I for MVS and VM Version 1 Release 2.

Using the COMPILE option we forced the production of TEXT decks that we later used (successfully) to create modules. We cannot recommend this as an approach, as the integrity of the resulting executable cannot be guaranteed. You will need the following environmental setup for compiling;

- `GLOBAL MACLIB  SCEESAMP SCEEMAC`
- `GLOBAL TXTLIB  SCEELKED SCEEMAC`
- `GLOBAL LOADLIB SCEERUN`

and either the command:

```
PLIOPT fn (MACRO FLAG(W) opt .. opt
```

or in the source program:

```
%PROCESS MACRO FLAG(W) opt ..opt
```

MACRO is a default compile option, and is required to correctly process the LE maclib members %INCLUDEd in the PL/I source.

8.2.2 LOAD and GENMOD Using the LE Libraries

Again, this was not as simple as expected. Initial attempts resulted in errors from one or more undefined names. The missing members were eventually located on the IBM Language Environment for MVS and VM product disks as, or in, lone TEXT decks. We were not able to determine why these objects had not been included in the build process for the product. To get around this issue, we built our own TXTLIB containing the missing TEXT decks, and included the library during the GENMOD processing. The following commands should establish the necessary environment for successful load:

- `SET LDRTBLS 12` (refer to the Installation PSP bucket)
- `GLOBAL TXTLIB  SCEELKED` (we also included our additional TXTLIB on this command)
- `GLOBAL LOADLIB SCEERUN`

Module generation was accomplished as follows:

```
LOAD fn (CLEAR NODUP
GENMOD fn (FROM PLISTART
```

Issuing `GENMOD fn (FROM CEESTART` as documented in the *IBM Language Environment for MVS and VM Programming Guide*, SC26-4818, created a module, but the module would not execute without error because our source program had been compiled with the OS PL/I Version 2 Release 3 compiler.

8.2.3 Execution of PL/I Modules Produced Using the LE Libraries

You must issue GLOBAL LOADLIB SCEERUN prior to executing your module(s) (for further discussion of this topic, see 8.3, "Using LE with Modules From OS PL/I Version 2 Release 3"). At execution time, initialization processing may be slower than that to which you are accustomed.

If you want to override the default LE runtime options, specify the LE options before any program parameters as in the following example:

```
MYPROG TRACE(ON),NATLANG(UEN) / parameters
```

8.3 Using LE with Modules From OS PL/I Version 2 Release 3

We wanted to find out what we had to do to be able to run modules generated using OS PL/I Version 2 Release 3 and Language Environment/370 Common Runtime Libraries.

You should consult *IBM Language Environment for MVS and VM Migration Guide*, SC26-8232, and *IBM Language Environment for MVS and VM Programming Guide*, SC26-4818, to determine the recommended migration path for your applications.

The SCEERUN LOADLIB contains the necessary resources to resolve external references at run time and replaces the PLILIB and IBMLIB LOADLIBs from Language Environment/370 Common Runtime Libraries. We found that provided we had "globaled" the loadlib (GLOBAL LOADLIB SCEERUN), we could execute our test modules without error.

As an experiment, we also tried renaming the SCEERUN LOADLIB to IBMLIB. Module execution was also successful using this setup.

Note that neither of these approaches is the recommended method of porting existing PL/I modules for use with LE.

The documented approach for our modules is to re-link them using the LE libraries:

```
GLOBAL TXTLIB  SCEELKED lib1 lib2 ...          1
GLOBAL LOADLIB SCEERUN
LOAD   progname prog1 prog2 ... ( CLEAR NODUP  2
GENMOD progname ( FROM CEESTART                3
```

1 "lib1 lib2 ..." are the names of any additional libraries that your application requires to resolve external references.

2 "prog1 prog2 ..." are the names of any additional TEXT files to be included in the module generation process.

3 Note that CEESTART replaces PLISTART as used by the Language Environment/370 Common Runtime Libraries and earlier releases of PL/I only if the source programs have been compiled with the IBM PL/I for MVS and VM Version 1 Release 2 compiler. If, as we did, you continue to use the OS PL/I Version 2 Release 3 compiler, then GENMOD fn (FROM PLISTART continues to be the correct way to generate your modules.

Migration Notes

Replacing Language Environment/370 Common Runtime Libraries with IBM Language Environment for MVS and VM requires you to modify your application bootstrap routines to issue the correct GLOBAL commands.

If you are using a fully LE enabled version of the PL/I compiler (e.g. IBM PL/I for MVS and VM Version 1 Release 2), then you must also modify any routines that you have for automating module generation to use CEESTART instead of PLISTART.

There is no LE equivalent for the FLOW/NOFLOW and COUNT/NOCOUNT runtime options.

8.4 Some PL/I Programming Considerations for Year 2000

8.4.1 The DATE Built-in Function

PL/I has various facilities for acquiring system date and time information. The DATE built-in function is very commonly used, particularly in older applications. It returns a six byte character string representing the current date in *yymmdd* format.

The DATETIME built-in function returns a character string of length 17 in the format *yyyymmddhhmmssttt*. This information is better suited for generating date tolerant applications.

We ran into a known problem with the DATE built-in function while testing applications with the system date set to year 2000 or beyond, as shown here. The output from the DATETIME built-in function is shown for comparison. (While running with system dates prior to year 2000, the DATE function returns values as expected.)

```
t
Date     : 001442            i
DATETIME : 20000108220925000
Ready; T=0.03/0.05 22:09:25
```

i Incorrect output from the PL/I Date built-in function.

PTF UN56803 (APAR PN50632) must be applied to correct this, giving the following results:

```
t1
Date     : 000108            ii
DATETIME : 20000108220856000
Ready; T=0.03/0.06 22:08:56
```

ii Output from the Date built-in function after PTF UN56803 has been applied, and the module has been re-linked.

Note: The problem may be circumvented by using the using the IBM Language Environment for MVS and VM runtime libraries. See 8.3, “Using LE with Modules From OS PL/I Version 2 Release 3” on page 75 for more information on this topic.

8.4.2 DATE versus DATETIME

In another part of one of the applications under test, a sub-routine had been developed some years previously for returning the current system year, including the current century. The relevant snippet of the code is shown below:

```
/*********************************************************************/
/*  CHECK THE YEAR OF THE SYSTEM DATE                                */
/*  SINCE THIS PROGRAM WAS WRITTEN IN 1985, YEARS LESS THAN 85 WILL  */
/*  BE IN THE NEXT CENTURY - 2000                                    */
/*  GOOD UNTIL 2084                                                  */
/*********************************************************************/
IF SUBSTR (DATE, 1, 2) < '85'
THEN
  CENTURY = '20';
ELSE
  CENTURY = '19';
SYSTEM_DATE = CENTURY || DATE;
```

This is a working example of a fixed window approach to the year 2000 issue, and will set SYSTEM_DATE correctly (assuming that DATE is working correctly - see 8.4.1, “The DATE Built-in Function” on page 76). However, the need for this can be avoided entirely by use of the DATETIME built-in function, as shown here:

```
SYSTEM_DATE = SUBSTR(DATETIME(),1,8);
```

This will give consistent results without the need for windowing, and requires less overhead to provide the same result.

Chapter 9. Testing VM/ESA in the Year 2000

This chapter deals with actual testing under a VM/ESA Version 2 Release 2 system, for the purpose of evaluating the impact while running from December 31st 1999 into the year 2000 and beyond.

9.1 Preparing for Year2000 IPL

You need to worry about:

- Password Expiry
- Tape Expiry
- Time Zone Setup
- Year2000 Product Support
- Actual Testing
- Backing Out
  - Residual Data
  - Date stamps

Once you have devised your plan, you can start your testing in the “future.”

9.2 Password Expiry

Most VM systems run with some Directory Maintenance product, such as IBM′s DirMaint, or with an External Security Manager (ESM), such as RACF. The first thing that is going to happen after bringing up VM/ESA with a date in the future is that your password will be expired, and you will have to change it. Since you will probably want to back out at a certain point, it will be advantageous to minimize the impact of expired passwords. You may find the following procedure useful:

- Identify a working user ID
- Give it **LOGONBY** authority to all the other user IDs you intend to use
- IPL with the latest date you intend to test
- Change its password under that date

This will allow you to work through this user ID, for all your Year2000 testing, without having to worry about expired passwords. Once you are done, you can change the password of your working user ID, if required.

9.3 Tape Expiry

Many VM systems run with a tape management product, for example Sterling Software′s VM:Tape. It is quite likely that some or all tapes will have “expired,” that is, passed their scratch due date. If you think you will want to back out at any stage, you must take steps to protect your data on tape. This can be achieved by suppressing “scratch” processing for the duration of the testing, or by defining different tapes or tape pools specifically for test use.

9.4 Time Zone Setup

If you do not code **TimeZone_Boundary** statements in advance in your SYSTEM CONFIG, the default time zone will be **UTC**, which is probably not what you want for your test environment. You should add TimeZone_Boundary statements to include all the date ranges with which you intend to test. Figure 20 shows an extract of the statements used in our testing.

```
/*------- Timezone Definitions and Boundary Conditions -------*/

TimeZone_Definition edt West 04.00.00
TimeZone_Definition est West 05.00.00

TimeZone_Boundary  on 1997-04-06  at 02:00:00  to edt
TimeZone_Boundary  on 1997-10-26  at 02:00:00  to est

TimeZone_Boundary  on 1998-04-05  at 02:00:00  to edt
TimeZone_Boundary  on 1998-10-25  at 02:00:00  to est

TimeZone_Boundary  on 1999-04-04  at 02:00:00  to edt
TimeZone_Boundary  on 1999-10-31  at 02:00:00  to est

TimeZone_Boundary  on 2000-04-02  at 02:00:00  to edt
TimeZone_Boundary  on 2000-10-29  at 02:00:00  to est

TimeZone_Boundary  on 2001-04-01  at 02:00:00  to edt
TimeZone_Boundary  on 2001-10-28  at 02:00:00  to est


        --- Several Entries Not Shown ---


TimeZone_Boundary  on 2039-04-03  at 02:00:00  to edt
TimeZone_Boundary  on 2039-10-30  at 02:00:00  to est

TimeZone_Boundary  on 2040-04-01  at 02:00:00  to edt
TimeZone_Boundary  on 2040-10-28  at 02:00:00  to est

TimeZone_Boundary  on 2041-04-07  at 02:00:00  to edt
TimeZone_Boundary  on 2041-10-27  at 02:00:00  to est
```

Figure 20. TimeZone_Boundary SYSTEM CONFIG Statements for Year2000 Testing

The complete copy of these statements is provided with the sample programs.

9.5 Year2000 Product Support

IBM maintains a list of products which are Year2000 ready. This list, along with the *Year 2000 Guide*, is accessible on the World Wide Web through the Year2000 home page at URL http://www.ibm.com/year2000/

There is also a Preventive Service Planning (PSP) “bucket” for Year2000 support. The *upgrade* id is **YR2000VM**, and the *subset* ids list the various VM products for which Year2000 service is available.

9.6 Actual Testing

This is the easy part. Certain things you want to be on the lookout for:

- What happens after midnight on December 31st, 1999?
- Sharing data with an older VM system
- 2000 is a Leap year: will your software recognize February 29th, 2000?
- Which date format(s) is(are) least confusing?
- UTC versus Local Time
- How far can I test?

9.6.1 December 31st, 1999

Our first test was to watch VM/ESA “roll over” after midnight on the last day of 1999. Figure 21 shows the actual IPL.

```
 03:44:04 VM/ENTERPRISE SYSTEMS ARCHITECTURE  V2 R2.0  SERVICE LEVEL 9705;
 03:44:08 SYSTEM NUCLEUS CREATED ON 05/06/97 AT 16:25:50, LOADED FROM 220RES
03:44:10
03:44:13 ************************************************************
03:44:14 * LICENSED MATERIALS - PROPERTY OF IBM*                      *
03:44:18 *                                                           *
03:44:21 * 5654-030 (C) COPYRIGHT IBM CORP. 1983, 1996. ALL RIGHTS    *
03:44:25 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,     *
03:44:28 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE   *
03:44:30 * CONTRACT WITH IBM CORP.                                    *
03:44:33 *                                                           *
03:44:36 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.           *
03:44:39 ************************************************************
03:44:42
 03:44:45 HCPZC06718I Using parm disk 1 on volume 220RES (device 65E6).
 03:44:45 HCPZC06718I Parm disk resides on cylinders 2367 through 2396.
03:44:45 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
03:44:45       (NOAUTOlog)) or (SHUTDOWN)
03:45:03
03:45:03 NOW 03:45:03 EDT THURSDAY 06/19/97
03:45:03 Change TOD clock (Yes|No)
03:45:18 ==> yes
03:45:18 Set date MM/DD/YY
03:45:38 ==> 12/31/99
03:45:38 Set time HH:MM:SS
03:45:56 ==> 23:30:00
03:45:56 Press "TOD ENABLE SET" key at designated instant.
23:30:00 NOW 23:30:00 EST FRIDAY 12/31/99
23:30:00 Change TOD clock (Yes|No)


                                                        CP Read   VMESA220
```

Figure 21. IPL on the Last Day of the Century

Then, just before midnight, as shown in Figure 22 on page 82, we watched the **HCPMID6001I** message, advising of the date change.

```
==> q t iso
23:59:59 TIME IS 23:59:59 EST FRIDAY 1999-12-31
23:59:59 CONNECT= 00:07:21 VIRTCPU= 000:00.52 TOTCPU= 000:04.50
Ready; T=0.01/0.01 23:59:59
00:00:00
00:00:00
00:00:00
00:00:00 HCPMID6001I  TIME IS 00:00:00 EST SATURDAY 01/01/00
00:00:00

==> q t iso
TIME IS 00:00:20 EST SATURDAY 2000-01-01
CONNECT= 00:32:49 VIRTCPU= 000:00.83 TOTCPU= 000:01.24
Ready; T=0.01/0.01 00:00:20

==> q t short
TIME IS 00:00:29 EST SATURDAY 01/01/00
CONNECT= 00:32:57 VIRTCPU= 000:00.83 TOTCPU= 000:01.25
Ready; T=0.01/0.01 00:00:29

==> q t full
TIME IS 00:00:36 EST SATURDAY 01/01/2000
CONNECT= 00:33:04 VIRTCPU= 000:00.83 TOTCPU= 000:01.25
Ready; T=0.01/0.01 00:00:36

                                                          CP Read   VMESA220
```

Figure 22. Rollover after Midnight on December 31st 1999

As you can see from the output of the **q t short** command, a date stamp of *01/01/00* is not immediately intuitive. The **FULdate** or **ISOdate** option is required here, with the latter being best suited for sorting.

9.6.2 Logging On

As mentioned at the beginning of this chapter, the working user ID's password was now expired. This test system had RACF as its External Security Manager (ESM).

Figure 23 on page 83 shows the logon, and the result of the RACF **LISTUSER (LU)** command.

```
L TIBO

Enter your password,
or
To change your password, enter: ccc/nnn/nnn
     where ccc = current password, and nnn = new password

RPIMGR042I PASSWORD EXPIRED

To change your password - enter: nnn/nnn  where nnn = new password
or,
enter LOGOFF to cancel

 ICH70001I TIBO     LAST ACCESS AT 07:51:06 ON WEDNESDAY, JUNE 18, 1997
 HCPRPW004I Password changed
There is no logmsg data
FILES: 0004 RDR,   NO PRT,   NO PUN
LOGON AT 00:01:30 EST SATURDAY 01/01/00
CMS V2.2.0 SystemPac/VM RSU9705

Ready; T=0.09/0.12 00:01:32
rac lu 1
USER=TIBO  NAME=UNKNOWN  OWNER=MAINT     CREATED=97.050
 DEFAULT-GROUP=SYS1      PASSDATE=00.001  PASS-INTERVAL=186
 ATTRIBUTES=SPECIAL OPERATIONS
 REVOKE DATE=NONE   RESUME DATE=NONE
 LAST-ACCESS=00.001/00:01:29
 CLASS AUTHORIZATIONS=NONE
 NO-INSTALLATION-DATA
 NO-MODEL-NAME
 LOGON ALLOWED   (DAYS)          (TIME)
 ---------------------------------------------
 ANYDAY                          ANYTIME
  GROUP=SYS1      AUTH=USE      CONNECT-OWNER=MAINT     CONNECT-DATE=97.050
    CONNECTS=   181  UACC=NONE     LAST-CONNECT=00.001/00:01:29
    CONNECT ATTRIBUTES=NONE
    REVOKE DATE=NONE   RESUME DATE=NONE
  GROUP=SYSTEM    AUTH=USE      CONNECT-OWNER=MAINT     CONNECT-DATE=97.050
    CONNECTS=    00  UACC=NONE     LAST-CONNECT=UNKNOWN
    CONNECT ATTRIBUTES=NONE
    REVOKE DATE=NONE   RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
 NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED
RPITMP002I ENTER RACF COMMAND OR "END" TO EXIT
Ready; T=0.09/0.12 00:01:35
                                                          Running   VMESA220
```

1 This is the RACF LISTUSER command, abbreviated.

Figure 23. Expired Password

As you can see, the ESM accepts the year 2000 date, but it only displays two-digit years.

9.6.3 VM Accounting

Now that we are in the year 2000, how will the account records report usage? Well, after issuing the **CP ACNT ALL** command, and linking to the accounting user ID, **FILELIST (ISO** showed the information in Figure 24.

```
 TIBO     FILELIST A0  V 169  Trunc=169 Size=90 Line=1 Col=1 Alt=0
Cmd Filename Filetype Fm Format Lrecl    Records     Blocks   Date       Time
    ACCOUNT  $A010100 C1 F         80          7          1 2000-01-01  0:32:24
    ACCOUNT  $A123199 C1 F         80         64          2 1999-12-31 23:40:06
    CMS      EXEC     C1 F         90         87          2 1999-12-31 23:35:22
    ACCOUNT  $A061897 C1 F         80         24          1 1997-06-18 17:00:18
    ACCOUNT  $A061797 C1 F         80         27          1 1997-06-17 17:26:06
    ACCOUNT  $A061697 C1 F         80         22          1 1997-06-16 16:08:24
    ACCOUNT  $A061397 C1 F         80         22          1 1997-06-13 10:46:42
    ACCOUNT  $A061297 C1 F         80          7          1 1997-06-12 10:55:02
    ACCOUNT  $A061197 C1 F         80          2          1 1997-06-11 16:51:22
    ACCOUNT  $A061097 C1 F         80         18          1 1997-06-10 11:36:24
    ACCOUNT  $A060697 C1 F         80         22          1 1997-06-06 15:23:37
    ACCOUNT  $A060597 C1 F         80          6          1 1997-06-05 17:20:38
    ACCOUNT  $A060497 C1 F         80         16          1 1997-06-04 17:18:55
    ACCOUNT  $A060297 C1 F         80         10          1 1997-06-02 15:35:37
    ACCOUNT  $A053097 C1 F         80         21          1 1997-05-30 17:20:23
    ACCOUNT  $A052997 C1 F         80          5          1 1997-05-29 16:30:32
    ACCOUNT  $A052897 C1 F         80         10          1 1997-05-28 17:01:18
    ACCOUNT  $A052797 C1 F         80         13          1 1997-05-27 15:32:47
    ACCOUNT  $A052697 C1 F         80         45          1 1997-05-26 17:17:05
    ACCOUNT  $A052397 C1 F         80         44          1 1997-05-23 14:50:36
    ACCOUNT  $A052297 C1 F         80          5          1 1997-05-22 15:51:51
    ACCOUNT  $A052197 C1 F         80         26          1 1997-05-21 17:06:56
    ACCOUNT  $A052097 C1 F         80          8          1 1997-05-20 17:59:36
    ACCOUNT  $A051997 C1 F         80          4          1 1997-05-19 13:08:10
    ACCOUNT  $A051697 C1 F         80         19          1 1997-05-16 16:45:01
1= Help       2= Refresh  3= Quit   4= Sort(type)  5= Sort(date)  6= Sort(size)
7= Backward   8= Forward  9= FL /n 10=            11= XEDIT/LIST 12= Cursor

====>
                                                          X E D I T  1 File
```

Figure 24. Checking the Contents of the Accounting User ID

The first two accounting files in the list were created as a result of our 1999 IPL and our ACNT ALL command. Running the **ACCOUNT** MODULE from the CMS Utilities Feature (CUF) produced no surprises (see Figure 25 on page 85). As most CUF programs, ACCOUNT is Year2000 ready. Although it will not display four-digit years, it will work in 2000 and beyond.

```
VM/ESA IPOE SYSTEM USAGE OVER THE PERIOD 01/01/00 TO 01/01/00      ALL SHIFTS      USER  ALL         ACCT  ALL

USERID   SESS    CONNECT RATIO   REAL-CPU   VIRT-CPU   PG READ  PG WRITE     SIO   PUNCH    PRINT     READ CYL/BLK TDSK TAPE DISK

DISKACNT    1 000001:05 03925 0000:00:01 0000:00:00       529      1066     279       0        0        0       0    0    0    0
EREP        1 000001:05 ***** 0000:00:00 0000:00:00        15      1054     118       0        0        0       0    0    0    0
OPERATOR    1 000001:05 00981 0000:00:04 0000:00:00      2649      1981     247       0      213        0       0    0    0    0
OPERSYMP    1 000001:05 ***** 0000:00:00 0000:00:00        12      1075     129       0        0        0       0    0    0    0
RACFVM      1 000001:05 00782 0000:00:05 0000:00:04      1227      1508     887       0       56        0       0    0    0    0
SYSTEM      1 000001:06 00668 0000:00:06 0000:00:00      1865      2715       0       0        0        0       0    0    0    0
TIBO        1 000001:01 01221 0000:00:03 0000:00:02        30        15     653       0       22       22       0    0    0    1
VMSERVR     1 000001:04 03896 0000:00:01 0000:00:01       929      1366     296       0        0        0       0    0    0    0
VMSERVS     1 000001:04 03895 0000:00:01 0000:00:01         0      1720     324       0        0        0       0    0    0    0

 TOTALS     9 000009:44 01670 0000:00:21 0000:00:08      7256     12500    2933       0      291       22       0    0    0    1
```

Figure 25. Report from CUF's ACCOUNT MODULE

9.6.4 SFPURGER

SFPURGER functions correctly in the year 2000 and beyond. Figure 26 on page 86 is an extract of a log file from an SFPURGER run with a system date of 1st January 2000.

SFPURGER stores output files with a filetype of *aaayyddd*, where *aaa* is a descriptive mnemonic and *yyddd* is the Julian date when the run took place; for example:

```
listfile spfpurger (date iso
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE       TIME
SFPURGER LOG00001 A1 F       1024          1          1 2000-01-01 21:19:15
SFPURGER RUN00001 A1 F       1024         22          6 2000-01-01 21:19:15
```

For Year2000 support, SFPURGER will assume the following for RUN, LOG and TST files:

- if “yy” is more than 70, then file is from year “**19**yy”
- if “yy” is 70 or less, then file is from year “**20**yy”

See APAR VM60540 for more information.

```
**                              **                                  **
**                      SFPURGER Console Log                        **
**         Created by VM/ESA 5654-030     1 Jan 2000   05:07:05     **
**                      SFPURGER at IE157                           **
**                              **                                  **

DMSCYS2469I SFPURGER OPTIONS file processed ...
DMSCYS2452I SFPURGER starting at 05:06:03 on 1 Jan 2000.
DMSCYS2453I Running in RUN mode - RUN00001.
DMSCYS2470I Using SFPURGER MODULE with SFPURGER CONTROL file.
DMSCYS2456I Erasing old output files till -1358. ??????

DMSCYS2496I Control card scan complete.

DMSCYS2459I Examining output file ...
DMSCYS2462I Spool file scanning begins ...
DMSCYS2482I Executing: CP PURGE userid   PRT 0010
```

Figure 26. SFPURGER Log File Extract

9.6.5 Some Other CUF Utilities

In general, CUF utilities accept Year2000 dates; they just won't display more than two digits of the year. For instance, **BROWSE** still displays the date in the format *mm/dd/yy*.

As another example, **FLIST** also displays the date as BROWSE, but when sorting by date will recognize Year2000 dates and put them before the 19xx dates. Figure 27 shows sorted output from the LISTFILE command, followed by an invocation of FLIST, after also sorting by date.

```
pipe cms listfile (date iso | sort 57-80 desc | take 10 | cons 1
TEST     FILE     A1 F         80          1          1 2099-06-23  4:00:00
DMSEXIFI OUTPUT   A1 V         19          6          1 2011-11-12 10:00:28
LASTING  GLOBALV  A1 V         90         32          1 2002-07-08  2:55:18
ISPPROF  ISPPROF  A1 V       1335          1          1 2002-07-07  8:30:35
ISPSPROF ISPPROF  A1 V       1719          1          1 2002-07-07  8:30:35
TABLES   MACLIB   A1 F         80         93          2 2000-01-01  0:58:57
ACCOUNT  CARDS    A1 F         80         85          2 2000-01-01  0:45:20
WAKEUP   TIMES    A1 F         80          2          1 2000-01-01  0:00:40
API      LOAD     A1 F        111          2          1 1999-12-31 23:43:06
PROFILE  EXEC     A1 V         44         22          1 1997-06-20  3:18:16


flist 2
 LVL 0 - A 191      1800 BLKS 3390 R/W     302 FILES 53%  FILE      1 OF    302
TEST     FILE     A1                       F    80      1      1  6/23/99  4:00
DMSEXIFI OUTPUT   A1                       V    19      6      1 11/12/11 10:00
LASTING  GLOBALV  A1                       V    90     32      1  7/08/02  2:55
ISPPROF  ISPPROF  A1                       V  1335      1      1  7/07/02  8:30
ISPSPROF ISPPROF  A1                       V  1719      1      1  7/07/02  8:30
TABLES   MACLIB   A1                       F    80     93      2  1/01/00  0:58
ACCOUNT  CARDS    A1                       F    80     85      2  1/01/00  0:45
WAKEUP   TIMES    A1                       F    80      2      1  1/01/00  0:00
API      LOAD     A1                       F   111      2      1 12/31/99 23:43
PROFILE  EXEC     A1                       V    44     22      1  6/20/97  3:18

--- Other entries not shown ---


 PF: 1 HLP 2 BRW 3 END 4 XED 5 SPL 6 /SB 7 SCB 8 SCF 9 /SD 10 /ST 11 >I  12 CAN
```

1 Use **pipe** here to get only the first 10 entries in most recent date order.
2 The screen is displayed here, after sorting by date (**/sd**).

Figure 27. FLIST in the Year 2000 and Date Sorting

Note that, despite the cryptic two-digit year date stamp, FLIST has properly sorted the files in the same order as in our previous LISTFILE. IBM CUF development is aware of the need for FLIST to display a four-digit year, so there should be a new FLIST in the near future.

The CUF utility that is probably most important here, as far as the date is concerned, is **WAKEUP**. As the other CUF utilities, it only displays two-digit years. However, it will properly recognise Year2000 dates, as shown in Figure 28 on page 88.

```
q t iso
TIME IS 00:00:17 EST SATURDAY 2000-01-01
CONNECT= 00:01:45 VIRTCPU= 000:00.42 TOTCPU= 000:00.57
Ready;

type wakeup times

ALL      00:00:01 12/31/97 MSG01
ALL      00:00:01 01/01/96 MSG02

Ready;
wakeup (file
DMSCYW2246I* 00001 ALL 00:00:01 01/01/00 MSG01
Ready(00003);

wakeup (file
DMSCYW2246I* 00002 ALL 00:00:01 01/01/00 MSG02
Ready(00003);

type wakeup times

ALL      00:00:01 01/01/00 MSG01
ALL      00:00:01 01/01/00 MSG02

Ready;


                                                    Running   VMESA220
```

Figure 28. WAKEUP 'Waking Up' on a Year2000 Date

WAKEUP evaluated the date as being later than '12/31/97' and '01/01/96', so it executed the instructions and updated the date stamp, even though, as with the other CUF utilities, it is still a two-digit year. The return code of *00003* is also the expected WAKEUP return code, when WAKEUP reaches a timer event from the TIMES file.

9.6.6 Sharing Data

On this particular test system, there is an **ISFC** (Inter System Facility for Communications) link to a VM/ESA V1R2.2 system, which gives us access to an SFS filepool from that system. Since VM/ESA V1R2.2 does not have Year2000 support, it cannot correctly represent a Year2000 date for a file created by a CMS 13 user on the VM/ESA V2R2 system.

In the sample output below, we've created file *TEST FILE* on January 1st 2000. We have accessed our VM/ESA V1R2.2 filepool as filemode 'B'. We then copy the file from 'A' to 'B' with the *OLDDATE* option.

```
copy test file a = = b(olddate
Ready; T=0.01/0.02 00:16:37

l test file *(short
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 F          80          2          1  1/01/00  0:08:40
TEST     FILE     B1 F          80          2          1  1/01/00  0:08:40
Ready; T=0.01/0.01 00:16:46

l test file *(full
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 F          80          2          1  1/01/2000  0:08:40
TEST     FILE     B1 F          80          2          1  1/01/1900  0:08:40
Ready; T=0.01/0.01 00:16:55
```

Note: In this case, the culprit is not the level of **CMS** running in the VM/ESA V1R2.2 SFS server, but the level of the **CMSFILES** DCSS on that system.

Although the LISTFILE output with the default SHORTDATE option is not revealing, the difference becomes quite clear with the FULLDATE option.

Caution

If you intend to switch between CMS levels while testing (see 4.4, "CMS" on page 39 for a discussion of this topic), you need to be aware of how this can affect file dates. For example, files created or updated using CMS 11 will carry only two-digit year information. Thus, if you create a file using CMS 13 with a date of 2000-01-01, then switch to CMS 11 and copy the file, you will change the date information from 2000 to 1900. Figure 29 on page 90 illustrates this point.

```
* We start on CMS 13 on 1 January 2000; the file already exists.

q cmslevel
CMS Level 13, Service Level 705
Ready; T=0.01/0.01 03:49:58
li test file * (d
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 F         80          1          1 2000-01-01  3:52:03
Ready; T=0.01/0.01 03:52:06

* Now we switch to CMS 11 ...

#cp det 190 #link * 190 190 rr #ipl cms
DASD 0190 DETACHED
VM/ESA REL. 2.2 06/26/97 13:50

                    Welcome to the Development VM Service

Ready; T=0.20/0.24 03:56:38
q cmslevel
CMS Level 11, Service Level 510:
READY; T=0.01/0.01 03:56:40

* and inspect the file date.

li test file a (d
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 F         80          1          1  1/01/00  3:52:03
Ready; T=0.01/0.01 03:56:44

* Copy the file with the OLDDATE option ...

copy test file a = = = (repl oldd
Ready; T=0.01/0.01 03:56:56

* and inspect the file date again; it appears unchanged.

li test file a (d
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 F         80          1          1  1/01/00  3:52:03
Ready; T=0.01/0.01 03:56:58

* Now switch back to CMS 13.

#cp det 190 #link * d190 190 rr #ipl cms13
DASD 0190 DETACHED
DASD 0190 LINKED R/O; R/W BY MAINT   ; R/O BY    27 USERS
CMS13
Ready; T=0.10/0.12 03:57:11

* Look at the file date now....

li test file a (d
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 F         80          1          1 1900-01-01  3:52:03
Ready; T=0.01/0.01 03:57:15
```

Figure 29. CMS Levels - Changing File Update Dates

9.6.7 Does February 29th, 2000 Exist?

Not yet, but it will. If all goes well, it should be on a Tuesday. 2000 is a **leap year**. Figure 30 shows the midnight rollover from our test system, which had been IPLed with a date of February 28th 2000. Other software products may not recognize 2000 as a leap year, so individual product testing may be required.

```
==> q t iso
23:59:55 TIME IS 23:59:55 EST MONDAY 2000-02-28
23:59:55 CONNECT= 00:07:21 VIRTCPU= 000:00.52 TOTCPU= 000:04.50
Ready; T=0.01/0.01 23:59:55
00:00:00
00:00:00
00:00:00
00:00:00 HCPMID6001I  TIME IS 00:00:00 EST TUESDAY 02/29/00
00:00:00
==> q t iso
00:00:10 TIME IS 00:00:10 EST TUESDAY 2000-02-29
00:00:10 CONNECT= 00:07:36 VIRTCPU= 000:00.52 TOTCPU= 000:04.51
Ready; T=0.01/0.01 00:00:10


                                                    Running   VMESA220
```

Figure 30. February 29th, 2000

9.6.8 UTC versus Local Time

Listing date stamps on files can lead to confusion, depending on the options used. In Figure 31 on page 92, we use the **OPENVM LISTFILE** to check the contents on the *root* directory, in the Byte File System (BFS). Note the difference in the highlighted date stamps.

```
==> openvm list
Directory = '/'
Update-Dt  Update-Tm Type  Links          Bytes Path name component
12/31/1999 23:47:24   F        1           1014 '.sh_history'
12/31/1999 23:46:58   D        -              - 'bin'
02/10/1997 11:48:19   D        -              - 'dev'
02/10/1997 16:12:38   D        -              - 'etc'
02/10/1997 11:48:09   D        -              - 'home'
02/10/1997 11:48:17   L        -              - 'lib'
02/10/1997 11:48:10   D        -              - 'opt'
02/10/1997 11:48:15   E3       -              - 'tmp'
02/10/1997 11:48:17   L        -              - 'u'
02/10/1997 11:48:14   D        -              - 'usr'
02/10/1997 11:48:15   E3       -              - 'var'
Ready; T=0.08/0.11 23:47:31
==> openvm list (utc
Directory = '/'
Update-Dt  Update-Tm Type  Links          Bytes Path name component
01/01/2000 04:47:24   F        1           1014 '.sh_history'
01/01/2000 04:46:58   D        -              - 'bin'
02/10/1997 16:48:19   D        -              - 'dev'
02/10/1997 21:12:38   D        -              - 'etc'
02/10/1997 16:48:09   D        -              - 'home'
02/10/1997 16:48:17   L        -              - 'lib'
02/10/1997 16:48:10   D        -              - 'opt'
02/10/1997 16:48:15   E3       -              - 'tmp'
02/10/1997 16:48:17   L        -              - 'u'
02/10/1997 16:48:14   D        -              - 'usr'
02/10/1997 16:48:15   E3       -              - 'var'
Ready; T=0.07/0.10 23:47:43

==> l temp file c(alldates
FILENAME FILETYPE FM CREATE-DT  CREATE-TM  LREF-DT  UPDATE-DT  UPDATE-TM
TEMP     FILE     C1 1999-12-31 23:48:59 2000-01-01 1999-12-31 23:48:59
Ready; T=0.01/0.01 23:49:42

                                                          Running   VMESA220
```

Figure 31. UTC versus Local Time

The list option used can change the year in the date stamp. Also note that in the case of the **LISTFILE** command, with the **ALLDATES** option, the dates and times are all *local*, except for the *last reference date (LREF-DT)*, which is expressed in **UTC**, as per the *Usage Notes* for HELP LISTFILE.

9.6.9 Ultimate Year2000 Test Date

VM/ESA V2R2 Year2000 support is currently based on the *Time Of Day (TOD)* clock. Since the system date must be representable by the TOD clock, which will reach its maximum value on September 19, 2042 at 19:53:47 (TOD value 'FFFFFFFF FFFFFFFF'), Year2000 support assumes a 20xx date whenever the range is 00 to 41, and a 19xx date whenever the range is 42 to 99. Figure 32 on page 93 shows what happens at midnight on December 31st, 2041.

```
 23:59:23 VM/ENTERPRISE SYSTEMS ARCHITECTURE  V2 R2.0  SERVICE LEVEL 9705;
 23:59:23 SYSTEM NUCLEUS CREATED ON 05/06/97 AT 16:25:50, LOADED FROM 220RES

--- IBM Copyright Notice not shown ---

 23:59:24 HCPZCO6718I Using parm disk 1 on volume 220RES (device 65E6).
 23:59:24 HCPZCO6718I Parm disk resides on cylinders 2367 through 2396.
23:59:44 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
23:59:44       (NOAUTOlog)) or (SHUTDOWN)
23:59:54 W DR NOAUTO
23:59:54 NOW 23:59:54 UTC FRIDAY 12/31/43
23:59:54 Change TOD clock (Yes|No)
24:00:00 Y
24:00:00 Set date MM/DD/YY
24:00:13 12/31/41
24:00:13 Set time HH:MM:SS
24:00:20 23:58:00
24:00:20 Press "TOD ENABLE SET" key at designated instant.
23:58:00 NOW 23:58:00 UTC THURSDAY 12/31/41
23:58:00 Change TOD clock (Yes|No)
23:58:04 NO
23:58:04 The directory on volume 220RES at address 65E6 has been brought online.
 23:58:13 HCPWRS2513I
 23:58:13 HCPWRS2513I Spool files available      420
 23:58:22 HCPWRS2512I Spooling initialization is complete.
23:58:24 DASD 65E7 dump unit CP IPL pages 3534
23:58:25 There is no logmsg data
23:58:25 FILES:   NO RDR, 0047 PRT,   NO PUN
23:58:25 LOGON AT 23:58:25 UTC THURSDAY 12/31/41
23:58:25 GRAF  0020 LOGON  AS  OPERATOR USERS = 1
23:58:25 HCPIOP952I 0032M system storage
23:58:25 FILES: 0000229 RDR, 0000113 PRT,      NO PUN

#cp q t iso
23:58:41 TIME IS 23:58:41 UTC THURSDAY 2041-12-31
23:58:41 CONNECT= 00:00:16 VIRTCPU= 000:00.23 TOTCPU= 000:03.75

00:00:00
00:00:00
00:00:00
00:00:00 HCPMID6001I  TIME IS 00:00:00 UTC FRIDAY 01/01/42
00:00:00

q t iso
00:00:07 TIME IS 00:00:07 UTC FRIDAY 1942-01-01
00:00:07 CONNECT= 00:01:42 VIRTCPU= 000:00.38 TOTCPU= 000:03.96
                                                        Running   VMESA220
```

Figure 32. VM/ESA after Year 2041

As you can see, it is quite easy to go back 100 years without really trying!

9.6.10 Other Considerations

VM development is continuing to consider further extensions to Year2000 support. For example, these could eventually include the date format on the LOGON, RECONNECT, DISCONNECT, and LOGOFF messages. These of course are of much lesser impact than some of the items previously discussed. Nevertheless, simple elements such as the **banner** page that delimits the output of a system-attached printer can become confusing, as seen in Figure 33 on page 94, which came from our sample accounting report on Figure 25 on page 85.

```
********************************************************************************************************************************************
********************************************************************************************************************************************
********************************************************************************************************************************************
**  VM/ESA   V2 R2.0 SERVICE LEVEL 9705 ** VMESA220 ** 01/01/00 ** 01:10:49 ** PRINTER 002E ** CPU SERIAL  00514  CPU MOD
L  9672 **


TIBO      TIBO      FILE NAME/TYPE=                          ORIGINID= TIBO               VV         VV MM         MM
TIBO      TIBO                                                                            VV         VV MMM       MMM
TIBO      TIBO      CREATION DATE/TIME= 01/01/00  00:45:53     SYSID=    VMESA220        VV         VV MM M     M MM
TIBO      TIBO                                                                            VV         VV MM  M   M  MM
TIBO      TIBO      COPY=   001   RECS= 0022      00000      00000     2222222          VV       VV  MM   M M   MM
TIBO      TIBO                                   0000000    0000000   222222222          VV     VV   MM    M    MM
TIBO      TIBO      CLASS=  W     SPID= 0062    00      00  00      00  22     22           VV   VV    MM         MM
TIBO      TIBO                                  00      00  00      00        22      EEEEEEEEEEEEEV VSSSSSSSSSS      MAAAAA
TIBO      TIBO      FORM=   STANDARD            00      00  00      00       22       EE           VVSS     MM  SS     AA   AA
TIBO      TIBO                                  00      00  00      00      22        EE            VSS     MM        AAM    AA
TIBO      TIBO      DEST=   OFF                 00      00  00      00     22         EE             SS             AA       AA
TIBO      TIBO                                  00      00  00      00    22          EEEEEEEEEE     SSSSSSSSSS   AAAAAAAAAAAA
TIBO      TIBO      CHAR=                        0000000    0000000    22222222      EE                        SS  AA         AA
TIBO      TIBO                                    00000      00000    222222222     EE                        SS  AA         AA
TIBO      TIBO                                                                      EE             SS         SS  AA         AA
TIBO      TIBO      TITLE:                                                          EEEEEEEEEEEE   SSSSSSSSSS   AA         AA


TAG DATA:


TTTTTTTT    IIIIII    BBBBBBBB    0000000
TTTTTTTT    IIIIII    BBBBBBBBB  000000000
   TT         II      BB      BB  00      00
   TT         II      BB      BB  00      00
   TT         II      BBBBBBBB    00      00
   TT         II      BBBBBBBB    00      00
   TT         II      BB      BB  00      00
   TT         II      BB      BB  00      00
   TT       IIIIII    BBBBBBBBB  000000000
   TT       IIIIII    BBBBBBBB    0000000

--- Several Lines Not Shown ---

********************************************************************************************************************************************
********************************************************************************************************************************************
********************************************************************************************************************************************
********************************************************************************************************************************************
```

Figure 33. Printer Output Banner Page in the Year 2000

9.7 Backing Out

It is not uncommon for processes or products to use the datestamp in the name of CMS files created, as may be seen in the file names in our accounting test back in 9.6.3, “VM Accounting” on page 84. PROP, the Programmable Operator facility, stores its daily log files with names in the format *LGyymmdd nodeid*, which means that, in our example, where we IPLed just before year 2000, PROP would have created the two following files, for December 31st 1999 and January 1st 2000:

```
LG991231 VMESA220
LG000101 VMESA220
```

Aside from the names being confusing before these dates are valid, the very existence of these files can lead to further confusion when December 31st 1999 really occurs, since PROP would merely continue appending to these logs.

If you have tested WAKEUP with a TIMES file, you will need to reset the two-digit year since WAKEUP is not currently running on a fixed or sliding window, and will assume that a year of ′00′ is less than a year of ′97′.

In some cases, programs that maintain files using the datestamp may fail altogether, not expecting the file to exist before the actual date. You should decide:

- What do I do with these files?
  - Rename?
  - Backup?
  - Erase?
- What do I do with the other files created during the test?
  - Erase?
  - Redate? (see Figure 40 on page 105 for an example)
- What do I do with the spool files created during the test?
  - Receive?
  - Purge?

Note

Any previous data backups of these files will be obsolete since they may not reflect the correct century. You should create new data backups.

Appendix A. Sample REXX Routines for Year2000

Note

In this Appendix are some examples, that may help understand the enhancements to some of the REXX and CSL routines related to Year2000.

These samples are from various sources, and are provided as is; tailor to your specific needs and environment.

The complete set of samples will be available on the **CD-ROM** version of this Redbook in the newest update of the *System/390 Redbooks Collection (SBOF-7201)*. How to download these samples is described in Appendix E, "Installing the Samples from CD-ROM" on page 149.

REXX/CSL Samples

- *Y2000 EXEC*

  Using diagnose X'00' to check for Year2000 support.

- *Y2K EXEC*

  Using YEAR2000 CMSFLAG to check for Year2000 support.

- *DMSERP EXEC*

  Using DMSERP (CSL Extract/Replace) to check the new *FILE_DATE_CENTURY* flag.

- *DMSEXIFI EXEC*

  Using DMSEXIFI (CSL File Exist) to override the default *SHORTDATE* on the resulting output file date.

- *DMSQEFL EXEC*

  Using DMSQEFL (CSL Query Function Level) to check CP and CMS levels.

- *DMSPLUEX EXEC*

  Using DMSPLU MODULE (Change CMS File Date) to see the new four-digit year support.

- *VMTMDTS EXEC*

  Using VMTMDTS (New CSL DateTimeSubtract) to get a date from a different timezone and in a different output format than the input.

- *VMTMDTS2 EXEC*

  Using VMTMDTS again, this time to actually subtract a value from an input date. Different formats used for input, subtraction and output.

- *HOLIDAYS EXEC*

  Using CMS 13's REXX date conversion support to do date arithmetic and calculate on which days statutory holidays will be taken.

A.1 Sample Diagnose X′00′

This REXX EXEC checks the bitmap for the VM/ESA version, and tells the user whether Year2000 support is there or not. Since diagnose X′00′ returns a 40-byte string for each level of VM you may be running under, it reports this information for each level.

```
type y2000 exec

/*+-Y2000 EXEC-------------------------------------------------------+
  | Sample use of Diagnose X'00' to check for Year2000 support.      |
  |                                                                  |
  | Bit 13 of the version indicator will be on for Year2000 support. |
  |                                                                  |
  +------------------------------------------------------SG24-2042-+*/

Result=Diag("00")
Levels=Length(Result)%40
Say "You are running VM on level" Levels"."

Do i=1 To Levels
   Indic = Substr(Result,((Levels-i)*40)+25,2)
   Bit=Substr(x2b(c2x(Indic)),14,1)
   If Bit Then Say "Year2000 support is ON  for your level" i "VM."
          Else Say "Year2000 support is OFF for your level" i "VM."
End
Ready; T=0.01/0.03 08:08:55



y2000
You are running VM on level 2.
Year2000 support is OFF for your level 1 VM.
Year2000 support is ON  for your level 2 VM.
Ready; T=0.01/0.02 08:09:01
                                                        Running   VMESA220
```

Support

Figure 34. Y2000 EXEC - Using Diagnose X′00′ to Check for Year2000

As you can see in this example, we are running VM/ESA second level, with Year2000 support. The first level host is pre-V2R2.

A.2 YEAR2000 CMSFLAG

This very simple REXX EXEC returns the value of the new *YEAR2000* CMSFLAG. The routine will return a '1' (true) if Year2000 support is present, or a '0' (false) if not.

```
type y2k exec

/*+-Y2K EXEC-------------------------------------------------------+
  | Use new VM/ESA V2R2.0 YEAR2000 CMSFLAG.                        |
  +-------------------------------------------------------SG24-2042-+*/
Y2K?: Signal On Syntax Name NoY2K
Return CMSFLAG("YEAR2000")
NoY2K: Return 0
Ready; T=0.01/0.02 08:09:09

y2k

Ready(00001); T=0.01/0.02 08:09:12

                                                          Running   VMESA220
```

Figure 35. Y2K EXEC - Using YEAR2000 CMSFLAG

A.3 Sample CSL Extract/Replace

Figure 36 on page 100 first shows a call to DMSERP (CSL Extract/Replace) to get the new *FILE_DATE_CENTURY* value from an existing file. Then it extracts the new *YEAR2000_SUPPORT* indicator and displays it.

```
/*+-DMSERP EXEC-----------------------------------------------------+
  | Sample CSL call to DMSERP (Extract/Replace).                    |
  |                                                                 |
  | First call is to get the new FILE_DATE_CENTURY indicator, which |
  | will be '0' if file is between 1900-1999 and '1' if 2000-2099.  |
  |                                                                 |
  | Second call is to get the value of the new YEAR2000_SUPPORT     |
  | information name.  This will be '1' only if both CP and CMS have |
  | Year2000 support.  Otherwise, a '0' is returned.                |
  |                                                                 |
  | For details of DMSERP syntax and field definitions, refer to    |
  | "CMS Appl. Dev. Ref.", SC24-5762, or for online info, enter:    |
  |                 HELP ROUTine DMSERP and HELP ROUTine INFONAME   |
   +-----------------------------------------------------SG24-2042-+*/

 rtn      = 'DMSERP'
 ret      = ''
 reset    = 'RESET'
 extract  = 'EXTRACT'                  /* Indicate which DMSERP       */
 numargs  = 2
 info     = 'FILE_DATE_CENTURY'        /* Infoname requested          */
 buff     = ' '
 datatype = ''
 buflen   = Length(CSLBUFF)
 flags    = '00000000'
 searchty = 'AND '                     /* Search type (2 arguments)   */
 token    = 0
 name1    = 'FILE_NAME'                /* Arg 1 infoname              */
 value1   = 'PROFILE '
 valtype1 = 32                         /* 32 = character string       */
 vallen1  = 8
 valcomp1 = 'EQ'                       /* Comparison value            */
 name2    = 'FILE_TYPE'                /* Arg 2 infoname              */
 value2   = 'EXEC    '
 valtype2 = 32
 vallen2  = 8
 valcomp2 = 'EQ'

 Call CSL 'RTN RET RESET'              /* First, RESET DMSERP         */

 Call CSL 'RTN RET EXTRACT',           /* Call DMSERP EXTRACT         */
          'NUMARGS INFO BUFF DATATYPE BUFLEN FLAGS SEARCHTY TOKEN',
          'NAME1 VALUE1 VALTYPE1 VALLEN1 VALCOMP1',
          'NAME2 VALUE2 VALTYPE2 VALLEN2 VALCOMP2'

 Say "Return Code" Ret "from" rtn extract info
 Say "File" value1 value2 "Century indicator:" Strip(buff)
 Say
```

Figure 36 (Part 1 of 2). DMSERP EXEC - CSL Calls to 'Extract/Replace'

```
/* Now we will setup DMSERP to check the YEAR2000_SUPPORT indicator  */
/* Just need to reset a few values before the call                   */

info     = 'YEAR2000_SUPPORT'         /* New infoname requested      */
numargs  = 0                          /* No arguments                */

Call CSL 'RTN RET RESET'              /* RESET DMSERP                */

Call CSL 'RTN RET EXTRACT',           /* Call DMSERP EXTRACT         */
         'NUMARGS INFO BUFF DATATYPE BUFLEN'

Say "Return Code" Ret "from" rtn extract info
Say "Year2000 Support indicator:" Strip(buff)
```

```
dmserp
Return Code 0 from DMSERP EXTRACT FILE_DATE_CENTURY
File PROFILE  EXEC     Century indicator: 0

Return Code 0 from DMSERP EXTRACT YEAR2000_SUPPORT
Year2000 Support indicator: 1
Ready; T=0.01/0.02 08:09:45
                                                    Running   VMESA220
```

Figure 36 (Part 2 of 2). DMSERP EXEC - CSL Calls to 'Extract/Replace'

In this example, we are checking the *FILE_DATE_CENTURY* flag on our PROFILE EXEC. A value of '0' means it is pre-2000. The second call to get the *YEAR2000_SUPPORT* indicator returns a '1' when both CP and CMS have Year2000 support.

A.4 Sample CSL DMSEXIFI (File Exist)

Figure 37 on page 102 shows DMSEXIFI (CSL File Exist) which now accepts the *DATEFORMAT* parameter with a default value of *SHORTDATE*. The EXEC instead uses the DATEFORMAT value from the user's environment.

```
type dmsexifi exec

/*+-DMSEXIFI EXEC-----------------------------------------------------+
  | Sample CSL call to DMSEXIFI (Check if file exists and stats).    |
  | This one will run on both V2R2 CMS (CMS13) and prior.  In the    |
  | CMS13 case, the CSL call will use the DATEFORMAT parameter as    |
  | set for the userid running the EXEC, as opposed to using the     |
  | DMSEXIFI default of SHORTDATE.                                   |
  |                                                                  |
  | For details of DMSEXIFI syntax and field definitions, refer to   |
  | "CMS Appl. Dev. Ref.", SC24-5762, or for online info, enter:     |
  |                         HELP ROUTine DMSEXIFI                    |
  +-------------------------------------------------------SG24-2042-+*/

fn_ft="DMSEXIFI EXEC"                /* Identify the file to check   */
filemode="*"
file=fn_ft filemode                  /* Load info into FILE variable */
length1=Length(file)
nocommit="NOCOMMIT"
"pipe cp q dateformat |",            /* Get user's DATEFormat        */
         "specs w4 1 |",
         "var dtype"
If rc=0 Then nocommit=nocommit dtype  /* Concatenate if it exists     */
length2=Length(nocommit)

Call CSL "DMSEXIFI RETCODE REASCODE", /* Actual CSL call, up to the   */
         "FILE LENGTH1 NOCOMMIT",     /* date and time info, as per   */
         "LENGTH2 RD_AUTH WR_AUTH",   /* 'Group 6' in SC24-5762       */
         "EXTERN_PROTECT STATUS",
         "FILEID FM_NO RECFORM",
         "REC_LEN NUM_BLKS NUM_RECS",
         "DATE TIME"

Say "CSL Call to DMSEXIFI to see if" file "exists ..."
Say "EXTRACT retcode :" retcode
Say "        reascode:" reascode
Say "        rd_auth :" rd_auth
Say "        wr_auth :" wr_auth
Say " extern_protect :" extern_protect
Say "         status :" status
Say "         fileid :" fileid
Say "          fm_no :" fm_no
Say "        recform :" recform
Say "        rec_len :" rec_len
Say "       num_blks :" num_blks
Say "       num_recs :" num_recs
Say "           date :" date          /* Date of last update          */
Say "           time :" time          /* Time of last update          */
Ready; T=0.01/0.02 08:10:12
                                                        Running   VMESA220
```

Figure 37. DMSEXIFI EXEC - CSL Call 'Exist File' using Caller's DATEFormat

And now, here is a sample invocation of the DMSEXIFI EXEC:

```
q dateformat
User Dateformat = ISODATE
Ready; T=0.01/0.02 08:10:23

dmsexifi
CSL Call to DMSEXIFI to see if DMSEXIFI EXEC * exists ...
EXTRACT retcode : 0
        reascode: 0
        rd_auth : 1
        wr_auth : 1
 extern_protect : 0
         status : 7
         fileid : DMSEXIFIEXEC
          fm_no : 1
        recform : V
        rec_len : 72
       num_blks : 1
       num_recs : 44
           date : 1997-06-20
           time : 11:42:36
Ready; T=0.01/0.02 08:10:36



                                                          Running   VMESA220
```

Figure 38. Sample Output from DMSEXIFI EXEC

Since this user has his DATEFormat set to *ISODATE*, the EXEC added this parameter to the CSL call. This resulted in the *date* value to be expressed as ′yyyy-mm-dd′.

A.5 Sample DMSQEFL Usage (Query Function Level)

The DMSQEFL CSL routine in Figure 39 on page 104 is not a new or enhanced Year2000 routine. It simply extracts the current CP and CMS function levels. These are represented as whole numbers. CP level 28 introduced Year2000 support, while CMS got it at level 32.

```
type dmsqefl exec

/*+-DMSQEFL EXEC-------------------------------------------------+
  | Use CSL Routine DMSQEFL (Query Functional Level) to get current |
  | levels of CP and CMS.                                          |
  +-----------------------------------------------------SG24-2042-+*/

CPY2K  = 28                        /* Level of CP  supporting Year2000 */
CMSY2K = 32                        /* Level of CMS supporting Year2000 */

Call CSL('DMSQEFL RCODE RS CPPROD CPLVL CPSRV CMSLVL CMSSRV CMSUSR')

If RCode <> 0 Then
    Do
      Say 'Error on CSL Call to DMSQEFL RC=' RCODE 'RS=' RS
      Exit RCODE
    End

If (CPLVL >= CPY2K) & (CMSLVL >= CMSY2K) Then
    say 'This System has Year2000 support.'
Else
    say 'This System does not have Year2000 support.'
Exit RCode
Ready;

dmsqefl
This System has Year2000 support.
Ready;


                                                      Running   VMESA220
```

Figure 39. DMSQEFL EXEC -Query Function Level CSL Routine

A.6 Sample DMSPLU Usage (Change File Date Utility)

Figure 40 on page 105 uses DMSPLU which now accepts a four-digit year format. For compatibility, it still accepts the two-digit year.

```
type dmspluex exec

/*+-DMSPLUEX EXEC-----------------------------------------------------+
  | Sample usage of DMSPLU MODULE (Change CMS file datestamp).        |
  |                                                                   |
  | First we create file TEST FILE.  Then we explicitely change its   |
  | datestamp to the year 2050.  Then we use the still-supported      |
  | 2-digit year to set it to the year 50.  Since DMSPLU is using a   |
  | sliding window technique (-50 to +49), the result is 1950.        |
  |                                                                   |
  | (Note: DMSPLU remains an unsupported utility).                    |
  +-------------------------------------------------------SG24-2042-+*/

 PLU = "DMSPLU TEST FILE A"            /* Setup DMSPLU command string */
 LIST= "LISTFILE TEST FILE A (DATE"    /* Setup LIST command string   */

 "pipe literal TEST FILE |> TEST FILE A"  /* Let's create this file   */

 LIST                                  /* Initial creation timestamp  */

 PLU "12/12/2050 23:59:59"             /* Zap date to end of 2050     */

 LIST                                  /* Check resulting timestamp   */

 PLU "12/12/50 23:59:59"               /* Zap date to end of '50      */

 LIST                                  /* Should show 1950 now        */
Ready; T=0.01/0.02 08:12:45

set dateformat short
Ready; T=0.01/0.02 08:12:48
dmspluex
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 V         10          1          1  6/25/97  3:03:36
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 V         10          1          1 12/12/50 23:59:59
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
TEST     FILE     A1 V         10          1          1 12/12/50 23:59:59
Ready;
set dateformat full
Ready; T=0.01/0.02 08:12:50
dmspluex
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 V         10          1          1  6/25/1997  3:03:51
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 V         10          1          1 12/12/2050 23:59:59
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS     DATE     TIME
TEST     FILE     A1 V         10          1          1 12/12/1950 23:59:59
Ready; T=0.01/0.02 08:12:52

                                                          Running   VMESA220
```

Figure 40. DMSPLUEX EXEC - DMSPLU Change File Date Utility

In the first invocation, with the **DATEFormat** set to *SHOrtdate*, although the date gets set to the year 2050, the LISTFILE only shows '12/12/50'. Of course, you can use the *FULldate*, or the *ISOdate* option on the LISTFILE command. Note that with the 'sliding window', year '50' is interpreted as '1950'.

A.7 Sample CSL VMTMDTS (DateTimeSubtract)

Figure 41 shows a call to the CSL *DateTimeSubtract* routine. This call is simply to get the result of an input date in a different format, and in a different timezone. No subtraction is performed.

```
/*+-VMTMDTS EXEC-----------------------------------------------------+
  | Sample CSL call to VMTMDTS (DateTimeSubtract).                   |
  |                                                                  |
  | We start off with December 31st 1995 at 2AM EST time (WEST5GMT). |
  | Timestamp is in USA format (MM/DD/YYYY).  We only want to get    |
  | the PST time (WEST8GMT), without subtracting anything, so we put |
  | a '0' in the subtracting value, expressed in MET format (Mission |
  | Elepased Time: ddddddd/hh:mm:ss).  We want the                   |
  | result in ISO format (YYYY-MM-DD).                               |
  |                                                                  |
  | For details of VMTMDTS syntax and field definitions, refer to    |
  | "CMS Appl. Dev. Ref.", SC24-5762, or for online info, enter:     |
  |                         HELP ROUTine DATETIMS                    |
  +-----------------------------------------------------SG24-2042-+*/

Call APILOAD "VMREXTMR"              /* APILOAD to get variables...  */
                                     /*... loaded from VMREXTMR COPY */

/* 'minuend' values: our starting point                              */

minuend_stamp = "12/31/1995 02:00:00" /* Initial Date                 */
minuend_stamp_length = Length(minuend_stamp)
minuend_stamp_format = vm_tmr_format_usa /* Initial Date Format is USA*/
minuend_stamp_bias = -18000          /* Timezone offset in seconds   */
minuend_stamp_window_type = 0
minuend_stamp_window_position = 0

/* 'subtrahend' values: our calculation (here: no subtraction)       */

subtrahend_stamp = "0/00:00:00"      /* Set subtract time to 0       */
subtrahend_stamp_length = Length(subtrahend_stamp)
subtrahend_stamp_format = vm_tmr_format_met /* Expressed in MET form. */
subtrahend_stamp_window_type = 0
subtrahend_stamp_window_position = 0
subtrahend_stamp_bias = 0

/* 'difference' values: how we want the result displayed             */

difference_stamp_buffer_size = 20    /* Set result buffer size       */
difference_stamp_buffer = Copies(" ",difference_stamp_buffer_size)
difference_stamp_format = vm_tmr_format_iso /* Want result ISO format */
difference_stamp_window_type = 0
difference_stamp_window_position = 0
difference_stamp_bias = -28800       /* Result timezone offset sec.  */
```

Figure 41 (Part 1 of 2). VMTMDTS EXEC - CSL DateTimeSubtract Routine

```
/* CSL Call to DateTimeSubtract (check RTNMAP VMTMDTS)                */

Call CSL "DateTimeSubtract",
         "retcode reascode",
         "minuend_stamp",
         "minuend_stamp_length",
         "minuend_stamp_format",
         "minuend_stamp_bias",
         "minuend_stamp_window_type",
         "minuend_stamp_window_position",
         "subtrahend_stamp",
         "subtrahend_stamp_length",
         "subtrahend_stamp_format",
         "subtrahend_stamp_bias",
         "subtrahend_stamp_window_type",
         "subtrahend_stamp_window_position",
         "difference_stamp_buffer",
         "difference_stamp_buffer_size",
         "difference_stamp_length",
         "difference_stamp_format",
         "difference_stamp_bias",
         "difference_stamp_window_type",
         "difference_stamp_window_position"

/* Display selected variables with the Result */

Say "retcode="retcode "reascode="reascode
Say "minuend_stamp          :" minuend_stamp
Say "minuend_stamp_bias     :" minuend_stamp_bias
Say "subtrahend_stamp       :" subtrahend_stamp
Say "difference_stamp_bias  :" difference_stamp_bias
Say "difference_stamp_buffer:" difference_stamp_buffer
```

```
vmtmdts
retcode=0 reascode=0
minuend_stamp          : 12/31/1995 02:00:00
minuend_stamp_bias     : -18000
subtrahend_stamp       : 0/00:00:00
difference_stamp_bias  : -28800
difference_stamp_buffer: 1995-12-30 23:00:00.
Ready; T=0.01/0.02 08:13:05

                                                    Running   VMESA220
```

Figure 41 (Part 2 of 2). VMTMDTS EXEC - CSL DateTimeSubtract Routine

This call used December 31st, 1995 at 2AM. The datestamp is expressed in **USA** format, and the timezone offset of -18000 seconds, works out to five hours west of GMT. The stamp is converted to **ISO** format, and the timezone offset of -28800 seconds gives us eight hours west of GMT.

A.8 Sample CSL VMTMDTS2 (DateTimeSubtract)

Figure 42 shows a call to the CSL *DateTimeSubtract* routine again. This time, we will actually perform a subtraction on the input date, and express the result in a different format than the input.

```
/*+-VMTMDTS2 EXEC-----------------------------------------------------+
  | Sample CSL call to VMTMDTS (DateTimeSubtract).                    |
  |                                                                   |
  | We start off with January 1st, 2000 at 10PM GMT (UTC).            |
  | This could be from, say, London England.  The timestamp is in     |
  | rexx_date_n format (dd Mmm yyyy).  We want to subtract 2 days,    |
  | and get the result, in full ISO format (YYYY-MM-DD) for Sydney    |
  | Australia (EAST9GMT).                                             |
  | For simplicity, we use the MET (Mission Elapsed Time) format for  |
  | the 2-day subtraction (dddddd/hh:mm:ss).                          |
  |                                                                   |
  | For details of VMTMDTS syntax and field definitions, refer to     |
  | "CMS Appl. Dev. Ref.", SC24-5762, or for online info, enter:      |
  |                         HELP ROUTine DATETIMS                     |
  +-------------------------------------------------------SG24-2042-+*/

Call APILOAD "VMREXTMR"               /* APILOAD to get variables...  */
                                      /*... loaded from VMREXTMR COPY */

/* 'minuend' values: our starting point                               */

minuend_stamp = "1 Jan 2000 22:00:00" /* Initial Date and Time        */
minuend_stamp_length = Length(minuend_stamp)
minuend_stamp_format = vm_tmr_format_rexx_date_n  /* <- rexx date     */
minuend_stamp_bias = 0                /* Timezone offset is 0 (GMT)   */
minuend_stamp_window_type = 0
minuend_stamp_window_position = 0

/* 'subtrahend' values: our calculation (here: no subtraction)        */

subtrahend_stamp = "2/00:00:00"       /* Set subtract time to 2 days  */
subtrahend_stamp_length = Length(subtrahend_stamp)
subtrahend_stamp_format = vm_tmr_format_met /* Expressed in MET form. */
subtrahend_stamp_window_type = 0
subtrahend_stamp_window_position = 0
subtrahend_stamp_bias = 0

/* 'difference' values: how we want the result displayed              */

difference_stamp_buffer_size = 20     /* Set result buffer size       */
difference_stamp_buffer = Copies(" ",difference_stamp_buffer_size)
difference_stamp_format = vm_tmr_format_iso /* Want result ISO format */
difference_stamp_window_type = 0
difference_stamp_window_position = 0
difference_stamp_bias = 32400         /* Result timezone is 9 hours + */
```

Figure 42 (Part 1 of 2). VMTMDTS2 EXEC - CSL DateTimeSubtract Routine

```
/* CSL Call to DateTimeSubtract (check RTNMAP VMTMDTS)                */

Call CSL "DateTimeSubtract",
         "retcode reascode",
         "minuend_stamp",
         "minuend_stamp_length",
         "minuend_stamp_format",
         "minuend_stamp_bias",
         "minuend_stamp_window_type",
         "minuend_stamp_window_position",
         "subtrahend_stamp",
         "subtrahend_stamp_length",
         "subtrahend_stamp_format",
         "subtrahend_stamp_bias",
         "subtrahend_stamp_window_type",
         "subtrahend_stamp_window_position",
         "difference_stamp_buffer",
         "difference_stamp_buffer_size",
         "difference_stamp_length",
         "difference_stamp_format",
         "difference_stamp_bias",
         "difference_stamp_window_type",
         "difference_stamp_window_position"

/* Display selected variables with the Result */

Say "retcode="retcode "reascode="reascode
Say "minuend_stamp          :" minuend_stamp
Say "minuend_stamp_bias     :" minuend_stamp_bias
Say "subtrahend_stamp       :" subtrahend_stamp
Say "difference_stamp_bias  :" difference_stamp_bias
Say "difference_stamp_buffer:" difference_stamp_buffer
```

```
vmtmdts2
retcode=0 reascode=0
minuend_stamp          : 1 Jan 2000 22:00:00
minuend_stamp_bias     : 0
subtrahend_stamp       : 2/00:00:00
difference_stamp_bias  : 32400
difference_stamp_buffer: 1999-12-31 07:00:00.
Ready; T=0.01/0.02 08:13:41

                                                    Running   VMESA220
```

Figure 42 (Part 2 of 2). VMTMDTS2 EXEC - CSL DateTimeSubtract Routine

Similar to our previous *VMTMDTS EXEC*, this one uses a starting datestamp of January 1st, 2000 at 10PM (GMT), expressed in **REXX** format. Two days are subtracted, and the result is expressed in **ISO** format, with Australia as the timezone offset.

A.9 Sample REXX Date Function (HOLIDAYS EXEC)

Figure 43 on page 111 illustrates how to call the REXX *date* routine for date conversion and date arithmetic. The days on which statutory holidays will be taken for a given year are calculated. A technique is illustrated for using the date function to determine if a given date is valid. As well, the fact that REXX 'B' (Base) type dates can be used to determine the day of the week is exploited by the example. If called as a function the calculated dates are returned to the caller. If called as a command from the console the calculated dates are displayed.

```
/*REXX*/
/*+-HOLIDAYS EXEC-----------------------------------------------------+
  |  Find on what days the statutory holidays for a given year       |
  |  will be taken in British Columbia.                              |
  +------------------------------------------------------SG24-2042-+*/

  arg year .
  year = strip(year)

  if \ValiDate(year'0101','S') then do  /* valid? */
    say "You must provide a positive year of the form  nnnn"
    return 4
  end

/*
 * New Years Day
 * January 1st (unless a weekend)
 */
  nyd = date('B',year'0101','S')
  if nyd//7 > 4 then
    nyd = nyd + (7 - (nyd//7))  /* move to Monday */
  holidays.1 = nyd "New Years Day"

/*
 * Good Friday (Friday before Easter Sunday)
 * Easter Monday (Monday after Easter Sunday)
 *
 * In order to figure out these two we figure out Easter Sunday.
 * These calculations are adapted from those in the paper
 * Calendrical Calculations from Software Practice and Experience
 * Volume 20(9).
 *
 * Easter Sunday is defined as the first Sunday after the first
 * full moon occurring on or before the vernal equinox. Read the
 * paper for more details.
 */
  cent = (year%100) + 1          /* what century we're in */
  epact = ((year//19)*11)+14+,   /* Nicaean Easter rule */
          -((cent*3)%4) +,       /* adjusted for Gregorian */
          (((cent*8)+5)%25) +,   /* adjusted for Metonic cycle */
          (cent*30)
  epact = epact//30
  adj_ep = epact
  if (epact = 0) | ((epact = 1) & (10 < (year//19))) then
    adj_ep = adj_ep + 1          /* adjusted for 29.5 day months */

  es = date('B',year'0419','S') - adj_ep + 7   /* Paschal moon */
  if es//7 <> 6 then
    es = es - (es//7) - 1        /* force back to previous Sunday */
  holidays.2 = es-2 "Good Friday"
  holidays.3 = es+1 "Easter Monday"
```

Figure 43 (Part 1 of 4). HOLIDAYS EXEC - Illustrate the Use of REXX Date Conversion

```
/*
 * Queen's Birthday (aka Victoria Day)
 * first Monday on or before the 24th of May
 */
  qbd = date('B',year'0524','S')
  qbd = qbd - (qbd//7)  /* force to previous Monday */
  holidays.4 = qbd "Victoria Day"

/*
 * Dominion Day (aka Canada Day)
 * July 1st (unless a weekend)
 */
  dd = date('B',year'0701','S')
  if dd//7 > 4 then
    dd = dd + (7 - (dd//7))  /* move to Monday */
  holidays.5 = dd "Canada Day"

/*
 * BC Day
 * First Monday in August
 */
  bcd = date('B',year'0801','S')
  if bcd//7 <> 0 then
    bcd = bcd + (7 - (bcd//7))  /* move to Monday */
  holidays.6 = bcd "BC Day"

/*
 * Labour Day
 * First Monday in September
 */
  ld = date('B',year'0901','S')
  if ld//7 <> 0 then
    ld = ld + (7 - (ld//7))  /* move to Monday */
  holidays.7 = ld "Labour Day"

/*
 * Thanksgiving Day
 * Second Monday in October
 */
  td = date('B',year'1001','S') + 7
  if td//7 <> 0 then
    td = td + (7 - (td//7))   /* move to Monday */
  holidays.8 = td "Thanksgiving Day"

/*
 * Remembrance Day
 * November 11th (unless a weekend)
 */
  rd = date('B',year'1111','S')
  if rd//7 > 4 then
    rd = rd + (7 - (rd//7))  /* move to Monday */
  holidays.9 = rd "Remembrance Day"
```

Figure 43 (Part 2 of 4). HOLIDAYS EXEC - Illustrate the Use of REXX Date Conversion

```
/*
 * Christmas Day
 * December 25th (unless a weekend)
 */
  cd = date('B',year'1225','S')
  if cd//7 > 4 then
    cd = cd + (7 - (cd//7))  /* move to Monday */
  holidays.10 = cd "Christmas Day"

/*
 * Boxing Day
 * December 26th (unless a weekend)
 * (basically the day after Christmas)
 */
  bd = cd+1
  if bd//7 > 4 then
    bd = bd + (7 - (bd//7))  /* move to Monday (or Tuesday) */
  holidays.11 = bd "Boxing Day"

  holidays.0 = 11   /* count of holidays */

  parse source sysname call_type .
  if call_type = "COMMAND" then do
    do i = 1 to holidays.0
      parse value holidays.i with date name
      say date('W',date,'B') date('N',date,'B') "is" name
    end
  end; else
    if sysname = "CMS" then
      'PIPE STEM HOLIDAYS. ',
      '| STEM HOLIDAYS. 1'
    else
      say "Use a decent operating system"

return 0

/*
 *  return true if the given date is valid for the
 *              given date format
 *  return false if it is not
 */
ValiDate:
  signal on syntax name invalid_date
  call date arg(2),arg(1),arg(2)
  return (1==1)
invalid_date:
  return (0==1)
```

Figure 43 (Part 3 of 4). HOLIDAYS EXEC - Illustrate the Use of REXX Date Conversion

```
holidays 1997
Wednesday 1 Jan 1997 is New Years Day
Friday 28 Mar 1997 is Good Friday
Monday 31 Mar 1997 is Easter Monday
Monday 19 May 1997 is Victoria Day
Tuesday 1 Jul 1997 is Canada Day
Monday 4 Aug 1997 is BC Day
Monday 1 Sep 1997 is Labour Day
Monday 13 Oct 1997 is Thanksgiving Day
Tuesday 11 Nov 1997 is Remembrance Day
Thursday 25 Dec 1997 is Christmas Day
Friday 26 Dec 1997 is Boxing Day
Ready; T=0.02/0.03 02:34:28
```

Figure 43 (Part 4 of 4). HOLIDAYS EXEC - Illustrate the Use of REXX Date Conversion

Appendix B. Sample ISPF Routines for Year2000

Note

In this Appendix are some helpful examples that are intended for use with ISPF.

These samples are from various sources, and are provided as is; tailor to your specific needs and environment.

The complete set of samples will be available on the **CD-ROM** version of this Redbook in the newest update of the *System/390 Redbooks Collection (SBOF-7201)*. How to download these samples is described in Appendix E, "Installing the Samples from CD-ROM" on page 149.

ISPF Samples

- *ISPA2042 EXEC*

  This is the bootstrap routine, which does basic setup and then invokes the default ISPF startup routine (ISPSTART EXEC).

- *ISPB2042 EXEC*

  This exec performs the dialog management. It completes the setup processing, then displays the panel and processes the user actions.

- *PANEL1 COPY*

  This is the panel definition, and must be generated into ISPF2042 MACLIB as member PANEL1 for the sample to function correctly (`MACLIB GEN ISPF2042 PANEL1`)

B.1 ISPF Sample Set

```
/*+-ISPA2042 EXEC-----------------------------------------------------+
  | Sample ISPF bootstrap                                             |
  |                                                                   |
  | Checks that VM level provides Y2K support, then checks to see     |
  | if it is enabled.                                                 |
  | If all is OK, sets up for ISPF, then starts the dialogue          |
  |                                                                   |
  | For more information about ISPF, refer to;                        |
  | "ISPF and ISPF/PDF Primer"                    SC34-4270           |
  | "ISPF Dialog Management Guide and Reference"  SC34-4273           |
  |                                                                   |
  | For more information about Diag 00, refer to                      |
  | "VM/ESA V2R2.0 CP Programming Services"       SC24-5760           |
  |                                                                   |
  +-------------------------------------------------------SG24-2042-+*/

Trace 'O'
address command

/* Check the system level */

v2r2      = 0
v2r1_y2k  = 0

parse value diagrc('00') with rc +10 sense +6 diagdata
if rc \= 0 then do
  say 'Ending due to error while attempting to obtain system information'
  say 'Diagnose: 00 RC: 'rc' Sense : 'sense
  exit rc
end

/* In this example of the use of Diagnose 00, we examine the entire
   string of information, rather than individual bit settings        */

parse var diagdata 25 sys_level +8 .
v2r2      = (sys_level = x2c('7FFE000000000000')) /* VM/ESA V2R2.0   */
v2r1_y2k  = (sys_level = x2c('7FFC000000000000')) /* VM/ESA V2R1.0   */
                                                  /*(with Year 200 support)*/
if v2r2 | v2r1_y2k then y2k_OK = 1
else y2k_OK = 0

if \y2k_OK then do
  say 'This system is not at the correct level to give the required'
  say 'Year 2000 support'
  exit 8
end

if \cmsflag('YEAR2000') then do
  say'I cannot continue because your system does not have Year 2000',
     'support'
  exit 8
end
```

Figure 44 (Part 1 of 2). ISPA2042 EXEC - ISPF Date Sample Bootstrap

```
'FILEDEF ISPPLIB DISK ISPF2042 MACLIB A (PERM CONCAT' /* panel library*/
if rc \= 0 then do
  say 'Error : 'rc' from Filedef for panel library. Terminating'
  exit rc
end

'EXEC ISPSTART CMD(%ISPB2042)'              /* standard IPSF start   */
if rc \= 4 then exit rc
```

Figure 44 (Part 2 of 2). ISPA2042 EXEC - ISPF Date Sample Bootstrap

```
/*+-ISPB2042 EXEC-----------------------------------------------------+
  | Sample ISPF dialog management routine                             |
  |                                                                   |
  | Performs the setup required for this ISPF session, then loops     |
  | displaying the panel, and taking action as requested              |
  |                                                                   |
  | For more information about ISPF, refer to;                        |
  | "ISPF and ISPF/PDF Primer"                    SC34-4270-00        |
  | "ISPF Dialog Management Guide and Reference"  SC34-4273-00        |
  |                                                                   |
  +-------------------------------------------------------SG24-2042-+*/

Trace 'O'
Address Command

ispf_rc = 0
myname  = userid()
parse value diag(8,'QUERY USERID') with . 'AT' sysid '15'x .

/* get the date, time and dateformat information                     */

parse value diag(8,'QUERY TIME') with . . mytime . . mydate '15'x .
parse value diagrc(8,'QUERY DATEFORMAT') with rc . '=' myform '15'x .
if rc \= 0 then do
  say 'You cannot QUERY DATEFORMAT, rc : 'rc', I am unable to proceed'
  exit rc
end

/* Add my variables to the ISPF variable pool                        */

varlist = 'myname mytime mydate myform sysid'

address ISPEXEC 'VPUT ('VARLIST') ASIS'

parse value 'HELP ____ Quit ____ Dflt' with zpf01 zpf02 zpf03 zpf04 zpf05
parse value 'Date ____ ____ ____ ___' with zpf06 zpf07 zpf08 zpf09 zpf10
parse value '____ ____ HELP ____ Quit' with zpf11 zpf12 zpf13 zpf14 zpf15
parse value '____ Dflt Date ____ ____' with zpf16 zpf17 zpf18 zpf19 zpf20
parse value '____ ____ ____ ____'      with zpf21 zpf22 zpf23 zpf24

address ISPEXEC 'VPUT (ZPF01,ZPF02,ZPF03,ZPF04,ZPF05) '
ispf_rc = ispf_rc + rc
address ISPEXEC 'VPUT (ZPF06,ZPF07,ZPF08,ZPF09,ZPF10) '
ispf_rc = ispf_rc + rc
address ISPEXEC 'VPUT (ZPF11,ZPF12,ZPF13,ZPF14,ZPF15) '
ispf_rc = ispf_rc + rc
address ISPEXEC 'VPUT (ZPF16,ZPF17,ZPF18,ZPF19,ZPF20) '
ispf_rc = ispf_rc + rc
address ISPEXEC 'VPUT (ZPF21,ZPF22,ZPF23,ZPF24) '
ispf_rc = ispf_rc + rc
```

Figure 45 (Part 1 of 2). ISPB2042 EXEC - ISPF Date Sample Dialog Routine

```
if ispf_rc \= 0 then do
  say 'A problem has been encountered when storing ISPF variables.'
  say 'I am unable to continue.'
  exit ispf_rc
end

/* Now loop displaying the screen and processing user input as reqd   */

do forever
  pfkey = ''                                /* reset vars each time  */
  zcmd  = ''                                /* round                 */
                                            /* and get fresh date    */
                                            /* information           */
  parse value diag(8,'QUERY TIME') with . . mytime . . mydate '15'x .
  parse value diag(8,'QUERY DATEFORMAT') with '=' myform '15'x .

  address ISPEXEC 'DISPLAY PANEL(PANEL1)'   /* display the panel     */
  if rc \= 0 then do
    say 'Error : 'rc' trying to display ISPF panel. Terminating'
    exit rc
  end

  if words(myform) \= 1 then myform = word(myform,1)
  else myform = strip(myform)

  select                                    /* process user action   */
    when (pfkey = 'PF03' | pfkey = 'PF15' | pfkey = 'QUIT') then leave
    when (pfkey = 'PF05' | pfkey = 'PF17' | pfkey = 'DFLT') then,
         call diag 8,'SET DATEFORMAT SYSDEFAULT'
    when (pfkey = 'PF06' | pfkey = 'PF18' | pfkey = 'DATE') then do
      select
        when myform = 'SHORTDATE' then do
          call diag 8,'SET DATEFORMAT FULLDATE'
        end
        when myform = 'FULLDATE' then do
          call diag 8,'SET DATEFORMAT ISODATE'
        end
        when myform = 'ISODATE' then do
          call diag 8,'SET DATEFORMAT SHORTDATE'
        end
      otherwise call diag 8,'SET DATEFORMAT SYSDEFAULT'
      end                                   /* end select myform     */
    end
  otherwise nop
  end                                       /* end select pfkey      */
end                                         /* end 'forever'         */

/* HAVE to exit rc 4 to suppress Console disposition panel           */
exit 4
```

Figure 45 (Part 2 of 2). ISPB2042 EXEC - ISPF Date Sample Dialog Routine

```
)ATTR
/*--PANEL1 MEMBER-----------------------------------------------------*/
/*  Sample ISPF panel                                                 */
/*                                                                    */
/*  For more information about ISPF, refer to;                        */
/*  "ISPF and ISPF/PDF Primer"                    SC34-4270-00        */
/*  "ISPF Dialog Management Guide and Reference"  SC34-4273-00        */
/*                                                                    */
/*--------------------------------------------------------SG24-2042--*/
 % TYPE(TEXT)  INTENS(LOW) COLOR(RED)
 + TYPE(TEXT)  INTENS(LOW) COLOR(BLUE) SKIP(ON)
 ¢ TYPE(TEXT)  INTENS(LOW) COLOR(PINK)
 _ TYPE(INPUT) INTENS(LOW) COLOR(GREEN)
 $ TYPE(TEXT)  INTENS(LOW) COLOR(GREEN)
 @ TYPE(TEXT)  INTENS(LOW) COLOR(WHITE)
)BODY
@&SYSID            %SAMPLE ISPF PANEL - DATE FORMATS
_ZCMD+
¢ &INFMSG
+ --------------- $PRESS PF06 TO TOGGLE DATE FORMATS +-------------
+                 $PRESS PF05 TO RESTORE DEFAULT
+
¢     @SYSTEM INFORMATION
¢     +USERID     +TIME         +DATE        +DATEFORMAT
+
¢     $&MYNAME    $&MYTIME      $&MYDATE    $&MYFORM
+
+
¢     @ISPF INFORMATION
¢     +ZYEAR      +ZTIME        +ZDATE       +ZDATEF
+
+     $&ZYEAR     $&ZTIME       $&ZDATE      $&ZDATEF
+
+
+
)INIT
  .ZVARS = ''
  .CURSOR = ZCMD
)PROC
  &ZCMD = TRUNC (&ZCMD,4)
  &PFKEY = TRANS (&ZCMD HELP,HELP PF02,PF01 QUIT,QUIT PF04,PF16
                        PF05,PF17 DATE,DATE DFLT,DFLT PF08,PF20
                        PF09,PF21 PF10,PF22 PF11,PF23 PF12,PF24
                        *,' ')
  &ZCMD = ''
  REFRESH ZCMD
)END
```

Figure 46. ISPF2042 COPY - ISPF Date Sample Panel

See Figure 47 on page 121 for sample execution of *ISPA2042*.

```
VMESA220          SAMPLE ISPF PANEL - DATE FORMATS


---------------  PRESS PF06 TO TOGGLE DATE FORMATS  -------------
                 PRESS PF05 TO RESTORE DEFAULT

     SYSTEM INFORMATION
     USERID      TIME          DATE         DATEFORMAT

     TIBO        08:26:14      2002-07-07   ISODATE (SYSDEFAULT)


     ISPF INFORMATION
     ZYEAR       ZTIME         ZDATE        ZDATEF

     02          08:26         02/07/07     YY/MM/DD



F1=HELP  F2=____  F3=Quit  F4=____  F5=Dflt  F6=Date  F7=____  F8=____
F9=____ F10=____ F11=____ F12=____
```

```
VMESA220          SAMPLE ISPF PANEL - DATE FORMATS


---------------  PRESS PF06 TO TOGGLE DATE FORMATS  -------------
                 PRESS PF05 TO RESTORE DEFAULT

     SYSTEM INFORMATION
     USERID      TIME          DATE         DATEFORMAT

     TIBO        08:26:27      07/07/02     SHORTDATE


     ISPF INFORMATION
     ZYEAR       ZTIME         ZDATE        ZDATEF

     02          08:26         02/07/07     YY/MM/DD



F1=HELP  F2=____  F3=Quit  F4=____  F5=Dflt  F6=Date  F7=____  F8=____
F9=____ F10=____ F11=____ F12=____
```

Figure 47. Running the ISPF Samples

The first panel shows the default date format; in this case, the user has the system default, which was set to *ISOdate*. After pressing the **PF6** key, the date format toggles to *SHOrtdate*, and so on. Note the ISPF date format, which has its own format of **YY/MM/DD**.

Appendix C. Sample Pl/I Routines for Year2000

Note

In this Appendix are some helpful examples that are intended for use with PL/I.

These samples are from various sources, and are provided as is; tailor to your specific needs and environment.

The complete set of samples will be available on the **CD-ROM** version of this Redbook in the newest update of the *System/390 Redbooks Collection (SBOF-7201).* How to download these samples is described in Appendix E, “Installing the Samples from CD-ROM” on page 149.

PL/I Samples

- *PLISMP1 PLIOPT*

  Calls DMSERP (Extract/Replace function) to get the date and century of the last update for a named file. Invoked by PLISMP1 EXEC.

- *PLISMP2 PLIOPT*

  Calls VMTMDTG (Date/Time Get) to get current date, time, zone and epoch information. Then uses VMTMDTS (Date Time Subtract) to perform some simple conversions. Invoked by PLISMP2 EXEC.

C.1 PL/I Sample Set

The requirements for the compile environment for these samples are listed in the comments at the beginning of the PL/I source files.

The samples were compiled using:

```
IBM OS PL/I OPTIMIZING COMPILER  VER 2 REL 3 MOD 0
```

Screen images are included at the end of each sample to demonstrate the output expected from successful execution.

Note

The sample programs do **not** run successfully on versions of CMS that do not have Year2000 support.

C.1.1 PL/I Sample Set One

This sample sets consists of two files:

- **PLISMP1 EXEC** This exec establishes the runtime environment for the PL/I module and executes it.
- **PLISMP1 PLIOPT** The source for the module.

PLISMP1 uses the CSL DMSERP (extract and replace) function to extract the century indicator, and last update date and time stamp, for a user selected file. It uses the century indicator to determine the correct century value, and builds and displays an ISO format date of last update for the file.

When run, PLISMP1 will prompt you to enter the name, type and (optionally) mode of the file to be checked. If not entered, file mode will be defaulted to “*” and the first file meeting the remaining criteria will be selected. If the file does not exist, PLISMP1 will terminate with an error.

```
/*+-PLISMP1 EXEC-----------------------------------------------------+
  | Invoke sample PL/I Module for CSL calls                          |
  |                                                                  |
  | Performs the setup required for the module, then invokes it.     |
  | Reports if module terminates rc /= 0                             |
  |                                                                  |
  +-----------------------------------------------------SG24-2042-+*/

Trace 'O'

'FILEDEF SYSPRINT DISK PLISMP1  SYSPRINT A' /* for PL/I error output */
'GLOBAL TXTLIB PLILIB IBMLIB'               /* runtime libs          */
'GLOBAL LOADLIB PLILIB'

address command 'PLISMP1'
if rc \= 0 then do
  say ''
  say 'PLISMP1 has failed with RC : 'rc
  say 'Check for the following :'
  say '  - a valid file name was entered'
  say '  - extra messages from the PLISMP1 module'
  say '  - that your CSL supports the function(s) requested'
  say '  - for a PLISMP1 SYSPRINT file, if a PL/I error has occurred'
  exit rc
end
```

Figure 48 (Part 1 of 7). PLISMP1 - PL/I Sample Set One

```
*PROCESS S A X NEST FLAG(I) MAR(2,72,1) SEQ(73,80) OPTIMISE(2);
*PROCESS OFFSET AG;
*PROCESS MACRO;

 /*-PLISAMP1 PLIOPT-----------------------------------------------------*/
 /* Sample PL/I program to demonstrate a call to the CSL to obtain    */
 /* file update information, including the CENTURY indicator          */
 /*                                                                   */
 /* Compile environment:                                              */
 /* FILEDEF DMSGPI   DISK DMSGPI   MACLIB   *                         */
 /* GLOBAL  TXTLIB   IBMLIB CMSLIB VMLIB PLILIB CMSSAA                */
 /*                                                                   */
 /* For more information about PL/1 refer to:                         */
 /*  'IBM PL/I For MVS and VM Programming Guide'     SC26-3113        */
 /*  'IBM PL/I For MVS and VM Language Reference'    SC26-3114        */
 /*  'OS PL/I Programming Guide'                     SC26-4307        */
 /*  'OS PL/I Programming Language Reference'        SC26-4308        */
 /*                                                                   */
 /*                                                                   */
 /* For more information about the VM CSL refer to:                   */
 /*  'CMS Application Development Guide'             SC24-5761        */
 /*  'CMS Application Development Reference'         SC24-5762        */
 /*                                                                   */
 /*                                                                   */
 /*-------------------------------------------------------SG24-2042--*/


 PLISmp1: Procedure Options (MAIN) ReOrder;

 /*-------------------------------------------------------------------*/
 /* BuiltIN Functions                                                 */
 /*-------------------------------------------------------------------*/

 Declare (ADDR, BASED, INDEX, OnSource, PLIRetc, SUBSTR) BUILTIN;

 /*-------------------------------------------------------------------*/
 /* External Function Declarations (CSL)                              */
 /*-------------------------------------------------------------------*/

 Dcl DMSCSL ENTRY EXTERNAL OPTIONS (ASSEMBLER INTER);

 Dcl False             Bit(01)  Init('0'B);
 Dcl i                 Fixed bin(31) Init(0);
 Dcl k                 Bit(1)   Init('1');   /* loop control        */
 Dcl in_stream         Char(20) Varying;
 Dcl disp_date         Char(08) Init('');  /* display variables     */
 Dcl disp_time         Char(08) Init('');
 Dcl disp_cent         Char(02) Init('');
```

Figure 48 (Part 2 of 7). PLISMP1 - PL/I Sample Set One

```
/*-------------------------------------------------------------------*/
/* File information                                                  */
/*-------------------------------------------------------------------*/

Dcl 1 FILEINFO,
      2 FileName Char(08) Init(' '),     /* I : FILE NAME          */
      2 FileType Char(08) Init(' '),     /* I : FILE TYPE          */
      2 FileMode Char(01) Init(' '),     /* I : FILE MODE          */
      2 Retcode Fixed Bin(31) Init(0),   /* O : Rc of CSL DMSERP   */
      2 Fcent   Fixed Bin(31);           /* O : Century indicator  */

/* File DATE information                                           */

Dcl 1 Rawdate    Char(12);              /* O : File Date          */

Dcl 1 Filedate based(addr(Rawdate)),    /* formatted date         */
      2 yy       Char(02) Init(' '),
      2 mth      Char(02) Init(' '),
      2 dd       Char(02) Init(' '),
      2 hh       Char(02) Init(' '),
      2 mm       Char(02) Init(' '),
      2 ss       Char(02) Init(' ');

/*-------------------------------------------------------------------*/
/* CSL Call args and variables                                       */
/*-------------------------------------------------------------------*/

Dcl cslfunc     Char(08)      Init('DMSERP'); /* function to call    */
Dcl retcode     Fixed Bin(31) Init(0);
Dcl func        Char(08)      Init('EXTRACT');
Dcl numarg      Fixed Bin(31) Init(03);
Dcl infoname    Char(20)      Init('FILE_DATE_TIME_C');
Dcl buffer      Char(12)      Init(' ');
Dcl datatyp     Fixed Bin(31) Init(32);
Dcl buflen      Fixed Bin(31) Init(80);
Dcl flags       Char(08)      Init('00000000');
Dcl srchtyp     Char(04)      Init('AND ');
Dcl token       Fixed Bin(31) Init(0);
Dcl sargNAM_1   Char(20)      Init('FILE_NAME');
Dcl sargVAL_1   Char(08)      Init('');
Dcl svalTYP_1   Fixed Bin(31) Init(32);
Dcl svalLEN_1   Fixed Bin(31) Init(08);
Dcl sargTYP_1   Char(02)      Init('EQ');
Dcl sargNAM_2   Char(20)      Init('FILE_TYPE');
Dcl sargVAL_2   Char(08)      Init('');
Dcl svalTYP_2   Fixed Bin(31) Init(32);
Dcl svalLEN_2   Fixed Bin(31) Init(08);
Dcl sargTYP_2   Char(02)      Init('EQ');
Dcl sargNAM_3   Char(20)      Init('FILE_MODE');
Dcl sargVAL_3   Char(01)      Init('');
Dcl svalTYP_3   Fixed Bin(31) Init(32);
Dcl svalLEN_3   Fixed Bin(31) Init(01);
Dcl sargTYP_3   Char(02)      Init('EQ');
```

Figure 48 (Part 3 of 7). PLISMP1 - PL/I Sample Set One

```
/*--------------------------------------------------------------------*/
/* Error handling                                                     */
/*--------------------------------------------------------------------*/

on error begin;                          /* on major errors just  */
  call pliretc(12);                      /* exit, RC 12           */
  stop;
end;

on conversion begin;                     /* for convs, reset var  */
  onsource() = 999;                      /* in error and retry    */
  display('Conversion error encountered, retrying');
end;                                     /* will probably term.   */

/*--------------------------------------------------------------------*/
/* Execution begins .......                                           */
/* Reset the CSL - don't really expect an error here, so checks are  */
/* primitive                                                          */
/*--------------------------------------------------------------------*/

Call DMSCSL(CSLFUNC,RETCODE,'RESET   ');
if retcode ¬= 0 then do;
  call pliretc(retcode);
  stop;
end;

/*--------------------------------------------------------------------*/
/* now loop getting user input of file name,type (and optional) mode */
/* Leave the loop by entering QUIT                                    */
/*--------------------------------------------------------------------*/

do while (k);
  display('Enter File Name, Type & Mode (or QUIT)') reply (in_stream);
  if in_stream = 'QUIT' then do;         /* request to leave      */
    k = (false);
  end;
  else do;                               /* else assume work to do*/
    i = index(in_stream,' ' );           /* more than 1 word ?    */
    if i ¬= 0 then do;                   /* YES !                 */
      FileName  = substr(in_stream,1,i); /* extract the fname     */
      in_stream = substr(in_stream,i+1); /* redefine the input    */
      i = index(in_stream,' ' );         /* still > 1 word ?      */
      if i = 0 then FileType = substr(in_stream,1); /*NO must be ft */
      else do;                           /* YES must be ft and fm */
        FileType  = substr(in_stream,1,i); /* so get them         */
        FileMode  = substr(in_stream,i+1);
      end;
    end;                                 /* if either fn or ft is */
    if (filename = ' ' | filetype = ' ') then do; /* blank, then    */
      filename = ' ';                    /* reset for next loop   */
      filetype = ' ';
      filemode = ' ';
    end;                                 /* if fn & ft NOT blank  */
```

Figure 48 (Part 4 of 7). PLISMP1 - PL/I Sample Set One

```
    else do;                                  /* then proc the CSL call*/
      sargVAL_1 = FileInfo.FileName;          /* setup CSL args        */
      sargVAL_2 = FileInfo.FileType;
      sargVAL_3 = FileInfo.FileMode;
      retcode   = 0;
      fileinfo.retcode = 0;
      disp_date = '';
      disp_time = '';
      disp_cent = '';

      infoname  = 'FILE_DATE_TIME_C';         /* args for call to get  */
      datatyp   = 32;                         /* file update stamp     */
      buffer    = ' ';
      buflen    = 12;
      call CSL_call;
      Rawdate   = buffer;
      Fileinfo.Retcode = Fileinfo.Retcode + retcode;

      infoname  = 'FILE_DATE_CENTURY';        /* args for call to get   */
      datatyp   = 9;                          /* file update century    */
      buffer    = ' ';
      buflen    = 1;
      retcode   = 0;
      call CSL_call;
      Fileinfo.Fcent   = buffer;

      if (Fileinfo.Retcode = 0 & retcode = 0) then do;
        disp_date = filedate.yy||'-'||filedate.mth||'-'
                    ||filedate.dd;
        disp_time = filedate.hh||':'||filedate.mm||':'
                    ||filedate.ss;
        disp_cent = substr('1920',(buffer=1)*2+1,2);

        display('File Updated on : '||disp_cent||disp_date);
        display('             at : '||disp_time);
      end;
      else do;
        display('Call to CSL DMSERP has failed');
        display('Return code : '||fileinfo.retcode);
        display('Flags       : '||flags);

        call pliretc(fileinfo.retcode);
        stop;
      end;
    end;
  end;
end;                                          /* end 'K'               */
return;
```

Figure 48 (Part 5 of 7). PLISMP1 - PL/I Sample Set One

```
/*-------------------------------------------------------------------*/
/* CSL_Call   : execute the call to the CSL function                 */
/*                                                                   */
/* Parms      : none                                                 */
/* Returns    : nothing                                              */
/* Notes      : caller is responsible for setting up the CSL args,   */
/*              AND for saving extracted information                 */
/*-------------------------------------------------------------------*/

CSL_CALL: Procedure;

If FileInfo.Filemode = '*' |
   FileInfo.Filemode = ' ' Then Do;
   Numarg = 2; /* only two search arguments */
   Call DMSCSL(CSLFUNC,RETCODE,FUNC,NUMARG,InfoName
    ,buffer,DATATYP,buflen,FLAGS,SRCHTYP,TOKEN
    ,sargNAM_1,sargVAL_1,svalTYP_1,svalLEN_1,Sargtyp_1
    ,sargNAM_2,sargVAL_2,svalTYP_2,svalLEN_2,sargTYP_2);
   End;
Else Do;
   CALL DMSCSL(CSLFUNC,RETCODE,FUNC,NUMARG,InfoName
    ,buffer,DATATYP,BUFLEN,FLAGS,SRCHTYP,TOKEN
    ,sargNAM_1,sargVAL_1,svalTYP_1,svalLEN_1,sargTYP_1
    ,sargNAM_2,sargVAL_2,svalTYP_2,svalLEN_2,sargTYP_2
    ,sargNAM_3,sargVAL_3,svalTYP_3,svalLEN_3,sargTYP_3);
   End;


Return;
End CSL_CALL;

END PLIsmp1;
```

Figure 48 (Part 6 of 7). PLISMP1 - PL/I Sample Set One

```
q cmslevel
CMS Level 13, Service Level 705
Ready; T=0.01/0.01 04:29:24

plismp1
Enter File Name, Type & Mode (or QUIT)
test file a
File Updated on : 1997-05-30
             at : 10:53:01
Enter File Name, Type & Mode (or QUIT)
temp file a
File Updated on : 2000-01-03
             at : 03:42:39
Enter File Name, Type & Mode (or QUIT)
quit
Ready; T=0.05/0.07 04:29:48




                                                        RUNNING   IE157
```

Figure 48 (Part 7 of 7). PLISMP1 - PL/I Sample Set One

C.1.2 PL/I Sample Set Two

This sample set consists of three files:

- **PLISMP2 EXEC** This exec establishes the runtime environment for the PL/I module and executes it.
- **PLISMP2 PLIOPT** The source for the module.
- **VMPLITMR COPY** The constants and function names for CSL Timer services.

To compile PLISMP2 PLIOPT successfully, PLICSL MACLIB must be available to the compiler, and must contain the VMPLITMR COPY file as member VMPLITMR. The MACLIB can be generated with the command:

```
MACLIB GEN PLICSL VMPLITMR
```

Alternatively, the VMPLITMR COPY file may be imbedded into the source of the program.

When run, PLISMP2 displays the current system date, time, time zone and epoch. It then performs some simple date calculations and transformations and displays the results.

```
/*+-PLISMP2 EXEC-----------------------------------------------------+
  | Invoke sample PL/I Module for CSL Date & Time function calls     |
  |                                                                  |
  | Performs the setup required for the module, then invokes it.     |
  | Reports if module terminates rc /= 0                             |
  |                                                                  |
  +-----------------------------------------------------SG24-2042-+*/

Trace 'O'

'FILEDEF SYSPRINT DISK PLISMP2  SYSPRINT A' /* for PL/I error output */
'GLOBAL TXTLIB PLILIB IBMLIB'               /* runtime libs          */
'GLOBAL LOADLIB PLILIB'

address command 'PLISMP2'
if rc \= 0 then do
  say ''
  say 'PLISMP2 has failed with RC : 'rc
  say 'Check for the following :'
  say '  - extra messages from the PLISMP2 module'
  say '  - that your CSL supports the function(s) requested'
  say '  - for a PLISMP2 SYSPRINT file, if a PL/I error has occurred'
  exit rc
end
```

Figure 49 (Part 1 of 12). PLISMP2 - PL/I Sample Set Two

```
*PROCESS S A X NEST FLAG(I) MAR(2,72,1) SEQ(73,80) OPTIMISE(2);
*PROCESS OFFSET AG;
*PROCESS MACRO;

 /*-PLISMP2 PLIOPT-----------------------------------------------------*/
 /* Sample PL/I program to demonstrate a call to the DateTimeSubtract */
 /* function of CMS Multitasking Services                             */
 /* Uses home grown VMPLITMR COPY file                                */
 /*                                                                   */
 /* Compile environment:                                              */
 /* FILEDEF PLICSL   DISK PLICSL   MACLIB   *                         */
 /* FILEDEF DMSGPI   DISK DMSGPI   MACLIB   *                         */
 /* GLOBAL  TXTLIB   IBMLIB CMSLIB VMLIB PLILIB CMSSAA                */
 /*                                                                   */
 /* For more information about PL/1 refer to:                        */
 /*  'IBM PL/I For MVS and VM Programming Guide'     SC26-3113        */
 /*  'IBM PL/I For MVS and VM Language Reference'    SC26-3114        */
 /*  'OS PL/I Programming Guide'                     SC26-4307        */
 /*  'OS PL/I Programming Language Reference'        SC26-4308        */
 /*                                                                   */
 /*                                                                   */
 /* For more information about the CMS MultiTasking functions,        */
 /* and the samples shown, refer to:                                  */
 /*  'CMS Application MultiTasking'                  SC24-5766        */
 /*                                                                   */
 /*                                                                   */
 /*-------------------------------------------------------SG24-2042--*/


 PLISmp2: Procedure Options (MAIN) ReOrder;

 /*-------------------------------------------------------------------*/
 /* BuiltIN Functions                                                 */
 /*-------------------------------------------------------------------*/

 Declare (Length, OnSource, PLIRetc, Substr) BUILTIN;

 /*-------------------------------------------------------------------*/
 /* External Function Declarations (CSL)                              */
 /*-------------------------------------------------------------------*/

 Dcl DMSCSL ENTRY EXTERNAL OPTIONS (ASSEMBLER INTER);

 /*-------------------------------------------------------------------*/
 /* Load home grown VMPLITMR COPY file                                */
 /*-------------------------------------------------------------------*/

 %INCLUDE PLICSL(VMPLITMR);
 Dcl CslRetCode                    Fixed Bin(31) Init(0);
 Dcl CslReasCode                   Fixed Bin(31) Init(0);
```

Figure 49 (Part 2 of 12). PLISMP2 - PL/I Sample Set Two

```
/*--------------------------------------------------------------------*/
/* Variables for DateTimeGet and date calculations                    */
/*--------------------------------------------------------------------*/

Dcl Get_Date                          Char(10) Init(' ');
Dcl Get_Time                          Char(08) Init(' ');
Dcl Get_Zone                          Char(03) Init(' ');
Dcl Get_Epoch                         Fixed Bin(31) Init(0);

Dcl start_date                        Char(20) Init(' ');
Dcl start_date_length                 Fixed Bin(31) Init(0);
Dcl start_date_format                 Fixed Bin(31) Init(0);
Dcl start_date_bias                   Fixed Bin(31) Init(0);
Dcl start_date_window_type            Fixed Bin(31) Init(0);
Dcl start_date_window_position        Fixed Bin(31) Init(0);
Dcl subtract_date                     Char(20) Init(' ');
Dcl subtract_date_length              Fixed Bin(31) Init(0);
Dcl subtract_date_format              Fixed Bin(31) Init(0);
Dcl subtract_date_bias                Fixed Bin(31) Init(0);
Dcl subtract_date_window_type         Fixed Bin(31) Init(0);
Dcl subtract_date_window_position     Fixed Bin(31) Init(0);
Dcl difference_buffer                 Char(80) Init(' ');
Dcl difference_size                   Fixed Bin(31) Init(0);
Dcl difference_length                 Fixed Bin(31) Init(0);
Dcl difference_format                 Fixed Bin(31) Init(0);
Dcl difference_bias                   Fixed Bin(31) Init(0);
Dcl difference_window_type            Fixed Bin(31) Init(0);
Dcl difference_window_pos             Fixed Bin(31) Init(0);

/*--------------------------------------------------------------------*/
/* Error handling                                                     */
/*--------------------------------------------------------------------*/

on error begin;                              /* on major errors just  */
  call pliretc(12);                          /* exit, RC 12           */
  stop;
end;

on conversion begin;                         /* for convs, reset var  */
  onsource() = 999;                          /* in error and retry    */
  display('Conversion error encountered, retrying');
end;                                         /* will probably term.   */

/*--------------------------------------------------------------------*/
/* Execution begins .......                                           */
/*--------------------------------------------------------------------*/


CslRetCode  = 0;                             /* reset each time       */
CslReasCode = 0;
start_date_format          = vm_tmr_format_iso;
```

Figure 49 (Part 3 of 12). PLISMP2 - PL/I Sample Set Two

```
Call DMSCSL(DateTimeGet,CslRetCode,CslReasCode,
         start_date_format,
         Get_Date,
         Get_Time,
         Get_Zone,
         Get_Epoch);

if ((CslRetCode + CslReasCode) ¬= vm_tmr_success) then do;
  display('Call to DMSCSL DateTimeGet has failed');
  Select (CslRetCode);
    When (vm_tmr_warning) do;
      display('with a WARNING');
    end;
    When (vm_tmr_error) do;
      display('with an ERROR');
    end;
  otherwise do;
    display('Return code : '||CslRetCode);
    end;
  end;                                      /* end select            */
  display('Reason code : '||CslReasCode);
  call pliretc(CslRetCode);
  stop;
end;
else do;
  display('GetTime Output');
  display('Date '||Get_Date||' Time  '||Get_Time);
  display('Zone '||Get_Zone||' Epoch '||Get_Epoch);
end;
display(' ');

/*--------------------------------------------------------------------*/
/* Start of Sample 1                                                  */
/*--------------------------------------------------------------------*/

start_date                 = '12/31/1995  09:00:00';
start_date_length          = length(start_date);
start_date_format          = vm_tmr_format_usa;
start_date_bias            = -18000;
start_date_window_type     = vm_tmr_window_fixed;
start_date_window_position = 0;

subtract_date              = '0/00:00:00';
subtract_date_length       = length(subtract_date);
subtract_date_format       = vm_tmr_format_met;
subtract_date_bias         = 0;
subtract_date_window_type  = vm_tmr_window_fixed;
subtract_date_window_position = 0;

difference_buffer          = ' ';
difference_size            = 32;
difference_length          = 0;
difference_format          = vm_tmr_format_usa;
difference_bias            = -28800;
difference_window_type     = vm_tmr_window_sliding;
difference_window_pos      = 0;
```

Figure 49 (Part 4 of 12). PLISMP2 - PL/I Sample Set Two

```
call csl_call('Sample 1 - convert 12/31/1995 09:00:00 EST to PST,'
              ||' keep same format');

/*--------------------------------------------------------------------*/
/* Start of Sample 2                                                  */
/*--------------------------------------------------------------------*/

start_date                 = '12/31/1995 09:00:00';
start_date_length          = length(start_date);
start_date_format          = vm_tmr_format_usa;
start_date_bias            = -18000;
start_date_window_type     = vm_tmr_window_fixed;
start_date_window_position = 0;

subtract_date              = '0000000000000000'x;
subtract_date_length       = 8;
subtract_date_format       = vm_tmr_format_tod_relative;
subtract_date_bias         = 0;
subtract_date_window_type  = vm_tmr_window_fixed;
subtract_date_window_position = 0;

difference_buffer          = ' ';
difference_size            = length(difference_buffer);
difference_length          = 0;
difference_format          = vm_tmr_format_pipe;
difference_bias            = -18000;
difference_window_type     = vm_tmr_window_sliding;
difference_window_pos      = 0;

call csl_call('Sample 2 - convert 12/31/1995 09:00:00 EST to CMS'
              ||' Pipeline format');

/*--------------------------------------------------------------------*/
/* Start of Sample 3                                                  */
/*--------------------------------------------------------------------*/

start_date                 = '0025613602932E00'x;
start_date_length          = 8;
start_date_format          = vm_tmr_format_tod_absolute;
start_date_bias            = 0;
start_date_window_type     = vm_tmr_window_fixed;
start_date_window_position = 0;

subtract_date              = '0/03:00:00';
subtract_date_length       = length(subtract_date);
subtract_date_format       = vm_tmr_format_met;
subtract_date_bias         = 0;
subtract_date_window_type  = vm_tmr_window_fixed;
subtract_date_window_position = 0;

difference_buffer          = ' ';
difference_size            = length(difference_buffer);
difference_length          = 0;
difference_format          = vm_tmr_format_pipe;
```

Figure 49 (Part 5 of 12). PLISMP2 - PL/I Sample Set Two

```
difference_bias            = -21600;
difference_window_type     = vm_tmr_window_sliding;
difference_window_pos      = 0;

call csl_call('Sample 3 - subtract 3 hours from a TOD value, show in'
              ||' CMS Pipeline format');

/*--------------------------------------------------------------------*/
/* Start of Sample 4                                                  */
/*--------------------------------------------------------------------*/

start_date                 = '12/31/95 09:00:00';
start_date_length          = length(start_date);
start_date_format          = vm_tmr_format_usa_short;
start_date_bias            = -18000;
start_date_window_type     = vm_tmr_window_fixed;
start_date_window_position = 1900;

subtract_date              = '0/00:00:00';
subtract_date_length       = length(subtract_date);
subtract_date_format       = vm_tmr_format_met;
subtract_date_bias         = 0;
subtract_date_window_type  = vm_tmr_window_fixed;
subtract_date_window_position = 0;

difference_buffer          = ' ';
difference_size            = length(difference_buffer);
difference_length          = 0;
difference_format          = vm_tmr_format_iso;
difference_bias            = -28800;
difference_window_type     = vm_tmr_window_fixed;
difference_window_pos      = 0;

call csl_call('Sample 4 - convert 12/31/1995 09:00:00 SET to PST and'
              ||' show in ISO format');

/*--------------------------------------------------------------------*/
/* Start of Sample 5                                                  */
/*--------------------------------------------------------------------*/

start_date                 = '12/31/1995 09:00:00';
start_date_length          = length(start_date);
start_date_format          = vm_tmr_format_usa;
start_date_bias            = -18000;
start_date_window_type     = vm_tmr_window_fixed;
start_date_window_position = 0;

subtract_date              = '0/03:00:00';
subtract_date_length       = length(subtract_date);
subtract_date_format       = vm_tmr_format_met;
subtract_date_bias         = 0;
subtract_date_window_type  = vm_tmr_window_fixed;
subtract_date_window_position = 0;

difference_buffer          = ' ';
difference_size            = length(difference_buffer);
```

Figure 49 (Part 6 of 12). PLISMP2 - PL/I Sample Set Two

```
difference_length          = 0;
difference_format          = vm_tmr_format_usa_short;
difference_bias            = -28800;
difference_window_type     = vm_tmr_window_sliding;
difference_window_pos      = -50;

call csl_call('Sample 5 - take 3 hours from 12/31/1995 09:00:00 '
              ||'EST, show as PST in US format');

return;

/*--------------------------------------------------------------------*/
/* CSL_Call   : execute the call to the CSL function                  */
/*                                                                    */
/* Parms      : none                                                  */
/* Returns    : nothing                                               */
/* Notes      : caller is responsible for setting up the CSL args,    */
/*              AND for saving extracted information                  */
/*--------------------------------------------------------------------*/

CSL_CALL: Procedure (Msg_Text);

Dcl Msg_Text                          Char(320);

CslRetCode  = 0;                            /* reset each time       */
CslReasCode = 0;

Call DMSCSL(DateTimeSubtract,CslRetCode,CslReasCode,
         start_date,
         start_date_length,
         start_date_format,
         start_date_bias,
         start_date_window_type,
         start_date_window_position,
         subtract_date,
         subtract_date_length,
         subtract_date_format,
         subtract_date_bias,
         subtract_date_window_type,
         subtract_date_window_position,
         difference_buffer,
         difference_size,
         difference_length,
         difference_format,
         difference_bias,
         difference_window_type,
         difference_window_pos);
```

Figure 49 (Part 7 of 12). PLISMP2 - PL/I Sample Set Two

```
if ((CslRetCode + CslReasCode) ¬= vm_tmr_success) then do;
  display('Call to DMSCSL DateTimeSubtract has failed');
  Select (CslRetCode);
    When (vm_tmr_warning) do;
      display('with a WARNING');
    end;
    When (vm_tmr_error) do;
      display('with an ERROR');
    end;
  otherwise do;
    display('Return code : '||CslRetCode);
    end;
  end;                                        /* end select           */
  display('Reason code : '||CslReasCode);
  call pliretc(CslRetCode);
  stop;
end;
else do;
  do while (Msg_Text ¬= ' ');
    display(substr(Msg_Text,1,80));
    Msg_Text = substr(Msg_Text,81);
  end;
  display('Result  : '||substr(difference_buffer,1,difference_length));
  display(' ');
end;

Return;
End CSL_CALL;

END PLIsmp2;
```

Figure 49 (Part 8 of 12). PLISMP2 - PL/I Sample Set Two

```
/************************************************************/
/*                                                          */
/*  NAME      - Timer Services binding file for PL/I        */
/*                                                          */
/*  FUNCTION  - Defines the timer services constants and    */
/*              function definitions.                       */
/*                                                          */
/*  STATUS    - SG24-2042                                   */
/*                                                          */
/************************************************************/
/*                                                          */
/************************************************************/

/*----------------------------------------------------------*/
/*     Constants for Timer Services functions               */
/*----------------------------------------------------------*/

/*----------------------------------------------------------*/
/*     Basic error codes                                    */
/*----------------------------------------------------------*/

Dcl vm_tmr_success                  Fixed Bin(31) Init(0);
Dcl vm_tmr_warning                  Fixed Bin(31) Init(4);
Dcl vm_tmr_error                    Fixed Bin(31) Init(8);

/*----------------------------------------------------------*/
/*     Timer types                                          */
/*----------------------------------------------------------*/

Dcl VM_Tmr_TimerType_Real           Fixed Bin(31) Init(0);
Dcl VM_Tmr_TimerType_CPU            Fixed Bin(31) Init(1);

/*----------------------------------------------------------*/
/*     Timer cycles                                         */
/*----------------------------------------------------------*/

Dcl VM_Tmr_Cycle_Single             Fixed Bin(31) Init(0);
Dcl VM_Tmr_Cycle_Cyclical           Fixed Bin(31) Init(1);

/*----------------------------------------------------------*/
/*     Interval units                                       */
/*----------------------------------------------------------*/

Dcl VM_Tmr_IntUnit_Micro            Fixed Bin(31) Init(0);
Dcl VM_Tmr_IntUnit_Milli            Fixed Bin(31) Init(1);

/*----------------------------------------------------------*/
/*     Time zone                                            */
/*----------------------------------------------------------*/

Dcl VM_Tmr_Zone_Local               Fixed Bin(31) Init(0);
Dcl VM_Tmr_Zone_GMT                 Fixed Bin(31) Init(1);
```

Figure 49 (Part 9 of 12). PLISMP2 - PL/I Sample Set Two

```
/*-------------------------------------------------------------*/
/*     Function definitions for timer services functions       */
/*-------------------------------------------------------------*/
/*     DateTimeGet function definition                         */
/*-------------------------------------------------------------*/

Dcl DateTimeGet                      Char(8) Init('VMTMDTG');

/*-------------------------------------------------------------*/
/*     TimerStartInt function definition                       */
/*-------------------------------------------------------------*/

Dcl TimerStartInt                    Char(8) Init('VMTMSIM');

/*-------------------------------------------------------------*/
/*     DateTimeSubtract function definition                    */
/*-------------------------------------------------------------*/

Dcl DateTimeSubtract                 Char(8) Init('VMTMDTS');

/*-------------------------------------------------------------*/
/*     TimerStartTod    function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerStartTod                    Char(8) Init('VMTMSTD');

/*-------------------------------------------------------------*/
/*     TimerStartMicros function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerStartMicros                 Char(8) Init('VMTMSMI');

/*-------------------------------------------------------------*/
/*     TimerTest        function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerTest                        Char(8) Init('VMTMTST');

/*-------------------------------------------------------------*/
/*     TimerTestMicros  function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerTestMicros                  Char(8) Init('VMTMTSM');

/*-------------------------------------------------------------*/
/*     TimerStop        function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerStop                        Char(8) Init('VMTMSTP');

/*-------------------------------------------------------------*/
/*     TimerStopMicros  function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerStopMicros                  Char(8) Init('VMTMSTM');
```

Figure 49 (Part 10 of 12). PLISMP2 - PL/I Sample Set Two

```
/*-------------------------------------------------------------*/
/*     TimerStopAll     function definition                    */
/*-------------------------------------------------------------*/

Dcl TimerStopAll                    Char(8) Init('VMTMSTA');

/*-------------------------------------------------------------*/
/*     Date and Time Formats                                   */
/*     NOTE: a PL/I compiler restriction requires the          */
/*           abbreviation of some of the variable              */
/*           names compared to other language copy files       */
/*                                                             */
/*-------------------------------------------------------------*/

Dcl vm_tmr_format_usa               Fixed Bin(31) Init(0);
Dcl vm_tmr_format_eur               Fixed Bin(31) Init(1);
Dcl vm_tmr_format_iso               Fixed Bin(31) Init(2);
Dcl vm_tmr_format_julian            Fixed Bin(31) Init(3);
Dcl vm_tmr_format_met               Fixed Bin(31) Init(4);
Dcl vm_tmr_format_sci_absolute      Fixed Bin(31) Init(5);
Dcl vm_tmr_format_pipe              Fixed Bin(31) Init(6);
Dcl vm_tmr_format_sci_relative      Fixed Bin(31) Init(7);
Dcl vm_tmr_format_tod_absolute      Fixed Bin(31) Init(8);
Dcl vm_tmr_format_tod_relative      Fixed Bin(31) Init(9);
Dcl vm_tmr_format_usa_short         Fixed Bin(31) Init(10);
Dcl vm_tmr_format_eur_short         Fixed Bin(31) Init(11);
Dcl vm_tmr_format_julian_short      Fixed Bin(31) Init(12);
Dcl vm_tmr_format_iso_short         Fixed Bin(31) Init(13);
Dcl vm_tmr_format_pipe_short        Fixed Bin(31) Init(14);
Dcl vm_tmr_format_rexx_time_l       Fixed Bin(31) Init(15);
Dcl vm_tmr_format_rexx_time_e       Fixed Bin(31) Init(16);
Dcl vm_tmr_format_rexx_date_b       Fixed Bin(31) Init(17);
Dcl vm_tmr_format_rexx_date_n       Fixed Bin(31) Init(18);
Dcl vm_tmr_format_rexx_date_n_short Fixed Bin(31) Init(19);
Dcl vm_tmr_format_db2               Fixed Bin(31) Init(20);
Dcl vm_tmr_format_db2_short         Fixed Bin(31) Init(21);
Dcl vm_tmr_format_csl               Fixed Bin(31) Init(22);
Dcl vm_tmr_format_csl_short         Fixed Bin(31) Init(23);

/*-------------------------------------------------------------*/
/*     Bias types                                              */
/*-------------------------------------------------------------*/

Dcl vm_tmr_bias_local    Fixed Bin(31) Init(2147483647);

/*-------------------------------------------------------------*/
/*     Window types for DateTimeSubtract                       */
/*-------------------------------------------------------------*/

Dcl vm_tmr_window_fixed             Fixed Bin(31) Init(0);
Dcl vm_tmr_window_sliding           Fixed Bin(31) Init(1);
Dcl vm_tmr_window_none              Fixed Bin(31) Init(2);
```

Figure 49 (Part 11 of 12). PLISMP2 - PL/I Sample Set Two

```
q cmslevel
CMS Level 13, Service Level 705
Ready; T=0.01/0.01 04:56:37

plismp2
GetTime Output
Date 2000-01-04 Time  04.56.43
Zone NZS Epoch    -1139059893

Sample 1 - convert 12/31/1995 09:00:00 EST to PST, keep same format
Result  : 12/31/1995 06:00:00.000000

Sample 2 - convert 12/31/1995 09:00:00 EST to CMS Pipeline format
Result  : 19951231090000000000

Sample 3 - subtract 3 hours from a TOD value, show in CMS Pipeline format
Result  : 19000130083155512115

Sample 4 - convert 12/31/1995 09:00:00 SET to PST and show in ISO format
Result  : 1995-12-31 06:00:00.000000

Sample 5 - take 3 hours from 12/31/1995 09:00:00 EST, show as PST in US format
Result  : 12/31/95 03:00:00.000000

Ready; T=0.07/0.09 04:56:43



                                                          RUNNING   IE157
```

Figure 49 (Part 12 of 12). PLISMP2 - PL/I Sample Set Two

Appendix D. Sample Scan Tool Routines for Year2000

Note

In this Appendix is an example of the tool that may be employed to locate date usage in programs.

This sample is provided as is; tailor to your specific needs and environment.

The complete set of samples will be available on the **CD-ROM** version of this Redbook in the newest update of the *System/390 Redbooks Collection (SBOF-7201)*. How to download these samples is described in Appendix E, "Installing the Samples from CD-ROM" on page 149.

Code Scanning Sample

- *SEEK EXEC*

Sample exec to scan files for a given string

D.1 Sample File Scanning Routine (SEEK EXEC)

This REXX EXEC accepts a string and a file specification. It searches all files that match the given specification for the given string. While the exec is running it may be queried about its progress or interrupted.

```
/*+-SEEK EXEC-------------------------------------------------------+
  | Sample exec for scanning CMS files for a specified string.      |
  +-----------------------------------------------------SG24-2042-+*/

   address command

   parse arg searcharg fileid

   searcharg = strip(searcharg)
   fileid = strip(fileid subword('* * A',words(fileid)+1))

   if searcharg = '' | searcharg = '?' then do
      'HELP SEEK'
      exit 0
   end
/*
**       Set up immediate commands.
*/
   'IMMCMD SET HI'
   'IMMCMD SET ?'
   call diag 08,'SET MDCACHE INSERT OFF',1
/*
**       Put report heading.
*/
   'PIPE LITERAL',
         "* Seeking ("searcharg") within files '"fileid"'.",
         '| > SEEK LIST A'
/*
**       Issue listfile.
*/
   address cms 'LISTFILE' fileid '( STACK'
   j = queued()
   do i = 1 to j
      'IMMCMD STATUS HI'
      if rc = 1 then call quit 1, 'Seek halted scanning for string' ,
                                   searcharg 'within file' i 'of' j'.'
      'IMMCMD STATUS ?'
      if rc = 1 then do
         say 'Seeking string' searcharg 'within file' i 'of' j || '.'
         end
      parse pull nextfile
      'PIPE <' nextfile,
           '| XLATE',
           '| SPECS NUMBER 1 1-* NW',
           '| LOCATE 12-* /'translate(searcharg)'/',
           '| SPECS W 1',
           '| JOIN * / /',
           '| SPECS /'nextfile'/ 1 1-* NW',
           '| >> SEEK LIST A'
   end
```

Figure 50 (Part 1 of 3). SEEK EXEC - Sample File Scanning Routine

```
   'EXEC FILELIST SEEK LIST A ( FILELIST'
   call quit 0
exit    /* never taken */
/*
** Quit: Issue error message, clear stack and exit.
*/
quit:
   parse arg retc, message
   if message \= '' then
     say message
   'DROPBUF 0'
   'IMMCMD CLEAR HI'
   'IMMCMD CLEAR ?'
   call diag 08,'SET MDCACHE INSERT ON',1
exit retc
```

Figure 50 (Part 2 of 3). SEEK EXEC - Sample File Scanning Routine

```
seek date * plisql c
?
Seeking string date within file 23 of 759.
?
Seeking string date within file 65 of 759.
hi
Seek halted scanning for string date within file 100 of 759.
Ready(00001); T=1.41/1.56 15:36:27
```

Figure 50 (Part 3 of 3). SEEK EXEC - Sample File Scanning Routine

Appendix E. Installing the Samples from CD-ROM

E.1 Description

To type in samples from a book can be rather tiring and error prone. We, therefore, decided to make all samples documented in this book (plus the ones which were used during this case study) available in machine-readable form.

To keep things as simple and straightforward as possible, it was decided to make use of the softcopy provided on the newest edition of the *System/390 Redbooks Collection (SK2T-2177)*.

The following files are available:

DMSERP EXEC Sample CSL call to DMSERP (Extract/Replace).

DMSERP2 EXEC Use CSL Routine DMSERP to check YEAR2000_SUPPORT flag.

DMSEXIFI EXEC Sample CSL call to DMSEXIFI (check if file exists).

DMSPLUEX EXEC
Sample usage of DMSPLU MODULE (change CMS file datestamp).

DMSQEFL EXEC Use CSL Routine DMSQEFL (Query Functional Level) to get current levels of CP and CMS.

ESAMIGR SAMPLY2K
Local Modifications - for potential date issues.

HOLIDAYS EXEC Find on what days the statutory holidays for a given year will be taken in British Columbia.

ISPA2042 EXEC Sample ISPF bootstrap. Checks that VM level provides Y2K support, then checks to see if it is enabled.

ISPB2042 EXEC Sample ISPF dialog management routine. Performs the setup required for this ISPF session, then loops displaying this panel, and takes action as required.

ISPF2042 MACLIB
Special macro library used by most samples.

PLISMP1 EXEC Invoke sample PL/I Module for CSL calls. Performs the setup required for the module, then invokes it.

PLISMP1 PLIOPT Sample PL/I program to demonstrate a call to the CSL to obtain file update information, including the CENTURY indicator.

PLISMP2 EXEC Invoke sample PL/I Module for CSL Date & Time function calls.

PLISMP2 PLIOPT Sample PL/I program to demonstrate a call to the DateTimeSubtract function of CMS Multitasking Services. Uses home grown VMPLITMR COPY file.

SEEK EXEC Sample exec for scanning CMS files for a specified string.

TIME ZONES Timezone definitions and boundary conditions.

VMPLITMR COPY
Sample CSL call to VMTMDTS (DateTimeSubtract).

VMTMDTS EXEC Sample CSL call to VMTMDTS (DateTimeSubtract).

VMTMDTS2 EXEC
Sample CSL call to VMTMDTS (DateTimeSubtract).

Y2K EXEC Use new VM/ESA V2R2.0 YEAR2000 CMSFLAG.

Y2000 EXEC Sample use of Diagnose X'00' to check for Year2000 support.

The individual files can be be found on the **CD-ROM only**. To upload these samples to your VM host system, the following steps should be taken:

1. Open book SG242042 on the CD-ROM.
2. Select the samples shown in appendix E.2.
3. In **Services** select ***Copy ...***, pick the sections in which you are interested, and copy them to the Copy file.
4. Upload this file to VM.
5. The samples in the CMS file you have just created are separated by normal SCRIPT header statements (for example, E.2.1, E.2.2, ..., E.2.21).
6. Clean up all alignments which were caused by the different format used for online viewing; normally a simple XEDIT SHIFT is all that's required.

E.2 The Actual Samples

From here on printing is suppressed; that is, the output is only stored on the CD-ROM.

Appendix F. Special Notices

This publication is intended to assist customer and IBM technical personnel in the migration of their VM/ESA systems and application programs to a Year2000-ready state.

The information in this publication is not intended as the specification of any programming interfaces that are provided by VM/ESA or any of its related products. See the PUBLICATIONS section of the IBM Programming Announcement for VM/ESA and its related products for more information about what publications are considered to be product documentation.

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:

AFP
AS/400
AT
BookManager
CICS
DB2
DFSORT
DirMaint
ESCON
IBM
IMS
Language Environment
OfficeVision
OfficeVision/VM
OS/390
PROFS
QMF
RACF
RAMAC
RS/6000
SP
SQL/DS
System/390
SystemPac
Virtual Machine/Enterprise Systems Architecture
VM/ESA

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

dVMCF and VM Timing Facility Monitors are trademarks of MiraSoft Inc.

VM:ACCOUNT, VM:ARCHIVE, VM:BACKUP, VM:BATCH, VM:CADBACK, VM:CENTER, VM:DIRECTOR, VM:MANAGER, VM:MIGRATE, VM:MONITOR, VM:NOTEKEEPER, VM:OPERATOR, VM:SCHEDULE, VM:SECURE VM:SPOOL and VM:TAPE are registered trademarks of Sterling Software, Inc.

VM:DBA, VM:DB/ADMIN, VM:DB/MONITOR, VM:DB/REORGANIZER, VM:DB/RESTORE, VM:DB/EDITOR, VM:DB/REXX, VM:DB/REPORTER, VM:PROREXX, VM:VANTAGE and VM:WEBSERVER are trademarks of Sterling Software, Inc.

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

Appendix G. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 155.

- *Preparing Your VSE System for the Year 2000*, SG24-4932
- *Moving Your VSE System to the Year 2000 - Using IBM Tools*, SG24-2044; this redbook will be published in December 1997

G.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
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

G.3 Other Publications

These publications are also relevant as further information sources:

Year2000 Related Publications

- *The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation*, GC28-1251, on the World Wide Web at URL `http://www.software.ibm.com/year2000/resource.html`
- *IBM Year2000 Home Page*, on the World Wide Web at URL `http://www.ibm.com/year2000/`
- *VM/ESA Year2000 Home Page*, on the World Wide Web at URL `http://www.vm.ibm.com/year2000/`

VM/ESA Publications

- *VM/ESA Home Page*, on the World Wide Web at URL `http://www.vm.ibm.com/`
- *VM/ESA Introduction and Features Summary*, SC24-5746
- *VM/ESA Planning and Administration*, SC24-5750
- *VM/ESA Planning Dynamic I/O Configuration*, GC24-5695
- *VM/ESA Conversion Guide and Notebook*, SC24-5831
- *VM/ESA Service Guide*, SC24-5749

- *IBM Directory Maintenance VM/ESA: Tailoring and Administration Guide*, SC23-0533
- *VM/ESA CP Command and Utility Reference*, SC24-5773
- *VM/ESA CMS Users Guide*, SC24-5775
- *VM/ESA CMS Command Reference*, SC24-5776
- *VM/ESA CMS Application Multitasking*, SC24-5766
- *REXX/EXEC Migration Tool for VM/ESA*, GC24-5752
- *VM/ESA REXX/VM Reference*, SC24-5770

Other Publications

- *Input/Output Configuration Users Guide and ESCON Channel to Channel Reference*, GC38-0401
- *ISPF Dialog Management Guide and Reference*, SC34-4273
- *DB2 Server for VM System Administration*, GC09-2405
- *DB2 Server for VSE & VM SQL Reference*, SC09-2404
- *QMF Reference*, SC26-4716
- *IBM Language Environment for MVS and VM Migration Guide*, SC26-8232
- *IBM Language Environment for MVS and VM Programming Guide*, SC26-4818

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Numeric

2-digit-year format. A format that provides a year date as two digits only to represent a year within a specific century. The two high-order digits of the year are truncated; for example 1995 is represented as 95.

20th century. The period of time 0000.00 hrs 1901-January-1 through 2400.00 hrs 2000-December-31.

21st century. The period of time 0000.00 hrs 2001-January-1 through 2400.00 hrs 2100-December-31.

4-digit-year format. A format that provides a year date as four digits: the two high-order digits represent the century and the two low-order digits represent the year within the century. For example, 1995 represents the year 1995; 2095 represents the year 2095.

A

accuracy. A qualitative assessment of freedom from error, with a high assessment corresponding to a small error.

algorithm. An ordered set of well-defined rules for the solution of a problem in a finite number of steps.

application. A collection of software components used to perform specific types of user-oriented work on a computer.

application program. A program that is specific to the solution of an application problem. Synonymous with application software.

authority. The right to access objects, resources, or functions.

B

backup file. A copy of a file made for possible later reconstruction of the file.

batch processing. Pertaining to the technique of executing a set of computer programs such that each is completed before the next program of the set is started.

bit. Either of the digits 0 or 1 when used in the binary numeration system. Synonymous with binary digit.

blank. A part of a data medium in which no characters are recorded.

block I/O. Input/output operations on blocks of data stored in random locations.

bucket. One or more fields in which the result of an operation is kept.

buffer. A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another.

bulletin board. A graphic object that simulates a real-life bulletin board in that it displays text and graphic information in the form of messages to the user from client applications that are currently running.

byte. A string that consists of a number of bits, treated as a unit, and representing a character.

C

cancel. To end a task before it is completed.

catalog. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, and blocking factors.

CCYY format. A 4-digit-year format that uses two century digits (CC) to indicate the century and two year digits (YY) to indicate the year within the century. The CC representation is provided as either the actual century digits (for example, 18, 19, or 20) or as an encoded value (for example, as 00 to represent 19, 01 to represent 20 as in, 0095 represents the year 1995 and 0195 represents the year 2095.)

century. Although IBM recognizes that the 21st century begins at 0000 hrs, 2001-January-01, for purposes of this document, we are defining the 20th—21st century boundary to be between 2400 hrs, 1999-December-31 and 0000 hrs, 2000-January-1. This allows a discussion of the 21st century to include all dates with a 20yy format inclusive of the year 2000. Hence, the year 2100 is likewise relegated to the 22nd century.

century byte. The high order byte of a field used to contain the two high order digits of a 4-digit year. (For example, 19 in 1995, 20 in 2000 and 2001).

character data. Data in the form of letters and special characters such as punctuation marks.

character string. A string consisting solely of characters.

clause. In COBOL, an ordered set of consecutive COBOL character-strings whose purpose is to specify an attribute of an entry.

clock (CLK). A device that generates periodic, accurately spaced signals used for purposes such as timing, regulation of the operations of a processor or generation of interrupts.

CMS nucleus. The portion of CMS that usually resides in the user′s virtual storage when CMS is executing. Each CMS user receives a copy of the CMS nucleus at initial program load (IPL) of CMS.

collating sequence. An ordering assigned to a set of items, such that any two sets in that assigned order can be collated.

command. A statement used to request a function of the system. A command consists of the command name abbreviation, which identifies the requested function, and its parameters.

compatibility. The capability of a hardware or software component to conform with the interface requirements of a given data processing system without adversely affecting its functions.

compilation. Translation of a source program into an executable program (an object program).

compile. To translate all or part of a program expressed in a high-level language into a computer program expressed in an intermediate language, an assembly language, or a machine language.

compiler. A program that translates instructions written in a high-level programming language into machine language.

configuration file. A file that specifies the characteristics of a system or subsystem.

console. A part of a computer used for communication between the operator or maintenance engineer and the computer.

control block. A storage area used by a computer program to hold control information.

conversion. The process of changing from one form of representation to another; for example, to change from decimal representation to binary representation.

convert. To change the representation of data from one form to another, without changing the information they convey; for example, radix conversion, code conversion, analog to digital conversion, media conversion.

cosmetic. Referring to a 2-digit-year date that is viewed by human eyes only, such as a print date on hardcopy output or a date on a selection panel. Because it is neither read nor further processed by a program you might be able to exclude its modification from your Year2000 work effort.

CP command. In VM, a command by which a terminal user controls his virtual machine. The VM/390 control program commands are called CP commands. The CP commands that perform console simulation are called console functions.

cursor. A movable, visible mark used to indicate a position of interest on a display surface.

customization. The process of designing a data processing installation or network to meet the requirements of particular users.

cycle. An interval of space or time in which one set of events or phenomena is completed.

D

data administration. The performance of functions of specifying, acquiring, providing, and maintaining the data of an organization.

data dictionary. A centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. It assists management, database administrators, system analysts, and application programmers in planning, controlling, and evaluating the collection, storage, and use of data.

data interchange. The use of data by systems of different manufacture. See also data exchange.

data type. The mathematical properties and internal representation of data and functions. The four basic types are integer, real, complex, and logical.

database. A collection of interrelated data organized according to a database schema to serve one or more applications.

default. Pertaining to an attribute, condition, value, or option that is assumed when none is explicitly specified.

default format. A preset format that is automatically implemented unless the user specifies otherwise. Synonymous with basic format.

default value. A value assumed when no value has been specified. Synonymous with assumed value.

device type. The name for a kind of device sharing the same model number; for example, 3330, ECKD, 3400. Contrast with device class.

dictionary. A database of specifications of data and information processing resources.

digit. A character that represents a nonnegative integer; for example, one of the characters 0 through F in the hexadecimal numeration system. Synonymous with numeric character.

directory. An index that is used by a control program to locate one or more blocks of data that are stored in separate areas of a data set in direct access storage.

disk file. A set of related records on disk that are treated as a unit.

displacement. The distance from the beginning of a record, block, or segment to the beginning of a particular field.

disposition. In file processing, the process of specifying whether a file is new, old, or shared, and how the file is to be shared.

dump. (1) To record, at a particular instant, the contents of all or part of one storage device in another storage device. Dumping is usually for the purpose of debugging.

E

end user. A person, device, program, or computer system that utilizes a computer network for the purpose of data processing and information exchange.

entity. Any concrete or abstract thing of interest, including associations among things; for example, a person, object, event, or process that is of interest in the context under consideration, and about which data may be stored in a database.

entry. In programming languages, a language construct within a procedure, designating the start of the execution sequences of the procedure. A procedure may have more than one entry; each entry usually includes an identifier, called the entry name, and may include formal parameters.

error message. An indication that an error has been detected.

event. An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an input/output operation.

execution time. Any instant at which the execution of a particular computer program takes place.

external side. The receiver of a data entity. A module or routine that accepts a 2- or 4-digit-date format entity for further processing from another module or routine.

F

failure. The termination of the ability of a functional unit to perform its required function. Synonymous with malfunction.

feature. A part of an IBM product that may be ordered separately by the customer. A feature is designated as either special or specify and may be designated also as diskette-only.

file. A named set of records stored or processed as a unit.

file name. A name assigned or declared for a file.

fixed window. A technique to determine the century (high-order digits) of a year when represented by two digits. The 2-digit year is compared against a hardcoded threshold. The century designation is limited to a 100-year range spanning only two centuries. For example, assume the threshold is 60, then if the 2-digit year is ≥ 60, the year is in the 20th century; if the 2-digit year is <60, the year is in the 21st century.

flag. A variable indicating that a certain condition holds.

G

Gregorian calendar. Today's general-use calendar of 12 months and 365 days that employs the current leap year algorithm (refer to **Leap year** below).

group control system (GCS). A component of VM that provides multiprogramming and shared memory support to virtual machines. It is a saved system intended for use with SNA products.

H

hardware. All or part of the physical components of an information processing system, such as computers or peripheral devices.

hex. See hexadecimal.

hexadecimal. Pertaining to a system of numbers to the base 16; hexadecimal digits range from 0 through 9 and A through F, where A represents 10 and F represents 15.

hit. A comparison of two items of data that satisfies specified conditions. Contrast with match.

host system. A data processing system used to prepare programs and operating environments for use on another computer or controller.

I

implementation. The system development phase at the end of which the hardware, software and procedures of the system considered become operational.

initialize. To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

install. To add a program, program option, or software to a system in such a manner that it is runnable and interacts properly with all affected programs in the system.

instruction. A statement that specifies an operation to be performed by a system and that identifies data involved in the operation.

integer date. A count of days since a specified date. Various IBM software products have defined integer dates as follows:

Language/Product	Days Since
C	1969-Dec-31
COBOL	1600-Dec-31
Language Environment	1582-Oct-14
MVS/CICS/DB2	1899-Dec-31

integrity. The protection of systems, programs, and data from inadvertent or malicious destruction or alteration.

interactive. Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational, that is, a continuous dialog exists between user and system. Contrast with batch.

interface. A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions.

interpret. To analyze and execute each statement in a source program before translating and executing the next statement.

interrupt. A suspension of a process, such as execution of a computer program caused by an external event, and performed in such a way that the process can be resumed.

inverse. A matrix that results from a mathematical operation on another matrix such that the two matrices can be multiplied together to obtain the unit matrix.

invocation. The activation of a program or procedure.

J

job. A collection of related programs, identified by appropriate job control statements.

Julian date. As a general term used widely in computer programming and this document: A date in the format YYDDD. A date format that contains the year in positions 1 and 2, and the day in positions 3 through 5. The day is represented as 1 through 366, right adjusted, padded with zeroes on the left. For example, 1996-August-29 is 96242.

However, the above definition is accurately referred to as the **Ordinal Day of Year** date, and an accurate definition of **Julian Day Number** is as follows:

The astronomical system that counts the days since the First of January in the year 4713 BCE (the year −4712 before the common era). This scheme was invented by the astronomer Joseph Scaliger in the 16th century and named by him for his father Julius. The leap year reforms implicit in the new scheme were adopted at the command of Pope Gregory XIII in the year 1582 - hence the Gregorian Calendar. The Gregorian, or New Style, calendar was adopted in Britain and their colonies in September of 1752. (September of that year was missing 11 days. The 14th followed the 2nd.)

The remainder left when dividing the Julian Day Number by 7 indicates the day of week of the specified date. Zero corresponds to Monday, 1 to Tuesday, up through 6 for Sunday.

For example, 1996-Aug-29 is equivalent to Julian Day 2450325. Further, the hour of the day is expressed as a decimal such that 2450325.5 is midnight 1996-Aug-29, based on the fact that a Julian Day begins at mid-day (noon).

K

keyword. In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier.

keyword parameter. A parameter that consists of a keyword, followed by one or more values.

L

language. A set of characters, conventions, and rules that is used for conveying information.

Leap year. A year either evenly divisible by 400 **or** evenly divisible by 4 and not evenly divisible by 100. For example, the years 1700, 1800, 1900, and 1995 are

not leap years, but the years 1600, 1996, and 2000 are leap years.

library. A named area on disk that can contain programs and related information (not files). A library consists of different sections, called library members.

Lilian date. The number of days since 1582-October-14. 1582-October-15 is Lilian day 1, 1582-October-16 is Lilian day 2, and so on. (Named for Aloysius Lilius (an advisor to Pope Gregory XIII) who, together with his brother, constructed the current Gregorian calendar.)

link. In computer programming, the part of a program, in some cases a single instruction or an address, that passes control and parameters between separate portions of the computer program. Synonymous with linkage.

literal. In programming languages, a lexical unit that directly represents a value; for example, 14 represents the integer fourteen, ″APRIL″ represents the string of characters APRIL, 3.0005E2 represents the number 300.05.

log off. To end a session. Synonymous with log out.

log on. To initiate a session. Synonymous with log in.

logon. The procedure by which a user begins a terminal session.

loop. A sequence of instructions that is to be executed iteratively.

loop control. The parts of a loop that modify the loop control variables and determine whether to execute the loop body or exit from the loop.

M

macroinstruction. An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions. Synonymous with macro, macro statement.

maintenance. Any activity intended to retain a functional unit in, or to restore it to, a state in which it can perform its required function. Maintenance includes keeping a functional unit in a specified state by performing activities such as tests, measurements, replacements, adjustments, and repairs.

match. A comparison to determine identity of items. Contrast with hit

microcode. A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not program- addressable.

migrate. To move to a changed operating environment, usually to a new release or version of a system.

minidisk. Synonym for virtual disk.

mnemonic. The field of an assembler instruction that contains the acronym or abbreviation for a machine instruction. Using mnemonics frees the programmer from having to remember the numeric operator codes of the computer.

module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

monitor. Software or hardware that observes, supervises, controls, or verifies operations of a system.

multitasking. A mode of operation that provides for concurrent performance, or interleaved execution of two or more tasks.

N

nucleus. That part of a control program resident in main storage. Synonymous with resident control program.

numeric data. Data in the form of numerals and some special characters; for example, a date represented as 81/01/01. Synonymous with numerical data.

O

object code. Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

offset. The distance from the beginning of an object to the beginning of a particular field.

operand. That which is operated upon. An operand is usually identified by an address part of an instruction.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

operator. A person who keeps a system running.

option. A specification in a statement that may be used to influence the execution of the statement.

Ordinal Day of Year. See **Julian Date**

output file. A file that has been opened in order to allow records to be written.

overhead. In a computer system, the time, operations, and resources used for operating system functions, rather than for application programs.

override. A parameter or value that replaces a previous parameter or value.

P

packed decimal format. A format in which each byte in a field except the rightmost byte represents two numeric digits. The rightmost byte contains one digit and the sign; for example, the decimal value + 123 is represented as 0001 0010 0011 1100. Contrast with unpacked decimal format.

panel. A set of logically related information displayed on the screen for the purpose of communicating information to or from a computer user.

parameter. A variable that is given a constant value for a specified application and that may denote the application.

parameter list. A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

parse. In systems with time sharing, to analyze the operands entered with a command and create a parameter list for the command processor from the information.

partition. A fixed-size division of storage. For instance, main storage partition, virtual partition.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to the data stored within it.

performance. One of the two major factors, together with facility, on which the total productivity of a system depends. Performance is largely determined by a combination of throughput, response time, and availability.

pipe. A one-way communication path between a sending process and a receiving process. See also pipeline.

pipeline. In CMS Pipelines, a series of programs, called stages, each performing part of a task and passing the results to the next stage. Several parts of different tasks can be performed concurrently.

platform. The operating system environment in which a program runs.

post. To note the occurrence of an event.

preventive service. The installation of one or more program temporary fixes (PTFs) to avoid the occurrence of anticipated problems.

privilege class. In VM, one or more classes assigned to virtual machine user in the user′s VM directory entry; each privilege class specified allows access to a logical subset of all the CP commands.

procedure. In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call.

processor. In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic and logic unit.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

PROFILE EXEC. In VM, a special EXEC procedure with a filename of PROFILE. The procedure is normally executed immediately after CMS is loaded into a virtual machine. It contains CP and CMS commands that are to be issued at the start of every terminal session.

program product. Deprecated term for licensed program.

programmer. A person who designs, writes, and tests computer programs.

project management. The activities concerned with project planning and project control.

prompt. A visual or audible message sent by a program to request the user′s response.

Q

quality assurance (QA). All actions that are taken to ensure that a development or organization delivers products that meet performance requirements and adhere to standards and procedures.

query. In interactive systems, an operation at a terminal that elicits a response from the system.

queue. A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed.

R

recipient. The user to whom a message is addressed.

record. A set of one or more related data items grouped for processing.

record type. The classification of records in a file.

recovery. The act of resetting a system or data stored in a system to an operable state following damage.

reliability. The ability of a functional unit to perform a required function under stated conditions for a stated period of time.

reset. To put all or part of a data processing device back into a prescribed state.

resource. Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, processing unit, data sets, and control or processing programs.

restart. To resume the execution of a computer program using the data recorded at a checkpoint.

return code. A code used to influence the execution of succeeding instructions.

rolling window. Synonymous with **sliding window**.

routine. A program, or part of a program, that may have some general or frequent use.

run time. Synonym for execution time.

S

scanning. The systematic examination of data.

scratch. To erase data on a volume or delete its identification so that it can be used for another purpose.

screen image. In computer graphics, a pattern of points, lines, and characters displayed on the display surface of a video display terminal (VDT).

server. A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server.

service level. In SNADS, one of the four levels of service (fast, status, data high, or data low) that determines if a distribution is put on the normal or priority distribution queue.

session. In network architecture, for the purpose of data communication between functional units, all the activities which take place during the establishment, maintenance, and release of the connection.

shared file system. A part of CMS that lets users organize their files into groups known as directories and selectively share those files and directories with other users.

shutdown. The process of ending operation of a system or a subsystem, following a defined procedure.

simulation. The use of a data processing system to represent selected behavioral characteristics of a physical or abstract system.

sliding window. A technique to determine the century (high-order digits) of a year when represented by two digits. The user specifies the number of years (both past and future) within a 100-year window spanning two centuries. For example, assume the window is set at 19 future years (1996-2014) and 80 past years (1915-1994). Dates in the range 00–14 (inclusive) are designated 21st century dates because they fall into the future window. Dates in the range 15–99 (inclusive) fall into the 20th century.

softcopy. One or more files that can be electronically distributed, manipulated, and printed by a user. Contrast with hardcopy. Synonymous with machine-readable information (MRI), machine-sensible information.

software. All or part of the programs, procedures, rules, and associated documentation of a data processing system. Software is an intellectual creation that is independent of the medium on which it is recorded.

sort key. One or more keys within an item, used as a basis for determining the sequencing of items in a set. Synonymous with sequencing key.

source code. The input to a compiler or assembler, written in a source language. Contrast with object code.

source directory. The directory from which information is read. Contrast with target directory.

source program. A set of instructions written in a programming language that must be translated to machine language before the program can be run.

spool file. A file that contains output data that has been saved for later processing.

statement. In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.

subroutine. A group of instructions that can be part of another routine or can be called by another program or routine.

subset. A set each element of which is an element of a specified other set.

swapping. In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage.

syntax. The rules governing the structure of a language.

system. A computer and its associated devices and programs.

system configuration. A process that specifies the devices and programs that form a particular data processing system.

system customization. The process of specifying the devices, programs, and users for a particular information processing installation.

system date. The date assigned by the system operator during the initial program load procedure.

system default. A default value defined in the system profile.

system programmer. A programmer who plans, generates, maintains, extends, and controls the use of an operating system with the aim of improving overall productivity of an installation.

system software. Software that is part of or made available with a computer system and that determines how application programs are run; for example, an operating system. Contrast with application software.

system time. The elapsed time from the point at which a system was started to the current time. If system time is changed to local time when the system is started, the current system time is the local time of day.

T

terminal. A functional unit in a system or communication network at which data may enter or leave.

terminate. To stop execution of a program.

test. The operation of a functional unit and comparison of its achieved result with the defined result to establish acceptability; for example, a device test or program test.

time stamp. The value on an object that is an indication of the system time at some critical point in the history of the object.

timer. A register whose contents are changed at regular intervals in such a manner as to measure time.

token. In a programming language, a character string, in a particular format, that has some defined significance.

trace. The process of recording the sequence in which the statements in a program are executed and, optionally, the values of the program variables used in the statements.

U

user ID. User identification.

utility. A program designed to perform an everyday task such as copying data from one storage device to another.

V

virtual. Pertaining to a functional unit that appears to be real, but whose functions are accomplished by other means.

virtual machine (VM). In VM, a functional equivalent of a computing system. On the 370 Feature of VM, a virtual machine operates in System/370 mode. On the ESA Feature of VM, a virtual machine operates in System/370, 370-XA, ESA/370, or ESA/390 mode. Each virtual machine is controlled by an operating system. VM controls the concurrent execution of multiple virtual machines on an actual processor complex.

volume. A data carrier mounted and demounted as a unit; for example, a reel of magnetic tape, a disk pack.

W

windowing. The technique for partitioning a display surface into two or more distinct areas in which different texts can be displayed concurrently and manipulated separately or together.

Y

Year2000 challenge. The potential problems and its variations that might be encountered in any level of computer hardware or software from microcode to application programs, files, and databases that need to correctly interpret year-date data represented in 2-digit-year format caused by the transition to the year 2000.

Year2000 support. The ability to provide **Year2000 readiness**.

Year2000 transition. The process of revising systems and databases) to correctly process date data both within and between the 20th and 21st centuries.

YY format. Synonymous with **2-digit-year format**.

YYYY format. Synonymous with **4-digit-year format** and a subset of **CCYY format**.

List of Abbreviations

ABEND	Abnormal End
AFP	Advanced Function Printing
APAR	Authorized Program Analysis Report
API	Application Program Interface
BATCH	Background computer run
BLKSIZE	Block Size
CHPID	CHannel Path ID
CICS	Customer Information Control System
CMS	conversational monitor system
COBOL	COmmon Business Oriented Language
CP	Control Program
CPU	Central Processing Unit
CRC	Cross Reference Code
CSL	Callable Services Library
CUF	CMS Utilities Feature
CYL	Cylinder
DASD	Direct Access Storage Device
DB2	DATABASE 2
DCSS	DisContiguous Shared Segment
DD	Data Definition
DFSORT	Data Facility SORT
DIRMAINT	DIRectory MAINTenance Program
DSECT	Dummy Control SECTion
EC	Engineering Change
EDT	Eastern Daylight Time
EREP	Environmental Error Record Editing and Printing Program
ESA	Enterprise Systems Architecture
ESCON	Enterprise Systems Connection
ESM	External Security Manager
EST	Eastern Standard Time
FL	FileList
FM	File Mode
FST	File Status Table
GCS	Group Control System
GMT	Greenwich Mean Time
GUI	Graphical User Interface
I/O	Input/Output
IBM	International Business Machines
IMS	Information Management System
IOCP	I/O Configuration Program

IPL	Initial Program Load
ISO	International Organization for Standardization
ISPF	Interactive System Productivity Facility
ISQL	Interactive Structured Query Language
ITSO	International Technical Support Organization
LE	Language Environment
LPAR	Logically PARtitioned mode
LRECL	Logical RECord Length
MACLIB	MACro LIBrary
MVS	Multiple Virtual Storage
NLS	National Language Support
OS/390	IBM System/390 Operating System
PL/I	Programming Language 1
PL/1	Programming Language 1
PLU	Primary Logical Unit
PLIOPT	PL/1 Optimizing Compiler
PROP	PRogrammable OPerator
PTF	Program Temporary Fix
QMF	Query Management Facility
R/O	Read-Only
R/W	Read/Write
RACF	Resource Access Control Facility
RAMAC	Brand name and trademark of IBM DASD family
RC	Return Code
RELEASE	Major software update
REXX	REstructured eXtended eXecutor Language
RSCS	Remote Spooling Communications Subsystem
SCK	Set Clock
SFS	Shared File System
SIO	Start I/O
SNA	Systems Network Architecture
SPOOL	Simultaneous Peripheral Operation On-Line
SQL	Structured Query Language
STCK	Store Clock
SVC	SuperVisor Call instruction
SYSRES	SYStem RESidence file/disk
TOD	Time Of Day
TPF	Transaction Processing Facility
USERID	USER IDentification
UTC	Universal Time Coordinated
V=R	Virtual = Real
VM	virtual machine
VM/ESA	Virtual Machine/Enterprise Systems Architecture

VM/SP	Virtual Machine Facility/System Product
VMLIB	VM Library
VSE	Virtual Storage Extended
WWW	World Wide Web
XA	Extended Architecture
XC	Extended Configuration
XEDIT	Extended Editor

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Y

ITSO Redbook Evaluation

VM/ESA Year 2000 Migration - A Case Study
SG24-2042-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

Printed in U.S.A.